applied sciences

Article

Training Artificial Intelligence Algorithms with Automatically Labelled UAV Data from Physics-Based Simulation Software

Jonathan Boone [1], Christopher Goodin [2], Lalitha Dabbiru [2], Christopher Hudson [2], Lucas Cagle [2,*] and Daniel Carruth [2,*]

[1] Information Technology Laboratory, United States Army Engineer Research and Development Center, 3909 Halls Ferry Road, Vicksburg, MS 39180, USA
[2] Center for Advanced Vehicular Systems, Mississippi State University, Box 5405, Mississippi State, MS 39762, USA
* Correspondence: ldc290@msstate.edu (L.C.); dwc2@cavs.msstate.edu (D.C.)

Abstract: Machine-learning (ML) requires human-labeled "truth" data to train and test. Acquiring and labeling this data can often be the most time-consuming and expensive part of developing trained models of convolutional neural networks (CNN). In this work, we show that an automated workflow using automatically labeled synthetic data can be used to drastically reduce the time and effort required to train a machine learning algorithm for detecting buildings in aerial imagery acquired with low-flying unmanned aerial vehicles. The MSU Autonomous Vehicle Simulator (MAVS) was used in this work, and the process for integrating MAVS into an automated workflow is presented in this work, along with results for building detection with real and simulated images.

Keywords: artificial intelligence; machine-learning; smart trained models; convolutional neural networks; simulator; synthetic image data; human-labeled

check for updates

Citation: Boone, J.; Goodin, C.; Dabbiru, L.; Hudson, C.; Cagle, L.; Carruth, D. Training Artificial Intelligence Algorithms with Automatically Labelled UAV Data from Physics-Based Simulation Software. *Appl. Sci.* **2023**, *13*, 131. https://doi.org/10.3390/app13010131

Academic Editors: M. Jamal Deen, Subhas Mukhopadhyay, Yangquan Chen, Simone Morais, Nunzio Cennamo and Junseop Lee

Received: 28 November 2022
Revised: 13 December 2022
Accepted: 16 December 2022
Published: 22 December 2022

1. Introduction

Semantic segmentation using neural networks has enabled rapid advances in machine vision and scene understanding in recent years [1]. Acquiring adequate labeled training data is often the most significant challenge when using machine learning. Recent work has shown that using physics-based simulation can increase the amount and diversity of labeled training data while simultaneously reducing the cost and time required to collect and semantically label raw data [2]. The problem with current approaches that use simulation is that it is difficult to automatically create digital assets—synthetic terrains with desired characteristics—without a human in-the-loop. In this work, we implement a technique where the digital assets are created automatically and regenerated for different images, thereby introducing both automation and randomization into the training process.

To address and reduce the substantial labeled data requirements of modern machine learning techniques, active learning has become an area of intense research. Active learning in its traditional formulation allows a machine learning algorithm to query an oracle to label unsupervised data during training [3]. Current research in active learning is directed towards selecting the most useful sample to label automatically. However, recent work has reversed the selection problem: instead of selecting the best sample to label from a pre-generated set, the problem is instead that of selecting a sample to generate. This can be in the form of moving a drone for novel viewing perspectives [4] or even selecting simulation parameters used to generate synthetic data batches [5].

More recent work has shown that automated machine-learning can be achieved without human intervention in the model training phase [6]. However, these results were achieved by using publicly available datasets of cats and dogs (CIFAR-10) and everyday objects (ImageNet). As these labeled datasets still required a tremendous of time and manpower to acquire and label, the process is not truly automated. In contrast, this work describes a fully automated pipeline for both generation, labeling, and training machine learning.

Object detection in aerial images is a challenging task as some objects are only a few pixels wide, some objects are occluded, and some are in the shade. With the cost of drones decreasing, there is a surge in the amount of aerial data, so it will be useful if models can extract valuable features from the aerial data. Convolutional neural networks (CNN) are a very useful tool for machine learning applications [7]. The CNN architecture combines the benefits obtained by a standard neural network training with the convolution operation to efficiently classify images. Further, being a neural network, the CNN reduces the images into a form that is easier to process and scalable for large datasets without losing features critical towards a good prediction. Pooling layers reduces the spatial size by extracting a representative value which reduces computational cost. Fully connected layers produce the final classification with features from the previous layers resulting in the output. Figure 1 shows the schematic diagram of a basic CNN architecture.

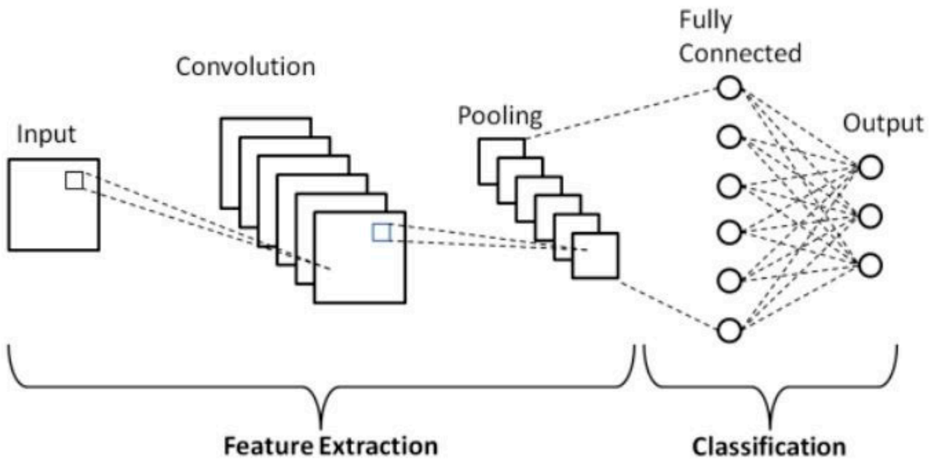

Figure 1. Schematic diagram of a basic convolutional neural network (CNN) architecture [7].

The R-CNN (Region Based Convolutional Neural Network) and faster R-CNN frameworks deal with the problem of efficient object localization in object detection [7]. These are based on two-stage detection where the first stage generates a sparse set of candidate object locations, and the second stage classifies each candidate location as one of the foreground and background classes using a CNN. As the two-stage detectors are slower, recent work focuses on one-stage detectors such as You Only Look Once (YOLO) [8] and single-shot multibox detector (SSD) [9]. The RetinaNet [10] one-stage object detection models have demonstrated promising results over existing single stage detectors.

In this work, we implemented the RetinaNet framework as its design features an efficient feature pyramid and uses anchor boxes which the model uses to predict the bounding box for an object. This aids in predicting the relative scale and aspect ratio of specific object classes. The model works well even with a limited training dataset and gives excellent detection accuracy.

The machine learning pipeline consists of data collection and/or generation, data preparation, data segregation, model training, and model evaluation. In data collection, large datasets are compiled that represent the population of classes to be labeled in the images. To prepare the data, the image features are labeled according to the class of the feature (e.g., building, vehicle, pedestrian, etc.). The datasets are then separated into a training dataset, used as input to the machine learning process to train the model, and a test dataset is used to evaluate the trained model's performance. The model is developed using machine learning techniques applied to the training dataset.

The resulting model performance is evaluated according to its ability to accurately label images in the test data. The model's label outputs are compared to the ground truth labels associated with the image. If the model is deemed sufficient to produce accurate predictions with new data, it may be deployed to production. However, if the model is deficient, it may be improved by improving the training dataset and by adjusting the parameters of the learning process. Different models will require different thresholds that are determined on a case-by-case basis.

Each of the steps listed above—data generation, data preparation, and model evaluation—can be automated. The first step in full automation is the generation of synthetic raw sensor data, which in turn requires automatically creating digital scenes to render. While [6] have achieved this to some degree, their empirical process nevertheless relied on field measurements.

In contrast, in this work the MSU Autonomous Vehicle Simulator (MAVS) [11] is used to automatically generate and label the training data during the simulation, avoiding the tedious and time-consuming process of hand-labeling data. Each object in the simulated scene was assigned a semantic label prior to the simulation. The final step in full automation is to automate the evaluation of the model. In a single iteration of the process, automation of the evaluation may simply calculate the Intersection over Union (IoU) to assess the accuracy of the model's labels. For an automated improvement process, the evaluation could assess the failures of the model and generate new training data to iteratively improve the model.

The process presented in this work is superior to existing approaches because the entire pipeline, including the scene generation step, is fully automated. The MAVS automatically generates synthetic scenes with the desired properties—in this case, pasture-like scenes with sparse vegetation and buildings—and creates truth-label data for training the learning algorithm.

2. Related Work

Automated machine-learning and data labeling have been used in a wide variety of processes in recent years. For example, [12] used memetic learning to tackle the capacitated arc routing problem. Shi et al. [13] used weakly supervised learning with a deep neural network to detect both rotational and reflectional symmetries. Additionally, Zhou, Wang, and Wan [14] used transfer learning with a variety of different CNNs to detect ore deposits in mining excavation.

In recent examples of CNN applications, Lu et al. [15] used a CNN with a cross-attention mechanism to detect defects in magnetic tiles while Win et al. [16] used a deep-feature interaction network to discover defects in aircraft engine blades. However, machine learning can also be used for more "soft-science" applications. For example, Qin et al. [17] used text extraction features along with deep learning to predict users' personality traits, while Shen et al. [18] used type-aware attentive path reasoning to model relation paths for knowledge graph completion.

Additionally, more theoretical problems can be addressed with deep learning. For example, Zhao et al. [19] recently used reinforcement learning to develop automated polices for online 3D bin packing. Furthermore, Zhou et al. [20] used Deconv blocks with CNN to show that RGB-D images could be used to construct an improved visual attention model. In contrast, Zhang et al. [21] used a depth-only model with particle filter optimization to produce 3D reconstruction with fast camera motion.

Deep learning can also be used in other 3D reconstruction processes. Li et al. [22] developed a novel neural network architecture for the synthesis of different 3D shapes. Similarly, Ban et al. [23] used multimodal histograms to align images during surgery, while Yang et al. [24] used a self-supervised stereo reconstruction framework to optimize deformable soft tissue surfaces. Finally, Huang et al. [25] used a dual-graph attention convolution network for object classification with 3D point clouds.

3. Automated Data-Generation and Labeling

The MAVS [11] was used to generate synthetic imagery for this work. An open-source software library for simulating the sensors, environment, and vehicle dynamics of autonomous ground vehicles (AGV), MAVS was created to provide real-time, physics-based simulation capability in a modular, customizable architecture that can be integrated with a variety of other systems and software like the Robotic Operating System (ROS). The MAVS uses ray-tracing to generate realistic sensor data for Light Detection and Ranging (lidar) sensors, cameras, and global-positioning system (GPS) [26]. It features a C++ API, as well as a Python interface to the API that allows for rapid development of simulated experiments.

MAVS is optimized for simulating AGV operating in off-road environments. Factors that influence off-road driving include soil type and strength, vegetation type and density, terrain slope and terrain roughness. The simulator includes physics-based models to account for all these off-road features. The lidar simulation accurately accounts for beam-scattering in dense vegetation, while the vehicle simulation accounts for soft-soil and surface roughness effects. In addition, MAVS can automatically create off-road scenes and generate semantically labeled lidar point clouds and camera images. It can also account for weather and environmental affects like rain, snow, dust, fog, and time-of-day, as shown in Figure 2.

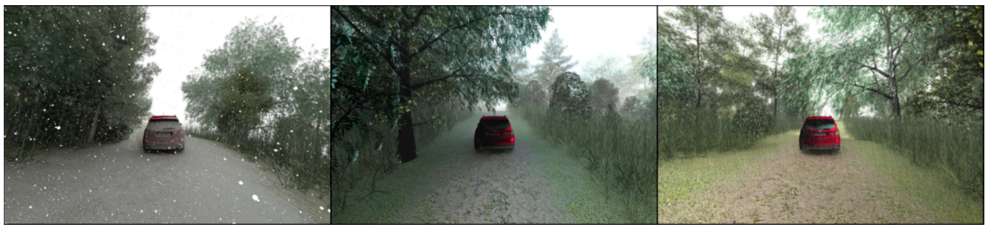

Figure 2. Simulation of snow (**left**), fog (**middle**), and rain (**right**) in MAVS. The terrain was created automatically with MAVS.

MAVS has been used for a variety of applications ranging from developing algorithms for automatically removing rain in images [1], to optimizing the placement and orientation of lidar sensors on an AGV [27], to assessing the impact of vegetation on autonomous navigation [28]. The past success of MAVS in supporting machine-learning studies make it a good candidate simulator for this research.

Three primary simulation components are required in order to generate realistic synthetic imagery:

1. Scene Generation–A model of the geometry, texture, and color of the surfaces in the environment;
2. Radiative Transfer–A model of how light reflects and propagates through the environment; and
3. Camera Simulation–A model of how the camera captures and processes light.

For this project we will use physics-based simulation to develop synthetic images. In this case, physics-based refers to all three phases of the simulation. In the following sections, each of these three components will be discussed in more detail.

3.1. Scene Generation

For this project, we chose scenes which are primarily off-road and/or rural in nature. The most prominent features in most outdoor scenes are the terrain slope and roughness and the type, size, and distribution of vegetation. In MAVS, outdoor scenes are automatically created using a multi-scale roughness model for the surface and a simulated growth/competition model for estimating the distribution of the vegetation.

The surface roughness is defined by two decades of Perlin noise [29], a spatially coherent noise model that is widely used in computer graphics and simulation. The first decade of noise has a wavelength of 10 m and simulates the appearance of rolling hills in the terrain. The second decade has a wavelength of 0.3 m and simulates the appearance and effect of surface roughness on passenger vehicles. In the MAVS software, the user defines the magnitude of each decade independently, allowing the roughness of the terrain to be adjusted independently of the rolling hills.

In order to simulate natural-appearing vegetation distributions, MAVS simulates years of plant growth and competition for the ecosystem being studied. MAVS implements the competition model described in [30].

MAVS uses an ecosystem definition input file where the user specifies the type of plants in the ecosystem, their maximum size and lifetime, regeneration rate, and growth rate. MAVS simulates 10–20 years of growth and competition and saves the resulting scene file for use in sensor simulations.

3.2. Radiative Transfer

Radiative transfer is the propagation of light (both visible and outside the visible spectrum) through a scene or environment. In computer graphics, radiative transfer simulations refer the class of techniques used to solve the rendering equation [31]. While many techniques of varying fidelity have been proposed for radiative transfer, path-tracing [32] is among the most popular because path-tracing calculations can theoretically converge upon an exact solution to the rendering equation. MAVS uses path-tracing to simulate the transport of light and interaction with different surfaces in a scene.

For outdoor scenes, MAVS treats the entire sky as a light source, using the Hosek-Wilkie sky model to account for the sky color based on the time of day, position of the sun, and aerosol content in the air [33]. This results in shadows with realistic soft-edges and appropriate color-tinting based on sky conditions. Path-tracing also gives reduced aliasing effects when compared to primary raytracing. The primary drawback of path-tracing is the computational cost. Mitigation techniques for slow image generation will be discussed in the Section 4 "Methods for Generating Randomized UAV Images".

3.3. Camera Simulation

The two primary components of a digital camera are the optics (usually a combinations of lenses) and the sensor at the focal plane array, typically a CCD or CMOS array. In MAVS, the lens system is simulated using a radial distortion model [34,35]. An example of the effect of distortion is shown in Figure 3. The corresponding distortion parameters are shown in Table 1, along with their values in this example.

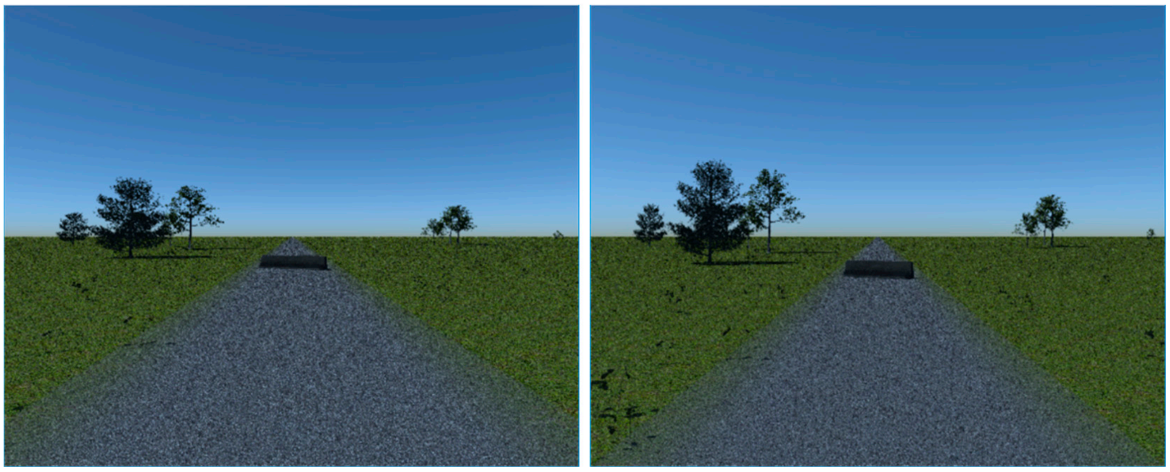

Figure 3. Example of the effect of distortion in the MAVS camera simulation. (**Left**) Distorted image. (**Right**) Undistorted image.

Table 1. Example parameters in the MAVS camera distortion model.

Parameter	Symbol	Value (Pixels)
Focal Length	(F_u, F_v)	(657.30, 647.74)
Principal Point	(C_u, C_v)	(307.72, 242.33)
Skew	α	0.00042
Radial Distortion	(R_1, R_4, R_6)	(307.72, 242.33)
Tangential Distortion	(T_1, T_0)	(−0.00028, 0.00005)

Note that in the aerial camera simulated for this project, there was no simulated distortion.

The incident model on the focal plane array is modeled using the Phong model with diffuse term

$$I_p = k_d\left(\vec{L}\cdot\vec{N}\right)i_d + k_s\left(\vec{R}\cdot\vec{V}\right)i_s$$

In this equation, I_p is the illumination at a given point, i_d is the incdient diffuse irradiance at that point, i_s is the incident specular irradiance, R is the specular direction, N is the surface normal, L is the direction from the point to the light source, and V is the vector pointing to the viewer. The material properties k_d and k_s specify the diffuse and specular reflectances. These have the constaint that they must both be positive and sum to less than 1.0 ($k_d + k_s < 1.0$). In the MAVS simulator, the specular component is set to zero for most natural surfaces and the the diffuse component is estimated from field data.

For each pixel, the total contribution of incident ration from all points in the scene is estimated using path tracing. The response of the focal plane array is then modeled using a power law model.

$$I_{ij} = AL_{ij}^{\gamma}$$

where I_{ij} is the output intensity of pixel (i,j), L_{ij} is the irradiance on pixel (i,j), γ is the compression factor, and A is the gain. The compression factor (γ) is usually set to less than one. The effect of range compression is shown in Figure 4, which demonstrates how the dynamic range of the color and intensity of the image is reduced when decreasing γ.

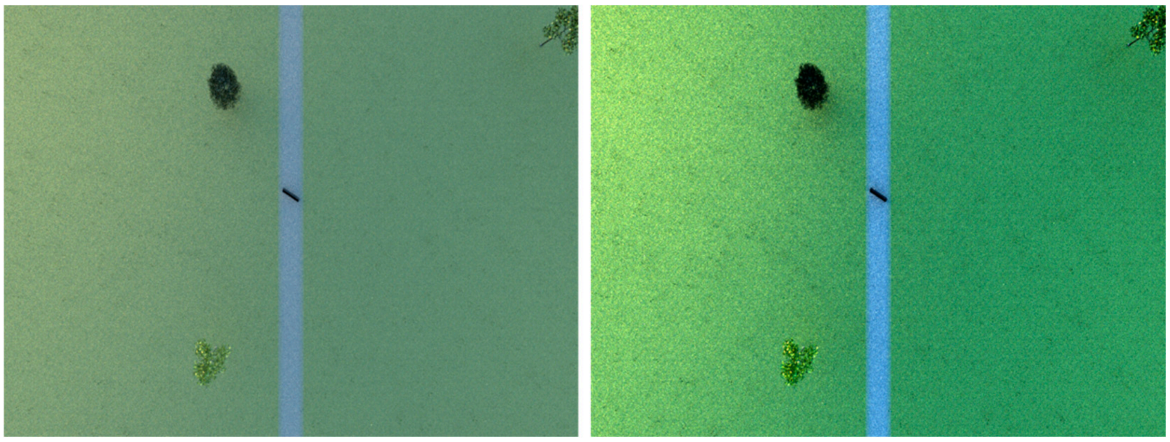

Figure 4. Comparison of γ = 0.5 (**left**) to γ = 1.0 (**right**).

In the experiments shown in this work, the radial and tangential distortion were set to zero and the compression factor was γ = 0.6.

4. Methods for Generating Randomized UAV Images

In this task, our goal was to develop methods for randomly generating synthetic data that incorporated the most relevant features of real data for machine learning. In order to achieve this, we decided to use a sample real dataset to compare to our synthetic set.

The real-world data were provided by MSU's Geospatial Research Institute (GRI) and are discussed below.

4.1. Real Data

Real images were acquired using the DJI Phantom 4 flown over the H. H. Leveck Animal Research Center, often referred to as South Farm. The data were acquired in July 2019 with the built-in Phantom-4 Pro RGB camera. The camera has a resolution of 4864×3648 pixels, focal length of 8.8 mm and a 70° field-of-view. The flight altitude was 122 m (400 feet) and the flight speed was around 7.1 m/second. A total of 119 images were analyzed for this project. The images consist primarily of open fields and pastureland bordered by both mature and scrub trees. In addition, the images contained some residential areas, barns, and other farming structures, as well as paved and dirt roads. An orthomosaic created from all 119 images using Agisoft Metashape Professional Edition with overlap and sidelap average of 80% is shown in Figure 5.

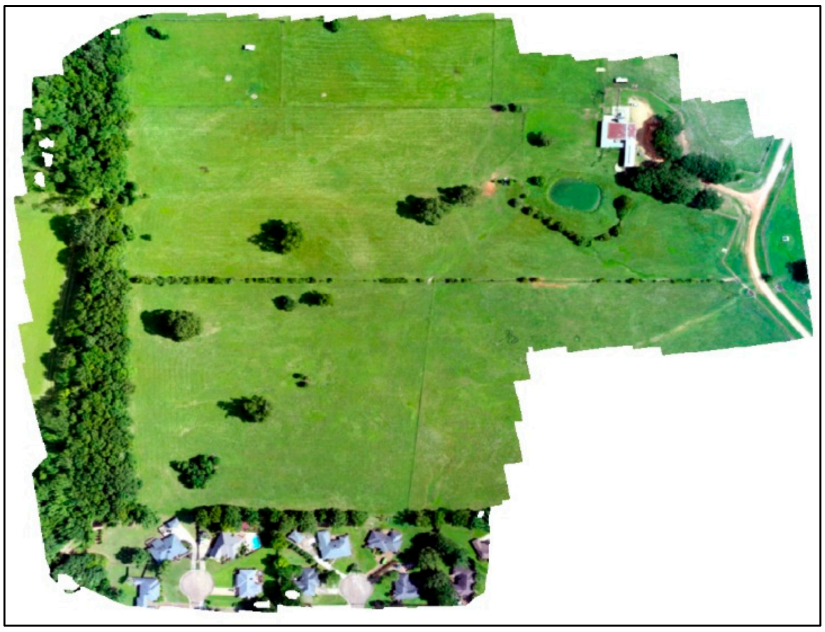

Figure 5. Mosaic created from 119 images of the MSU South Farm.

4.2. Simulated Data

Data generation is considered as a three-step process of sampling from multi-variate distributions: one each for scene, environment, and camera parameters. As scene and environment generation is computationally expensive, scenes are reused for multiple environments, which are in turn reused for multiple cameras from which one image is generated. Thus, the total number of generated images in a generation run can be described as $N = S \times E_s \times C_E$ where S is the number of scenes, E_S the number of environments per scene, and C_E the number of cameras per environment.

4.2.1. Scene, Environment, and Camera Sampling

Scenes are randomly generated by MAVS. The statistics and settings used are themselves sampled from configurable distributions. This allows scene generation to be tailored. Currently, three such of the settings exposed by MAVS can be sampled from random distributions: low frequency surface noise, high frequency surface noise, and plant density. A similar scheme is in place for environment settings. Time of day (hour), rain rate, and cloud cover can all be sampled from random distributions to ensure a spread of different environments are used. Cameras are placed facing down at an elevation of 200 m above a random point in the scene.

4.2.2. Simulated Camera

To generate images sufficiently realistic for sim-to-real transfer, a path tracing camera is essential. The most crucial parameter of such a camera is the number of rays per pixel used to generate the image. The number of rays used has a direct effect on the time required to render an image, so a tradeoff exists between image quality and render time.

Performance tradeoff studies were performed to quantify and gauge the consequences of rays on image quality and render time. The same scene was rendered using a broad spread of rays and the render time and image error were used to compare images. Image error was computed with respect to the highest ray count image, representing the closest sample to a ground-truth image.

4.2.3. Denoising

In an additional effort to reduce render time without impacting quality, Intel's Open Image Denoising Network (OIDN) was tested and ultimately incorporated. OIDN is a network, with accompanying API and binaries, trained to denoise images rendered with insufficient rays [36]. This allows fewer rays to be used initially. The raw images are too noisy to be useful but can be denoised by the network to produce a result of notably better quality. The network's inference time is substantially faster than the equivalent number of rays necessary to produce the higher quality image. The same experiment as used to evaluate the number of rays for raw images was repeated with the addition of using OIDN. Image error was still computed using error with respect to the highest ray un-denoised image. The runtime of the OIDN network was included in the render time of the denoised images to properly provide overall runtime comparisons.

4.2.4. Results

The mean-squared error (MSE) versus number of rays per pixel for both raw and denoised images are shown in Figure 6 with a logarithmic scale for rays. At low ray counts, the denoised images display higher error than the corresponding raw images. However, the denoised images display substantially reduced error roughly between 2 and 200 rays. Ray counts above this range become too computationally expensive for the marginal improvements in image quality. This can be seen in Figure 7, which displays the raw and denoised images for 1, 100, and 10,000 rays. Qualitatively, the denoised 100 ray image is sufficiently like the 10,000-ray raw image to be used in training of machine learning algorithms.

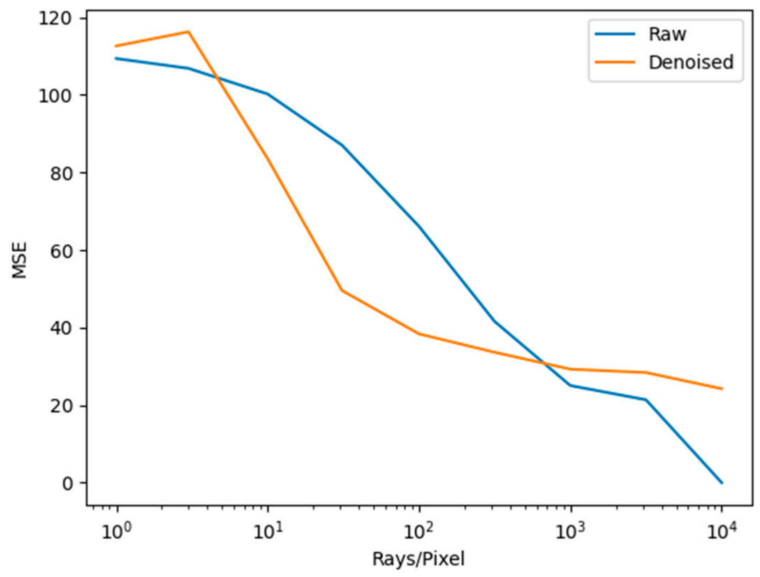

Figure 6. Mean squared error (MSE) of images, raw and denoised.

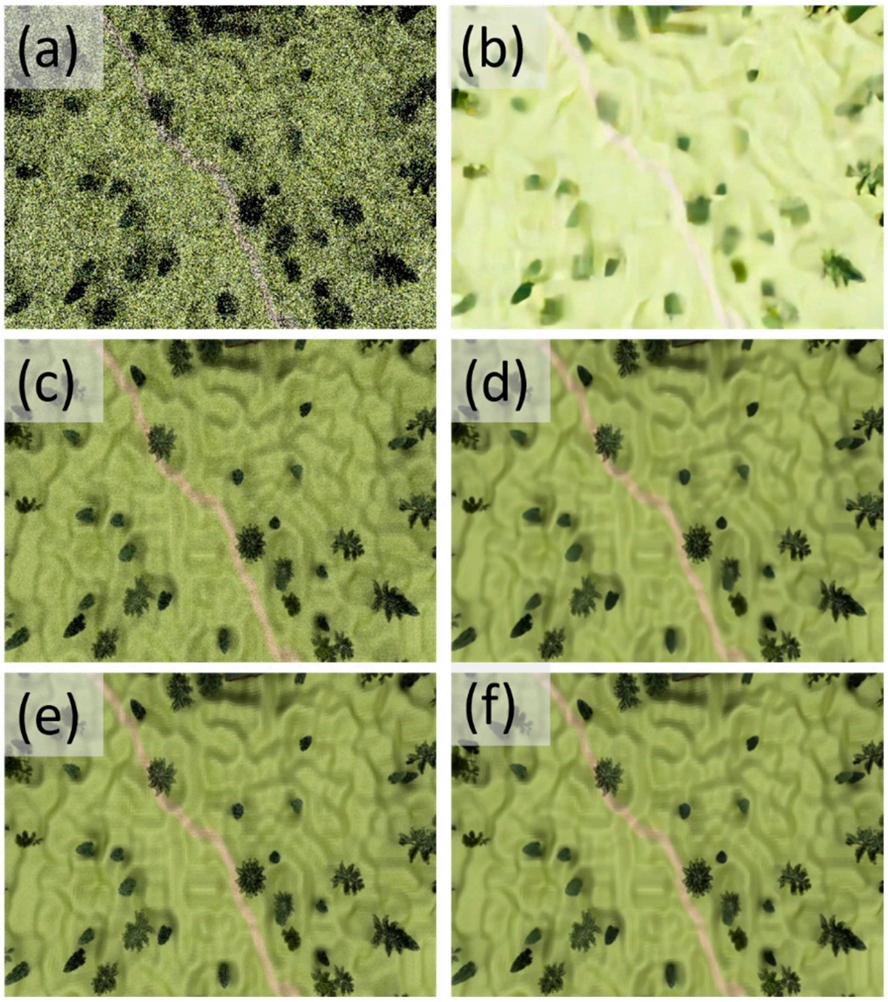

Figure 7. Sample Raw Images on the left (**a**,**c**,**e**) and Denoised Images on the right (**b**,**d**,**f**). Images (**a**,**b**) 1 rays per pixel, (**c**,**d**) 100 rays per pixel, and (**e**,**f**) 10,000 rays per pixel.

5. Environment Features for Labeling and Feature Classifications

Consideration of terrain features includes pavement, gravel, dirt, trees, grass, vehicles, and pedestrians. After our initial assessment of the real dataset, semantic classes were selected to provide adequate discretization and coverage of the data. The initial set of semantic classes included grass, soil, water, trees, paved and unpaved roads, fences, shrubs, and a class for miscellaneous manmade objects. An example image labeled with these classes is shown in Figure 8.

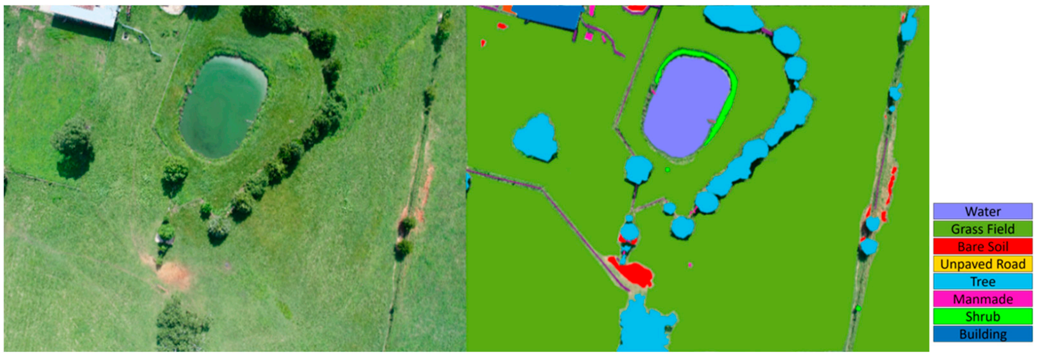

Figure 8. Example of semantic labeling of real-world data.

However, we discovered that there were limitations with our selection of semantic labels. First, because of the high-resolution and detail of the real images, each image took about 30 min to semantically label. Second, the boundaries between many of the classifications such as soil, grass, and unpaved road were unclear, making consistent labeling difficult.

Therefore, because our primary goal in this work was to develop the automation process, rather than develop new techniques in machine learning or investigate optimal methods for hand-labeling sensor data, we chose to simplify the labeling process by focusing only on buildings. This allowed us to use box-labels, rather than pixel labels. In addition, since there only a few buildings per image, the average labeling time was reduced to only a few minutes per image. An example box-labeled image is shown in Figure 9.

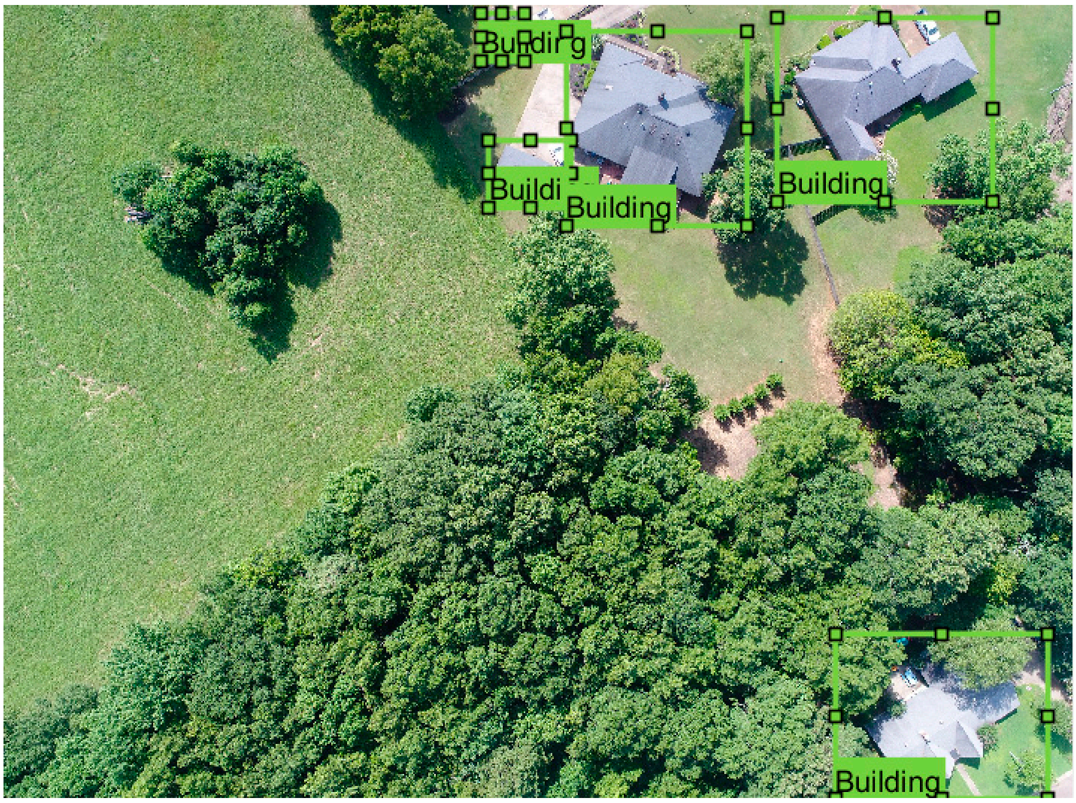

Figure 9. Example box-labeling of real-world data.

Of the 119 real images, 37 had at least one building, while the remainder did not have buildings. Of those images with buildings, 16 had exactly one building, while the other 21 had 2–8 buildings.

For this project, we used the MATLAB Image Labeler [37] to perform the labeling tasks. Although we considered other solutions for performing the labeling, MATLAB was chosen because we could perform the labeling on our local machines without uploading the data to a 3rd-party server.

6. Object Detection Modeling

Training data were generated with MAVS. Scenes were automatically generated with random degrees of surface roughness. The buildings and trees in each scene were randomly oriented and scaled to produce a wide range of different training data. In addition, some vehicles were placed in the scenes in random positions and orientations. For example, some red vehicles can be seen in bottom right corner Figure 10a.

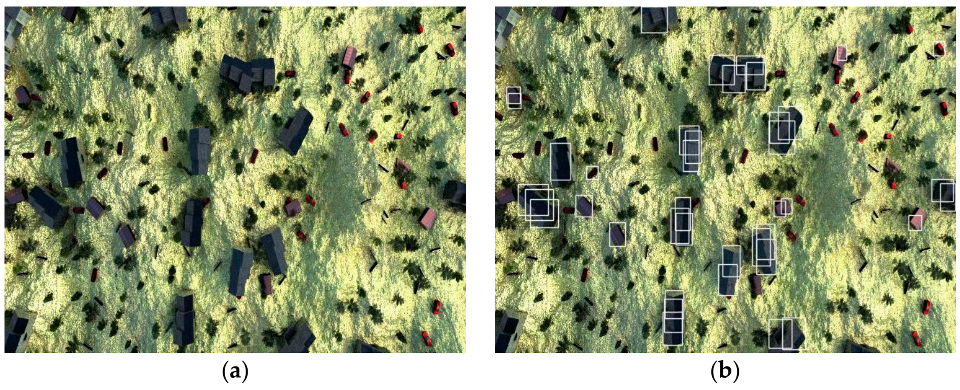

Figure 10. (**a**) Simulated UAV original image. (**b**) Detection of "buildings" using RetinaNet, marked in white boxes.

To match the weather conditions of the real data, the MAVS simulations used mid-day conditions with no cloud cover, rain, or fog. The camera altitude was 122 m, like the real camera. The simulated camera had a horizontal field-of-view of 60.5°. Simulations were run on Mississippi State University's supercomputer, Shadow, a Cray CS300-LC on multiple Xeon-Phi cores.

Object detection in aerial images is a challenging task as some objects are only a few pixels wide. We have implemented keras-retinanet on the simulated aerial data generated using MAVS. RetinaNet is one of the best single stage object detection models that has proven to work well with dense and small-scale objects [38]. Moreover, it is one of the popular object detection models to be used with aerial imagery. RetinaNet makes use of keras, tensorflow, tensorboard, CUDA, CUDNN, and several other Python libraries.

Keras-RetinaNet uses feature pyramid network (FPN) and focal loss for training. FPN is a structure for multiscale object detection which combines low resolution, high resolution, and semantically weak features in a top down pyramid architecture [37]. The focal loss is designed to address the problems with imbalance where there are many background classes and a few foreground classes.

The simulated aerial dataset consists of 871 aerial images of 640 $\times$ 480 resolution as shown in Figure 10a. Keras-Retinanet framework with ResNet 50 as backbone was implemented for object detection. The training dataset has 696 images with bounding box annotations and labels in PASCAL VOC format. The model was evaluated with 175 images in the validation dataset. Anchor boxes are fixed size boxes that the model uses to predict the bounding box of an object. The hyperparameters used in the model are the pixel size of the anchor box, the distance between the centers of two neighboring anchor boxes (strides), and the height/width ratio of the box to predict a relative scale of the object.

In this preliminary work, we detected "buildings" in the aerial imagery. Each location on a given feature map has nine anchor boxes (at three scales and three ratios). The model is trained using 15 epochs and 500 steps. The metrics used to evaluate the model are mAP (mean Average Precision) and IoU (Intersection of Union). The mean average precision value is the average AP over all classes and the IoU indicates how much bounding boxes overlap, which is crucial in determining the true positives and false positives. Our results achieved a mAP accuracy of 88% at IoU = 0.3. As shown in Figure 10b, the algorithm detected each roof section as a building. The detected image has one false positive, where a truck on the top right was detected as a "building". As seen in Figure 10a, the building roofs are in blackish gray except a few roofs with red color. There were very few images with red roof structures and the algorithm detected a part of these structures too. Overall, the deep learning network successfully detected objects in the aerial imagery.

7. Conclusions

We implemented the RetinaNet framework as its design features an efficient feature pyramid and uses anchor boxes, which the model uses to predict the bounding box for an object. This aided in predicting the relative scale and aspect ratio of specific object classes. The model worked well even with a limited training dataset and gives excellent detection accuracy.

Our goal was to develop methods for randomly generating synthetic data that incorporated the most relevant features of real data for machine learning. In order to achieve this, we decided to use a sample real dataset to compare to our synthetically generated dataset. We point out that the real data is not necessary to run the automated pipeline. Nevertheless, incorporating real data as a baseline provided several benefits to this project.

The automated architecture is currently under development to facilitate training and testing of machine learning algorithms [39]. This pipeline will be broken into two sub-tasks; training and testing. The training pipeline will leverage MAVS environments in order to generate image data sets. These data sets will be used to train specific algorithms, which will be tested against a subset of data for accuracy. Should the accuracy of the algorithms fall below an acceptable level, additional data will be selected for generation by MAVS based on meta data tags, used to identify classes which are under performing.

Planned improvements to the machine learning algorithm include running the algorithm on a higher GPU machine, training with more simulated data, and increasing the number of epochs to achieve better accuracies of detection.

Author Contributions: J.B., conceptualization, writing, idea proposal, resources, submission, and preparation; C.G., conceptualization, writing, data curation, software development, investigation; L.D., software development, data curation, editing, and visualization; C.H., data curation, writing, editing, and visualization; L.C., writing data curation and visualization; D.C., review, writing, results. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the United States Department of Army, Research, Development, Test & Evaluation, Army; High Performance Computing Modernization Program, Program Element Number 0603461A.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: This study did not report any publicly available data.

Acknowledgments: This study was conducted for the US Army Engineer Research and Development Center's (ERDC) Information Technology Laboratory (ITL) under the High Performance Computing Modernization Program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sharma, S.; Hudson, C.; Carruth, D.; Doude, M.; Ball, J.E.; Tang, B.; Dabbiru, L. Performance analysis of semantic segmentation algorithms trained with JPEG compressed datasets. *Real-Time Image Process. Deep. Learn.* **2020**, *2020*, 1140104.
2. Goodin, C.; Sharma, S.; Doude, M.; Carruth, D.; Dabbiru, L.; Hudson, C. *Training of Neural Networks with Automated Labeling of Simulated Sensor Data*; SAE Technical Paper; SAE: Warrendale, PA, USA, 2019.
3. Budd, S.; Robinson, E.C.; Kainz, B. A Survey on Active Learning and Human-in-the-Loop Deep Learning for Medical Image Analysis. *arXiv* **2019**, arXiv:1910.02923. [CrossRef] [PubMed]
4. Teng, E.; Iannucci, B. Learning to Learn in Simulation. *arXiv* **2019**, arXiv:1902.01569.
5. Ruiz, N.; Schulter, S.; Chandraker, M. Learning to Simulate. *arXiv* **2018**, arXiv:1810.02513.
6. Chen, M.; Feng, A.M.; Prasad, P.B.; Soibelman, L.; Enloe, M. Fully Automated Photagrammetric Data Segmentation and Object Information Exraction Approach for Creating Simulation Terrain. In Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Orlando, FL, USA, 28 November–2 December 2020; pp. 1–12.
7. Phung, V.H.; Rhee, E.J. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Appl. Sci.* **2019**, *9*, 4500. [CrossRef]
8. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
9. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.

10. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
11. Hudson, C.; Goodin, C.; Miller, Z.; Wheeler, W.; Carruth, D. Mississippi State University Autonomous Vehicle Simulation Library. In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium, Novi, MI, USA, 12–14 October 2020; pp. 11–13.
12. Cao, B.; Zhang, W.; Wang, X.; Zhao, J.; Gu, Y.; Zhang, Y. A memetic algorithm based on two_Arch2 for multi-depot heterogeneous-vehicle capacitated arc routing problem. *Swarm Evol. Comput.* **2021**, *63*, 100864. [CrossRef]
13. Shi, Y.; Xu, X.; Xi, J.; Hu, X.; Hu, D.; Xu, K. Learning to detect 3D symmetry from single-view RGB-D images with weak supervision. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**. [CrossRef]
14. Zhou, W.; Wang, H.; Wan, Z. Ore Image Classification Based on Improved CNN. *Comput. Electr. Eng.* **2022**, *99*, 107819. [CrossRef]
15. Lu, H.; Zhu, Y.; Yin, M.; Yin, G.; Xie, L. Multimodal Fusion Convolutional Neural Network with Cross-attention Mechanism for Internal Defect Detection of Magnetic Tile. *IEEE Access* **2022**, *10*, 60876–60886. [CrossRef]
16. Yin, M.; Zhu, Y.; Yin, G.; Fu, G.; Xie, L. Deep Feature Interaction Network for Point Cloud Registration, With Applications to Optical Measurement of Blade Profiles. *IEEE Trans. Ind. Inform.* **2022**. [CrossRef]
17. Qin, X.; Liu, Z.; Liu, Y.; Liu, S.; Yang, B.; Yin, L.; Liu, M.; Zheng, W. User OCEAN Personality Model Construction Method Using a BP Neural Network. *Electronics* **2022**, *11*, 3022. [CrossRef]
18. Shen, Y.; Ding, N.; Zheng, H.T.; Li, Y.; Yang, M. Modeling relation paths for knowledge graph completion. *IEEE Trans. Knowl. Data Eng.* **2020**, *33*, 3607–3617. [CrossRef]
19. Zhao, H.; Zhu, C.; Xu, X.; Huang, H.; Xu, K. Learning practically feasible policies for online 3D bin packing. *Sci. China Inf. Sci.* **2022**, *65*, 1–17. [CrossRef]
20. Zhou, W.; Lv, Y.; Lei, J.; Yu, L. Global and local-contrast guides content-aware fusion for RGB-D saliency prediction. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *51*, 3641–3649. [CrossRef]
21. Zhang, J.; Zhu, C.; Zheng, L.; Xu, K. ROSEFusion: Random optimization for online dense reconstruction under fast camera motion. *ACM Trans. Graph. (TOG)* **2021**, *40*, 1–17.
22. Li, J.; Xu, K.; Chaudhuri, S.; Yumer, E.; Zhang, H.; Guibas, L. Grass: Generative recursive autoencoders for shape structures. *ACM Trans. Graph. (TOG)* **2017**, *36*, 1–14. [CrossRef]
23. Ban, Y.; Wang, Y.; Liu, S.; Yang, B.; Liu, M.; Yin, L.; Zheng, W. 2D/3D Multimode Medical Image Alignment Based on Spatial Histograms. *Appl. Sci.* **2022**, *12*, 8261. [CrossRef]
24. Yang, B.; Xu, S.; Chen, H.; Zheng, W.; Liu, C. Reconstruct Dynamic Soft-Tissue With Stereo Endoscope Based on a Single-Layer Network. *IEEE Trans. Image Process.* **2022**, *31*, 5828–5840. [CrossRef]
25. Huang, C.Q.; Jiang, F.; Huang, Q.H.; Wang, X.Z.; Han, Z.M.; Huang, W.Y. Dual-Graph Attention Convolution Network for 3-D Point Cloud Classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–13. [CrossRef]
26. Dabbiru, L.; Goodin, C.; Scherrer, N.; Carruth, D. LiDAR Data Segmentation in Off-Road Environment Using Convolutional Neural Networks (CNN). *SAE Int. J. Adv. Curr. Prac. Mobil.* **2020**, *2*, 3288–3292.
27. Hudson, C.; Goodin, C.; Doude, M.; Carruth, D.W. Analysis of dual lidar placement for off-road autonomy using MAVS. In Proceedings of the 2018 World Symposium on Digital Intelligence for Systems and Machines, Košice, Slovakia, 23–25 August 2018; pp. 137–142.
28. Foroutan, M.; Tian, W.; Goodin, C.T. Assessing Impact of Understory Vegetation Density on Solid Obstacle Detection for Off-Road Autonomous Ground Vehicles. *ASME Lett. Dyn. Syst. Control* **2020**, *1*, 021008. [CrossRef]
29. Perlin, K. Improving Noise. In Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, San Antonio, TX, USA, 23–26 July 2002; pp. 681–682.
30. Lane, B.; Prusinkiewicz, P. Generating Spatial Distributions for Multilevel Models of Plant Communities. *Graph. Interface* **2002**, 69–87.
31. Kajiya, J.T. The Rendering Equation. In Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, Dallas, TX, USA, 18–22 August 1986; pp. 143–150.
32. Jensen, H.W. Importance Drivin Path Tracing Using the Photon Map. In *Eurographics Workshop on Rendering Techniques*; Springer: Vienna, Austria, 1995; pp. 326–335.
33. Hosek, L.; Wilkie, A. An Analytic Model for Full Spectral Sky-dome Radiance. *ACM Trans. Graph. (TOG)* **2012**, *31*, 1–9. [CrossRef]
34. Zhang, Z. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; pp. 666–673.
35. Heikkila, J.; Silven, O. A four-step camera calibration procedure with implicit image correction. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kerkyra, Greece, 20–27 September 1997; pp. 1106–1112.
36. Intel. Open Image Denoise. Retrieved from Intel Overview. 2022. Available online: https://www.openimagedenoise.org/ (accessed on 9 December 2022).
37. MATLAB. Image Labeler. Retrieved from MATLAB Help Center. 2020. Available online: https://www.mathworks.com/help/vision/ref/imagelabeler-app.html (accessed on 1 March 2021).

38. Sharma, S.; Goodin, C.; Doude, M.; Hudson, C.; Carruth, D.; Tang, B.; Ball, J.E. *Understanding How Rain Affects Semantic Segmentation Algorithm Performance*; SAE Technical Paper; SAE: Detroit, MI, USA, 2020; 2020-01-0092.
39. He, X.; Zhao, K.; Chu, X. AutoML: A Survey of the State-of-the-Art. *Knowl.-Based Syst.* **2020**, *212*, 106622. [CrossRef]