Procedurally Generated Simulated Datasets for Aerial Explosive Hazard Detection

Jeffrey Kerley[a], Aaron Fuller[a], Madeline Kovaleski[a], Peter Popescu[a], Brendan Alvey[a], Derek T. Anderson[a], Andrew Buck[a], James M. Keller[a], Grant Scott[a], Clare Yang[b], Ken E. Yasuda[b], and Hollie A. Ryan[b]

[a]Department of Electrical Engineering and Computer Science, University of Missouri, Columbia MO, USA
[b]U.S. Army DEVCOM C5ISR Center, Fort Belvoir, VA, USA

ABSTRACT

Recent advancements in signal processing and computer vision are largely due to machine learning (ML). While exciting, the reality is that most modern ML approaches are based on supervised learning and require large and diverse collections of well annotated data. Furthermore, top performing ML models are black (opaque) versus glass (transparent) boxes. It is not clear what they are doing and when/where they work. Herein, we use modern video game engine technology to better understand and help create improved ML solutions by confronting the real world annotated data bottleneck problem. Specifically, we discuss a procedural environment and dataset collection process in the Unreal Engine (UE) for explosive hazard detection (EHD). This process is driven by the underlying variables impacting EHD: object, environment, and platform/sensor (low altitude drone herein). Furthermore, we outline a process for generating data at different levels of visual abstraction to train ML algorithms, encourage improved features, and evaluate ML model generalizability. Encouraging preliminary results and insights are provided relative to simulated aerial EHD experiments.

Keywords: artificial intelligence, machine learning, U-Net, simulation, unreal engine

1. INTRODUCTION

Data and computing are two of the primary rock stars driving leaps in supervised learning-based *machine learning* (ML). While data is ubiquitous, accurately labeled large real world data sets for ML is not. As a result, numerous companies have emerged and raised tens of billions to label data for ML.[1] However, as numerous theoretical and experimental studies suggests, the answer is not as simple as "go collect more data and do better." What data should we collect? How much data? Under what contexts (environment, camera, object, etc.)? etc.[*] Another complication is our current theoretical and practical lack of ML understanding. While many ML algorithms, e.g., deep learning algorithms like *convolutional neural networks* (CNNs), are powerful *universal function approximators* (UFA), their data-driven nature has resulted in black box vs. transparent solutions. This challenge has manifested itself into a new field of so-called *explainable artificial intelligence* (XAI). In this pursuit of trustworthy data-driven ML/AI, we must also confront the real world practical *expense* of data collection; time, cost, storage, etc. All of these compounding factors have led to our current predicament that suggests that the real world might not be the ideal destination to research and develop ML/AI.

Herein, we propose a new exploratory research platform powered by simulation to investigate questions tied to trustworthy and explainable data-driven ML/AI. Namely, we use the *Unreal Engine* (UE) to procedurally generate a controlled and perfectly *ground truthed* (GT) set of simulated data across the "*reality spectrum*" (RS). On one extreme of the RS (see Figure 1) is abstract and stylized imagery, e.g., artistic rendering. On the other extreme is real world data. Examples of subjective categories in-between include *cartoon* (TOON), modern video *game engine quality* (GEQ), and photo-realistic (e.g., UE5) data[†]. The current article is focused

[*]We would like to note that while topics like unsupervised learning and self-supervised learning might help reduce the burden of exploiting data, they are clearly also subject to what data (available information) they are provided.

[†]See Section 3.3 for a more detailed explanation of each category

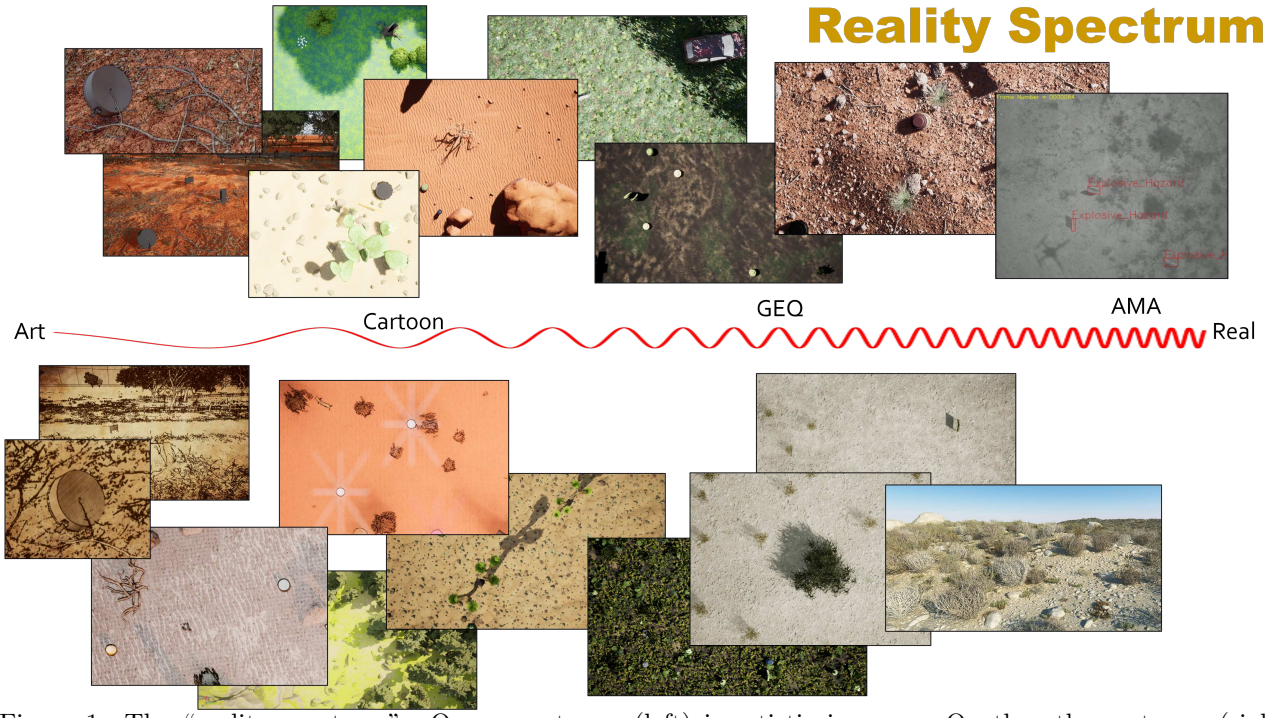


Figure 1: The "reality spectrum". On one extreme (left) is artistic imagery. On the other extreme (right) is real data. Example sample locations in the RS include cartoon and modern game engine quality data (not photo-realistic). Imagery shown are examples we produced using Epics Unreal Engine.

Table 1: Notations and Acronyms

Acronym	Description
AMA	Altitude Modulated Augmentation
GEQ	Game Engine Quality Simulated Data
RS	Reality Spectrum
LV	Linguistic Variable
EH	Explosive Hazard
EHD	EH Detection
ML	Machine Learning
AI	Artificial Intelligence

on establishing a procedural simulation framework to generate large collections of training data and support experiments in/across the RS. While the prior is of use for an application like EHD, the latter enables deeper questions surrounding XAI, e.g., what was learned, how generalizable is an ML/AI model, etc.

This paper puts forth the following specific contributions. First, a procedural algorithm is implemented in the UE for pseudo-random RGB and 3D dataset generation within a users specified attribute range with respect to object, environment, and platform/sensor variables for EHD. Second, we discuss the generation of TOON, modern video GEQ, and stylized or artistic imagery in the UE. Third, these tools are used to perform a quantitative study about the impact of a single environment variable, time of day, on an EHD pre-screener. Fourth, qualitative and quantitative open ended experiments are provided to understand how well a model generalizes and abstracts.

The remainder of the article is structured as follows. In Section 2 we discuss related work. Section 3 discusses our simulation process, EHD variables, procedural scene and data collection, and definitions are provided for categories in the RS. Section 4 details our experiments and results. Figure 2 is a high-level overview of our proposed article and Table 1 is acronyms.

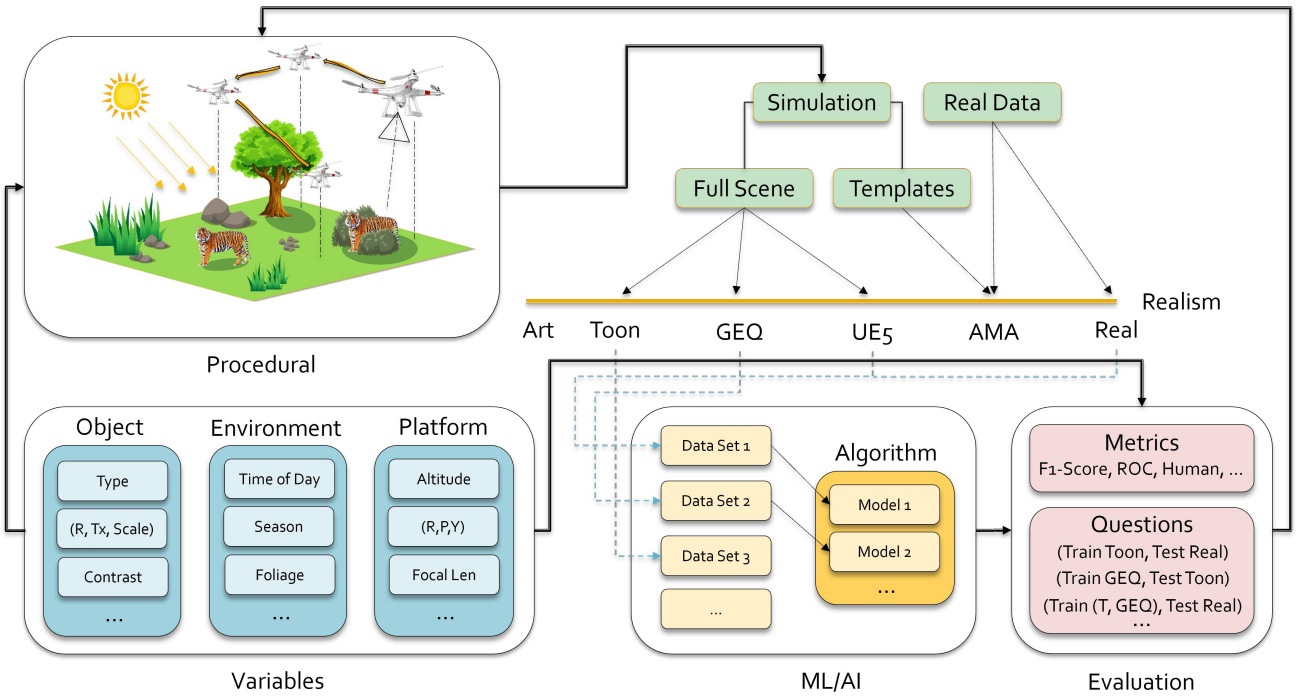


Figure 2: Illustration of this paper: (i) underlying domain variables that drive the task at hand; (ii) generation of data at different points in the "reality spectrum"; (iii) training ML/AI models relative to subsets of the data from the RS; (iv) evaluation; and (v) feedback to procedural scene generation and data collection.

2. RELATED WORK

2.1 Explosive Hazard Detection

EHD is a real world challenge that is not going away. EHs are not trivial to detect as they vary with respect to factors like size, shape, composition, material, and context in/across environments and environmental conditions. The task of detecting EHs is one better suited for a UAV than a precious human being. UAVs are replaceable and they offer a way to keep humans at a safe standoff distance. They can be equipped with high resolution cameras observing different portions of the electromagnetic spectrum as well as position sensors (e.g., GPS and IMU). However, detecting EHs from an aerial platform is not a solved nor trivial problem. For example, imagery collected by a UAV at an altitude of 30 meters looking straight down (nadir) looks vastly different from a UAV at a lower altitude looking at a different pitch. The point is, UAV-based EHD has great potential, but it is a complex technological task that is full of many sub-challenges.

While our current article is focused on UAVs, a number of technologies have been explored to date. An early and well-known technique is the so-called "metal detector", which can be used to detect metal buried in the ground. However, one limitation with this form of detection is that threats that contain low amounts of metal may go undetected. Increasing the sensitivity of the device does not necessarily counteract this, as the number of false alarms would likely dramatically increase. To increase the robustness of detection, many different combinations of sensing methods have and are being explored, such as *infrared* (IR), *ground penetrating radar* (GPR), *electromagnetic induction* (EMI), and *hyperspectral imaging* (HSI), to name a few. The two predominant approaches to date for detecting explosives is vehicle-mounted detectors and hand-held detectors. While the latter is predominantly used in a downward looking fashion, the prior comes in a multitude of forms, e.g., forward looking,[2] downward looking,[3] and even side looking.[4] The reader can refer to[5] for a recent review of computational intelligence algorithms in EHD. Herein, we focus on UAVs, which can operate in each of the above modalities. This method of data collection has the potential to help search areas faster, especially in the case of a swarm of UAVs, and with behaviors to facilitate dynamically interrogation of regions of interest.

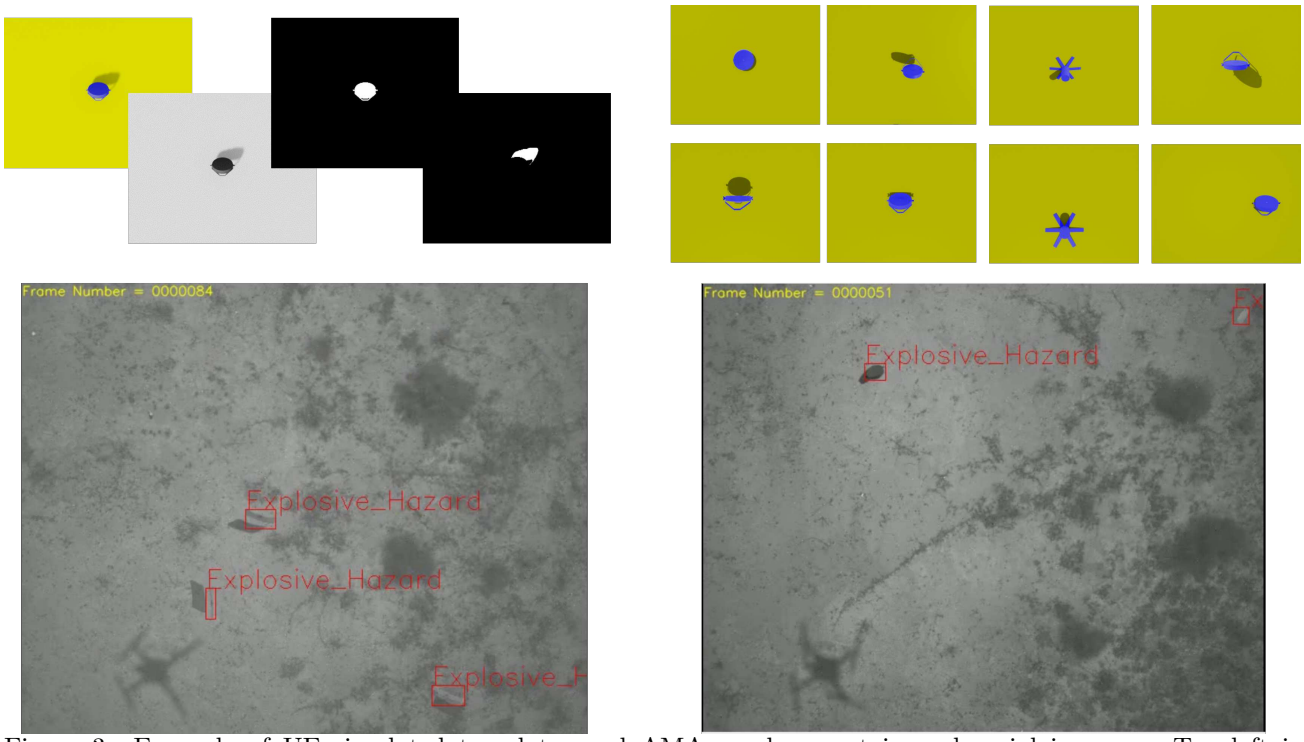


Figure 3: Example of UE simulated templates and AMA emplacement in real aerial imagery. Top left is a simulated false color RGB image (R=grayscale image, G=object pixels, B=shadow pixels) and individual channels. Top right is simulated objects at different camera poses and sun positions. Bottom left and right is AMA placing the above simulated templates into real aerial data (EHs highlighted as red AABBs).

On a final note, it is important to note that the signatures being exploited here derive from line of sign spatial features, the targets under consideration are surface-deployed, and the solution space being worked is *electro-optical/infrared* (EOIR) against line-of-sight EH only.

2.2 Simulation

We are not the first to use game engines to train and evaluate ML/AI techniques. In 2015, Gaidon et al. used Unity to make a *VIRTUAL KITTI* (VKITTI) autonomous car non-photorealistic dataset.[6] In 2015, Chen et al.[7] used *The Open Racing Car Simulator* (TORCS)[8] to train a *deep NN* (DNN) to drive (again, non-photorealistic imagery). In 2017, Martinez et al. used the video game Grand Theft Auto to generate high quality rendered imagery for training, testing, and enhancing DL for self-driving cars.[9] In 2018, Martinez et al. proposed UnrealROX,[10] which used UE4 to create realistic looking indoor scenes for robots to interact with objects in simulation. In 2018, Muller et al. explored Sim4CV[11] in UE4 for generic CV research. In 2020, Drouin et al.[12] used real ortho-photos to simulate aerial data collections, and in 2021, Nouduri et al.[13] generated synthetic views from a dense 3D point cloud. In Alvey, et al.,[14] we proposed an UE and AirSim based framework and workflow, with open source code[‡] and training videos[§], for accelerating computer vision and unmanned low altitude aerial vehicle research. Examples were highlighted for object detection, passive ranging, context-driven fusion,[15] and augmented reality. While Sim4CV is the closest to our current article and previous UE-based work,[14] the content used and imagery produced is not photorealistic, no detailed workflow is outlined, no supplemental online training is available, and results are simulator-focused vs. real world and simulation cross-validated.

In,[16] see Figure 3, we previously put forth a process called *altitude modulated augmentation* (AMA). AMA uses the UE to render images of objects in different contexts. For example, we render target EHs in different object

[‡] https://github.com/MizzouINDFUL/UEUAVSim
[§] https://bit.ly/MizzouINDFUL

poses, camera relative poses, and solar positions (which impact illumination and shadows). As Figure 3 shows, the result in a false color RGB image where the red channel is the grayscale (monochromatic) image, green is a per-pixel map of object locations, and the blue channel is a per-pixel map of shadow locations and values[¶]. These images, which we refer to as "templates", are then inserted into real low altitude aerial data. The object and shadow layers are used as alpha transparency maps. AMA uses the platform/sensor metadata (drone altitude, pose, etc.) to decide where and how to insert templates into the real data (see[16] for more details). The goal of AMA is to use simulation to render novel contexts of objects that are not observed in an existing data collection. AMA can be regarded as a type of data augmentation. As shown in,[16] the performance of AMA is notably higher than what is current achieved using full image simulated data. This should be expected as AMA aims to use existing real world data for a specific environment. It is hypothesized, but not yet proven, that AMA is perhaps ideal for scenarios like training or transferring an AI/ML model to a new operating environment using only a small collection of non-target background data. This is an attempt to reduce the expense (cost, time, etc.) of acquiring training data required for a new environment. While useful, AMA has limitations. For example, AMA requires background (non-target) data from the new environment. Furthermore, AMA only enriches existing data with respect to the object. It does not help with variables like time of day, environmental differences, emplacement context (e.g., occlusion), etc. This is a goal of the current article. We wish to use simulation to increase our training data set with respect to a wider range of environments, environmental conditions, and platform (drone and camera) contexts.

The next topic is procedural generation. Procedural is a massive topic that has found utility in a number of areas spanning decades. Examples include terrain generation, materials, animation, game play, AI, etc. Video games known for their use of procedural techniques include, but not limited to,"Borderlands," "Dwarf Fortress," "Left 4 Dead," and "Spore." The reader can refer to the following thesis for procedural character animation,[17] procedural terrain generation,[18] Perlin's historical work on noise functions and procedural generation of textures and materials (e.g., marble, wood, etc.),[19] and Valve's "Left 4 Dead" procedural AI system and procedural gameplay.[20] These are just a few of the hundreds or possibly thousands of works related to procedural methods for entertainment (e.g., games and movies). Procedural methods have made their way out of the hobby or purely academic circles. Companies like Houdini[21] exist to support the generation of landscapes, cities, swarms of people, and beyond for well-known profitable movies like "Lord of the Rings," Disney's various billion dollar Pixel and Marvel movies. Most recently, Lucasfilm and Disney used the UE and Houdini to produce the well-known and renowned "The Mandalorian" TV series. While procedural has found its way into content generation for movies and games, What is the role of procedural with respect to training and evaluating ML/AI?

The next topic is concept learning in the context of modern deep learning. In Geirhos et al.,[22] it was demonstrated modern deep learners, specifically convolutional neural networks, are biased toward learning information such as texture versus shape. However, this contradicts human vision and higher-level cognition. The authors propose a data augmentation methodology based on image stylization. Specifically, they used Stylized-ImageNet, a neural network that takes existing images in and produces stylized imagery out. The resultant image often looks like oil paintings or other abstract art. The authors trained on this stylized imagery and showed that the resultant deep models were significantly more robust to recognizing shape. Whereas the models previously failed with an object was re-textured, e.g., a cat with elephant skin, or when silhouettes of objects were provided, the new models were able to recognize the objects correctly. This research suggests that if a machine is provided with many variations with different texture and color and contrast, but shape is similar and consistent, the machine is faced with the dilemma to learn features and memorize each image or find better features that generalize. We regard this work as demonstration of the weaknesses of a deep learning algorithm to pay attention to false correlations in a data set, namely those with high frequency and abundance. Shape is relatively low frequency and sparser. Herein,[22] is interesting because shape is something that we propose is important, perhaps vital, to learn in EHD, and it demonstrates that stylization can help learning by providing a 'decision' on behalf of the algorithm, abstract and learn or memorize. We claim that this helps motivate our current article as one of our objectives is to use simulation and shaders to produce samples at different points on the RS. Our artistic and

[¶]If additional per-pixel information needs to be stored, e.g., RGB color, then different UE targets, e.g., custom depth, custom stencil, render targets, etc., can be used. The reader can learn more by reading about custom UE materials, post processing effects and materials, and the Movie Render Queue.

cartoon imagery possess less texture and more shape, color, and contrast, with subtle variations.

Last, in,[23] Hinterstoisser et al. demonstrated experiments and results that are relevant to the current article. Namely, they demonstrated that simulation, or simulated objects inserted into real imagery, can sometimes introduce artifacts. Examples of artifacts in templates inserted into real data include edge insertion details, color, contrast, and/or texture differences, and simulated object artifacts. Examples of artifacts in simulated data include lighting, 3D model and/or texture details, and aspects related to the simulation environment, e.g., popping at distance with respect to level of detail changes. The point is, Hinterstoisser et al. demonstrated that it can be an advantage to learn features in the real domain, lock the lower level visual features (edges, colors, texture), and just update or transfer learn the higher level weights that people tend to associate with behavior like component and semantic reasoning. The model does not need to learn artifacts in the simulated data. Instead the networks can learn the deeper associations and relationships, which we expect will transfer back over to the real world. The point is, training methods exist to build and update deep models using full simulated and/or partially simulated imagery.

3. SIMULATION

As discussed above, this article is focused on the generation of entirely simulated data. The goal is to have control over the ability to alter object, environment, and platform/sensor variables that drive ML/AI and the task at hand (e.g., EHD). While different simulators could have been used, e.g., Unity,[24] VANE and ANVEL,[25] etc., we focus on UE herein because of its longevity (decades of R&D), wide integration of assets (3D models, textures, etc.), simplicity of use, online documentation, and high quality global illumination and realtime rendering capabilities (see NVIDIA *deep learning super sampling* (DLSS)). Historically, UE was created for video games, but it is now used in film,[26] computer graphics,[27] architecture,[28] and beyond. The bottom line is, high fidelity custom dynamic scenes can be manually produced in little time or purchased. It is also easy to mix content, manually or via scripting, to vary or cater to niche applications where scenarios are costly, hard, or not practical to obtain. UE is also constantly improving and making free content avilable, e.g., Quixel megascans 3D scanned real world objects and materials[29] (see Figure 5) and military free 3D objects,[30] as well as supporting tools for procedural content generation, e.g., large scale terrain generation with Instant Terra (UE plugin[31]) and Houdini for object/geometry, texture, terrain, animation, city generation and beyond (see UE Houdini plugin[21]). Furthermore, UE has a rich support for models, materials, effects, and more on their UE Marketplace[32] (see Figure 4), including the artistic shaders we used herein (cel shader[33] and artistic pen and paper shader[34]).

3.1 Underlying Problem Domain

Our EHD problem is fundamentally driven by underlying domain variables. The three categories of variables explored herein include: object, the thing we wish to detect; environment, the world where our object resides; and platform, the camera (RGB and/or IR) and device (drone) doing the remote sensing. Example object variables include: type, size, pose (roll, pitch, yaw), texture, contrast, color, occlusion, etc. Example environment variables include: season, time of day, region, foliage type, foliage density, foliage height, nature clutter, man made clutter, etc. Example platform/sensor variables include: elevation, pose (roll, pitch, yaw), standoff or number of pixels on target, speed, image resolution, focal length, aperture, etc. The point is, challenges like EHD consist of many variables and the sheer combinatorial size of this variable space is overwhelming and needs to be addressed. This is relevant to the current article because these are the factors that drive our procedural scene generation, automated data collection, and ultimately it is the deep domain questions we are trying to understand. Table 2 summarizes our procedural algorithm variables and their constraints.

Our procedural simulation is not based on brute forcing (its not possible) nor grid searching the EHD variable space. We avoided the latter because we do not want to inadvertently insert a bias with respect to the grid. Instead, we define operating intervals for each variable. Our procedural scene and data generation process (see Section 3.2) pseudo-randomly samples the constrained variable space. In this respect, we can run the procedural tools multiple times in the same ranges/context and build similar, but not exactly similar datasets. If there is deviation in performance across these results then that is something we want to know and study.

Figure 4: Example urban city for purchase on the Epic Marketplace.[35]

Figure 5: Example free PBR-based interactive high geometry and quality vegetation from Quixel.[36]

3.2 Procedural Generation

As detailed in Algorithm 1 and illustrated in Figures 8, our process for generating pseudo-random simulated EH data in UE is relatively simple. The first step is choosing relevant game assets for the scene composition. The scenes being captured were based upon desert climates. After a climate is chosen, the style, TOON or GEQ, is picked to decide what subset of assets to use. Higher quality, complex geometry and textures are used for GEQ. Simple, low polygon count geometry and plain solid textures for TOON. Once the scene assets are chosen, defining how the data will be captured is next. Starting with the path of the camera, a grid walk approach

Table 2: EH Variables Adjusted in the Current Article

Variable	Description
Object Position, Pose, and Size	We randomly select object Yaw in $[0, 360]$ degrees, position was randomly determined (see Section 3.2), and size is randomly adjusted by $[-10, +30]\%$ of its size (which was approximately the size of the object in the real world).
Object Texture, Color, Contrast	We picked a set of textures with dark color and no texture, bright color with no texture, no texture and color similar to environment, and texture and color low, medium, and high similarity to background textures.
Object Occlusion	Our experiments included occluded and unoccluded objects.
Time of Day	See the experiments, but we generated data in the morning, at solar noon, and in the late afternoon; which resulted in lux and shadow differences.
Region	We focused on arid destinations for this article.
Terrain	We picked a few PBR (Physically Based Rendering) textures from Quixel and the Unreal Marketplace to match our desired arid scenes.
Foliage	We picked a small set of objects for our arid region: rocks, bushes, grass, flowers, etc.
Clutter	We picked a small set of objects common in arid regions: cacti, plant debris, etc. In addition, we made sure to include objects that appear similar to our targets, in shape, size, texture, and color.
Drone Elevation	We collected data at 30m
Drone Pose	We randomly adjusted camera pitch by $[-5, +5]\%$, Yaw was random, and Roll was kept constant. These parameters were in support of a camera on a drone that is "mostly looking nadir, plus or minus natural platform motion".

was used to construct the the camera's path as a set of waypoints[‖]. The grid was randomly perturbed to allow for more diversity in camera position. Instead of uniformly selecting points along a regular grid, random points are selected along a circular arc from each regularly spaced point. The amount of variance can be adjusted by changing the radius of the circular camera spawn zone around each point. See Figure 6 for a visualization of a generated flight path. Adding on top of this idea, the camera's yaw is randomized so that we are not always looking in the same direction. As a result, even if the camera goes to a similar camera position more than once, it will likely result in a significantly different image. This improves data diversity for shadows, and object orientation.

This approach for controlling the camera allows for repeated looks in varied positions, a randomly generated path, and a data collection that is guaranteed to see all of the targets. By randomizing the collection parameters as described above, a more robust dataset is created. A robust training dataset is important to prevent ML algorithms from memorizing information, such as specific background details that might be baked into the landscape, or other distracting information. Along with camera position and orientation, foliage and targets are also produced randomly. At each waypoint, a number of targets and foliage instances are placed into the map. They are given a random yaw and scale, and are spawned within a bounded square from the center of each zone. Proving that each target type is spawned at least a certain number of times, both foliage and targets are placed using a uniform probability distribution, with some variance. Looking at scene generation specifics, the number of targets and foliage will be different each generation, but will be within the distribution bounds. Again, the purpose of this is to try to force the algorithm to learn more generalized features, rather than memorizing dataset specifics such as how many targets should be detected in each frame.

Each target that is emplaced has one of two material styles applied. The two styles explored herein are TOON and GEQ. TOON styled objects have flat, mostly non-textured materials. GEQ styled objects have high resolution, highly textured, complicated materials applied. An example which demonstrates the separation

[‖]We selected a uniform grid walk strategy because game engines primarily operate local to the "player". As such, random sampling across a map results in unstable results as the engine needs to stream in assets and many effects take multiple frames to stabilize. Random perturbations around a uniform grid gives the engine adequate time to stream and stabilize locally to achieve best results.

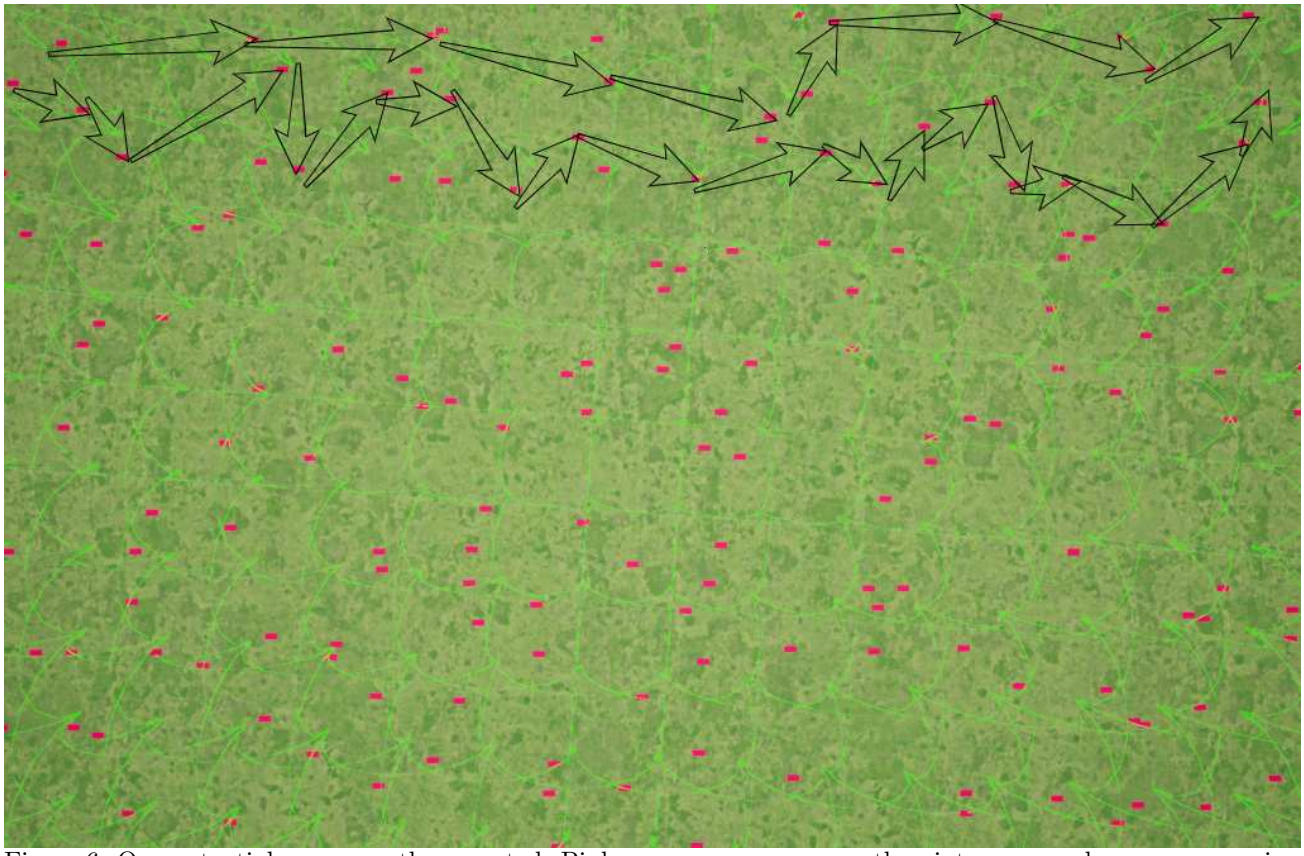


Figure 6: One potential camera path generated. Pink squares are camera path points, green spheres are spawning zones, and arrows are a camera path. As can be seen, this accounts for drone motion, in a randomly controlled manner versus perfect linear flight patterns between waypoints.

between TOON and GEQ targets is shown in Figure 7. The material is swapped on an object each frame, which allows the camera to capture each target with many different materials. Swaps are randomly chosen from a list of 10 possible material choices for each style. The purpose of randomizing the applied material is to create a dataset which when trained on, produces a model that is insensitive to texture / material. We do not want to train an explosive hazard detector which cannot recognize a particular mine if it painted a different color or the surface is scratched in the field. Furthermore, the difference in material properties between TOON and GEQ allow us to evaluate the level of detail and its effect on target detection. That is, how does material texture/quality effect target detection?

Scene generation is done using the UE4 landscape foliage spawner, and manual landscape creation. The foliage spawner takes in any number of assets, grass, trees, rocks, etc., and places them based on density and jitter amount to vary the generation pattern. Placement location is calculated using the current material of the landscape. In the future, a generation pipeline that creates many landscape textures (done manually for now) would allow for greater foliage diversity. Specific grass, tree, rock, and all other clutter location should not be important for detection. As a result, placement location should be different for each simulated collection, which the above approach achieves. If a scene were to be analyzed, and a comparison between the same location on different collections were made, we would find that the background clutter (or targets for that matter) does not fit any specific pattern. Again, a random approach for foliage/clutter requires the algorithm to learn target detection that is not dependent on background and is more generalized.

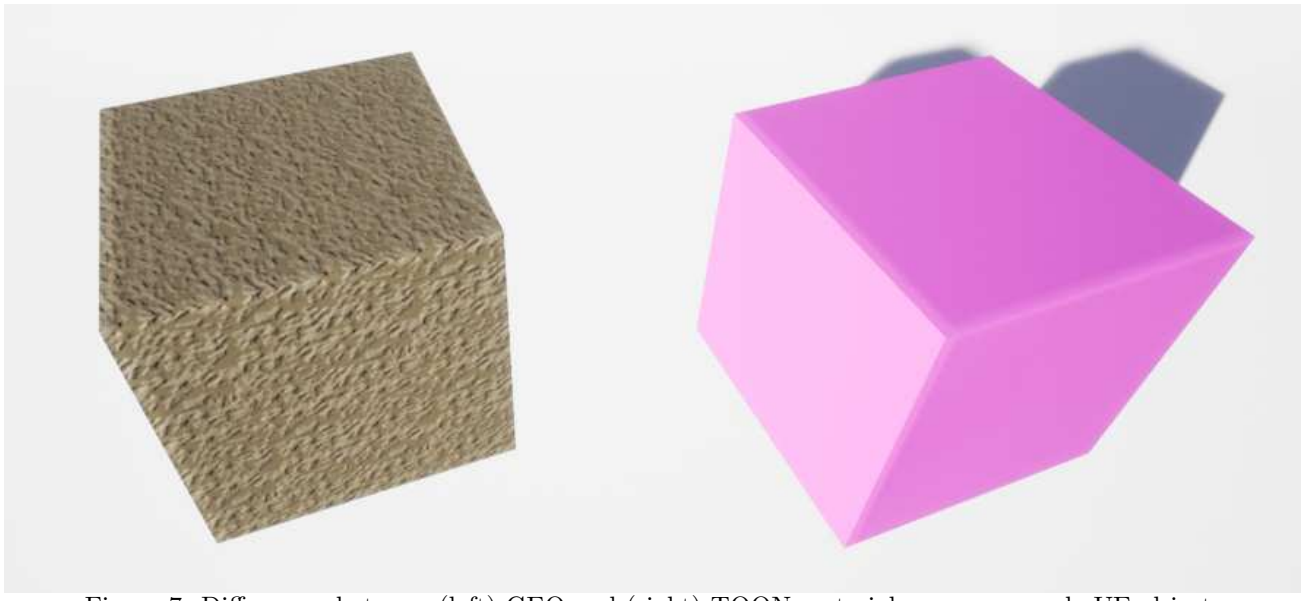


Figure 7: Differences between (left) GEQ and (right) TOON materials on an example UE object.

Algorithm 1: Procedural Scene Generation and Data Collection Workflow in UE

1 Choose scene assets: grass, bushes, targets, rocks, cars, etc.
2 Add desired lighting, materials, and textures to match rendering (TOON/GEQ) style.
3 Generate a set of regularly spaced camera spawn points in a grid
4 Perturb each base camera point by selecting a random position along a circle with a user defined radius from the base point.
5 Randomly spawn chosen targets and background assets at camera points, within a bounded distance from each point with respect to user defined constraints on variables: object (pose, size, contrast, texture, etc.) and environment (foliage height, density, target occlusion percentage, etc.)
6 Collect data (RGB images, depth, object and instance IDs, 2D AABBs, 3D AABBs, normal maps, etc.) as the camera moves through the generated waypoints, randomizing altitude 30m herein), and camera pitch (±5 (degrees) off-nadir herein). At each game engine update, update any moment-to-moment elements, e.g., rotate object textures.

3.3 Cartoon (TOON) Data

The reader can refer to Figure 9 for examples of what we are calling cartoon; referred to as TOON hereafter. Our informal definition of TOON is imagery with a limited range of color, simplified object shape with lower and simpler geometry, stylized or low-to-no texture, and quantized illumination and shadows. These images are also usually high in color saturation. Fundamentally, the reason for including TOON is its our belief that this imagery possess quality features such as shape, color, and contrast, which are sparser and harder for an AI/ML algorithm to possibly learn in more realistic looking data. At the end of the day, this is the simplest to produce content. If ML/AI can be trained, or enhanced, using this information, that will be a win as a great abundance can be produced. On a final note, this category is also a wonder reality check, as if an ML/AI model trained on more realistic data cannot recognize it then we are forced to wonder if the machine is simply memorizing, rather than generalizing.

3.4 Game Engine Quality Data

The reader can refer to Figure 10 for examples of what we consider as modern video GEQ data. Our informal working definition of GEQ is imagery with a wide range of realistic colors (not oversaturated like in TOON), more realistic and complex object shape and geometry, more natural looking texture, and more realistic looking illumination and shadows. Essentially, GEQ is limitations with present generation content and real time gaming.

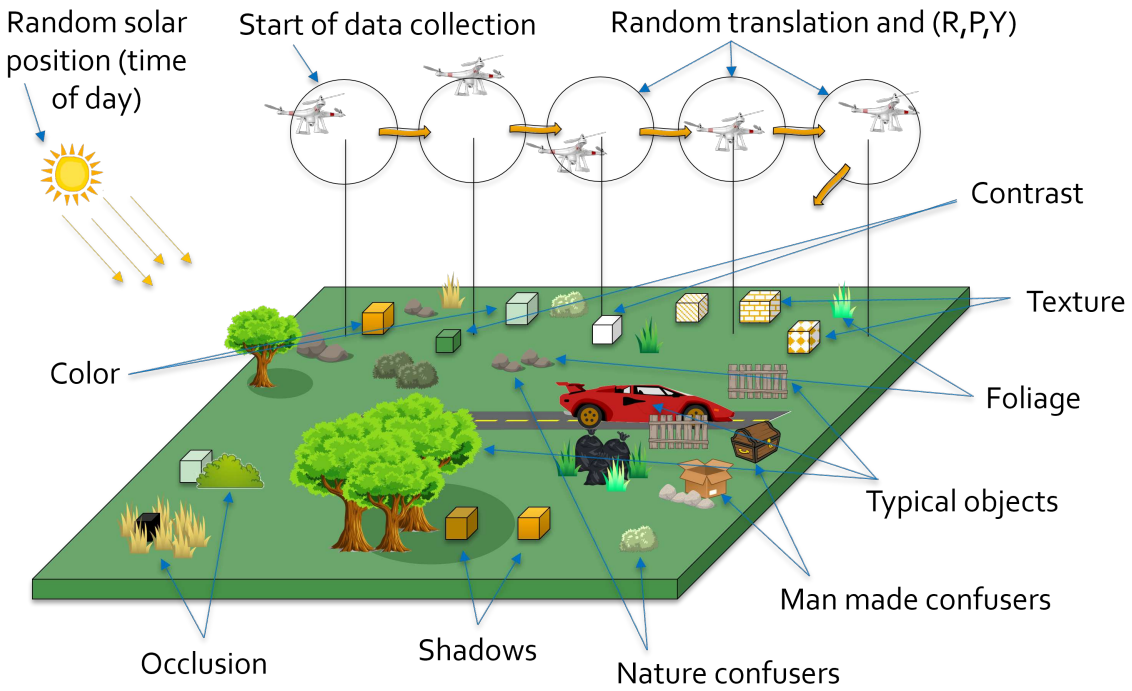


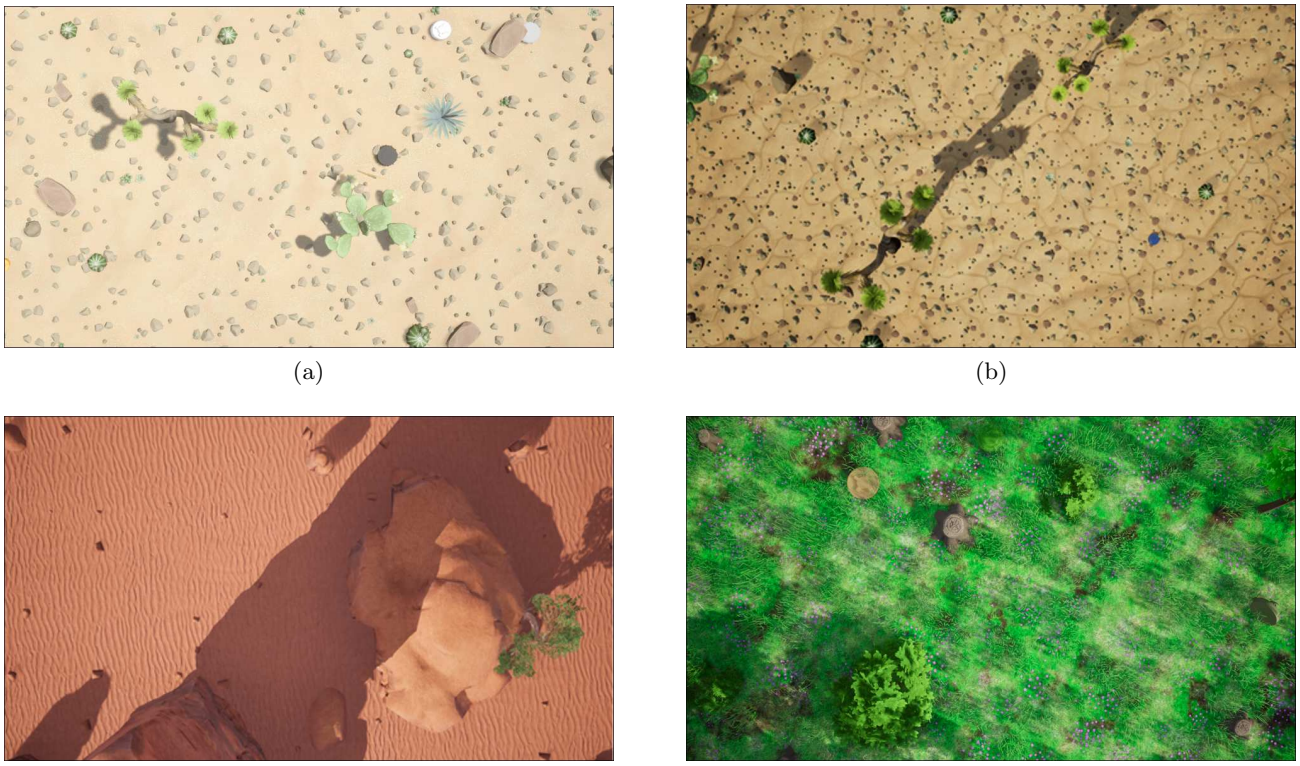
Figure 8: Illustration of our procedural map/scene generation and data collection in the UE. See Section 3.2 for full details. The main idea is to create a dataset that has a dense set of targets, close confusers (nature and man made), and common environment objects (nature and man made). As the primary goal of this initial paper is training a pre-screener for EHD, no effort is put forth to model local or global contextual relationships (spatial, spectral, etc.). The idea is to maximize the number of looks on objects in different contexts with respect to underlying low level visual features; color, contrast, occlusion, etc.

Fundamentally, the reason for including GEQ is its our belief that this "looks close" to real imagery and it substantially simpler to produce – textures, 3D models, real-time rendering, etc. – than photo-realistic imagery. From a ML/AI model cross-validation standpoint, this is an interesting category of data to test on because we might hope, or perhaps expect, a real world trained ML/AI model to work on this, otherwise overfitting is likely a reality and one should not expect the model to generalize to new settings.

3.5 Artistic and Stylistic Quality Data

The reader can refer to Figure 11 for examples of what we are calling stylized or artistic imagery. This is an extremely challenging category: *what is art*? Examples herein include sketches, ink or pen, Cel Shading, etc. These are imagery that we can generate using well defined post processing shaders in UE. In future work we might consider extending this category to methods like styleGANs (generative adversarial networks) for specific styles like Picasso or Vincent van Gogh artwork. Herein, our category of artistic shaders can be broken down into fundamental elements properties like alteration type and degree of color, texture, contrast, and etc.

Fundamentally, the reason why we have included stylized or artistic data is to explore with just how far we can push the envelope of learning and evaluating a ML/AI model. As mentioned above, existing work like Geirhos[22] exist and suggests that important features like shape are not currently being learned and data augmentation tricks that stylize imagery might encourage a machine to learn what might otherwise not arise. Herein, we take our GEQ and TOON data and apply post processing shaders from the Unreal Marketplace. These shader packs included cel shading,[33] and a toolkit with many newspaper, comic, and grey scale effects.[37] Each of these post processing tool kits allowed the target scenes to achieve differing levels of art abstractions.



(a) (b) (c) (d)

Figure 9: Example procedurally generated TOON imagery for arid and forested environments.

4. EXPERIMENTS AND RESULTS

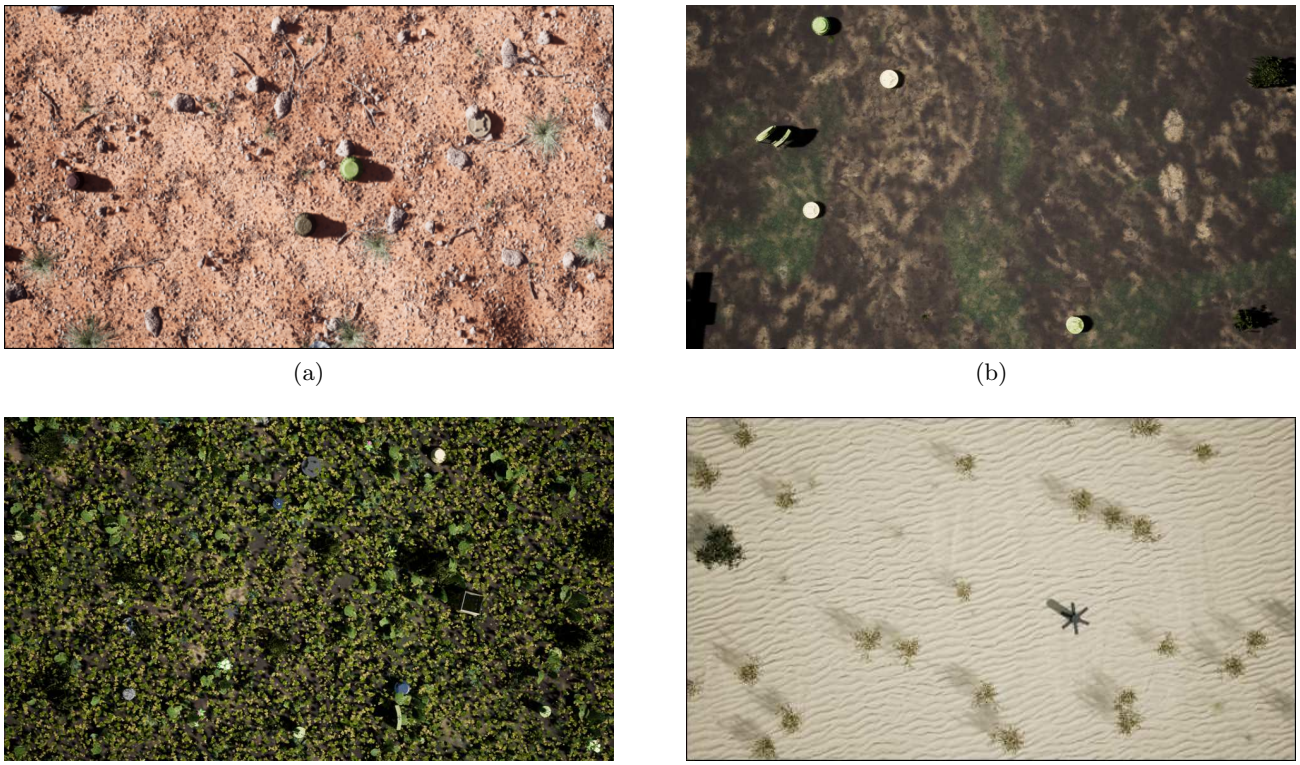
Herein, we explore the U-Net[38] neural network architecture as an EHD prescreener. While U-Nets have been extensively used for image semantic segmentation, they have also become the backbone for problems like passive ranging, deep diffractive neural networks and holography, and beyond. We selected a U-Net for the following reasons. First, they can be used for segmentation at a per-pixel level, something not possible via detection and localization networks like YOLO[39,**]. Second, a U-Net can be engineered to model behaviors similar to attention in humans.[41] This is in effect what we are trying to achieve: an algorithm that queues AOIs requiring further analysis; which is also supported indirectly in networks like YOLO through cost function encouragement ("objectness"). Third, a U-Net can achieve a fair amount with its limited parameters due to weight sharing and skip connections. A U-Net can be reduced in number of parameters, trained and used computationally on a limited resource embedded device like an NVIDIA Jetson.

While many variants exist, in general the U-Net architecture consists of a downsampling path to extract features (the "what"), and an upsampling path to identify target class locations in an image (the "where"). These paths consist of blocks connected by skip connections, allowing for more detailed image construction across scale from encoder to decoder. The paths are symmetric, meaning that for each encoder convolution that decreases in resolution, there is a corresponding decoder convolution that increases the resolution, giving the U-Net its name in the way the visualization of the architecture mirrors the letter "U."

The U-Net used herein was pulled from an existing codebase for semantic segmentation[††]. Our network was trained on grayscale image datasets where the weights were updated based on a Dice Loss function and a Cosine

[**]The reader can refer to[15,16,40] for our prior works using simulation for training and characterizing localization and detection AABB detectors (specifically YOLO) vs. segmentation.

[††]Semantic segmentation network library used herein can be found at https://github.com/qubvel/segmentation_models.pytorch/blob/master/docs/index.rst and https://smp.readthedocs.io/en/latest/



(a) (b) (c) (d)

Figure 10: Example procedurally generated GEQ imagery for arid and forested environments.

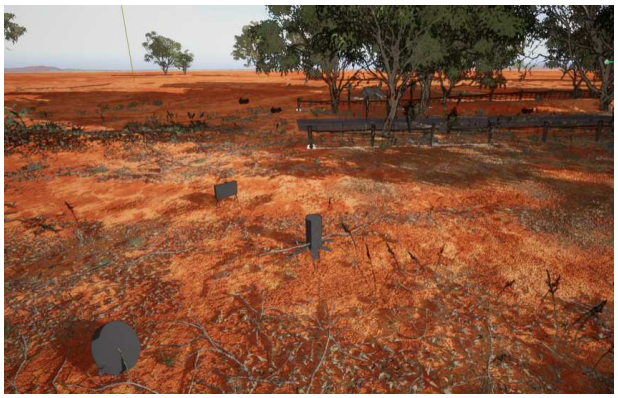
Annealing learning rate scheduler. A per-pixel ground truth was provided for each training image, where zero indicated not a target and one indicated target. The upsampling and downsampling paths of the U-Net both consisted of five stages, respectively.

Seven datasets (see Table 3) were generated in support of our experiments (see Table 4). Each dataset consists of 1,000 images. Experiment 1 (E1) is focused on evaluating performance across different times of day. To this end, datasets 1 through 3 (DS1, DS2, and DS3) represent "train at solar noon then operate earlier or later in the day" and "train and test across different times of day." DS1-DS3 are focused on GEQ data while DS4-DS6 are the same experiments for TOON quality imagery. In DS1 and DS4, the sun is at its highest point for the day, lux is maximized, and shadows are minimized. Statistically speaking, our data has greater intensity and highest contrast. On the other hand, in DS2 and DS5 lux is lower and shadows are more complex. As the reader can see (Figure 12), early and late day have lower intensities and lower contrast.

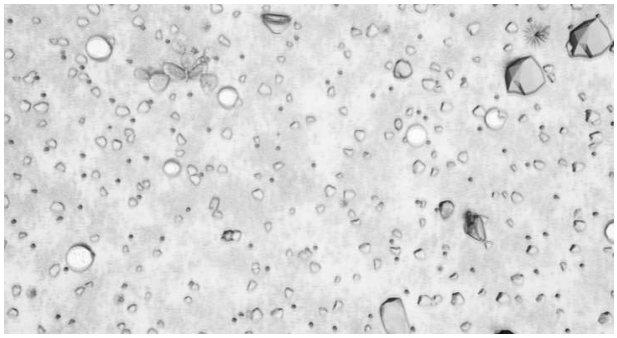
In E1, two models were trained, one for only solar noon and one for all time of day. Independent procedurally generated test data sets were created to avoid re-substitution. In Experiment 2 (E2), the three TOON datasets were used to train a model that is tested on GEQ quality data. In Experiment 3 (E3), a U-Net was trained on GEQ imagery and tested on TOON imagery. The intent of E2 and E3 is to observe if simplified and relatively easy to generate TOON data results in a model that generalizes to more detailed GEQ data. This experiment is ultimately in support of determining if simpler imagery can be used to train models for more complex data. Conversely, E3 is intended to help us understand if models trained on more complex imagery are able to abstract. Should the reader be confident in a model that only works on real world data? Does this suggest memorization versus features and reasoning more like humans? Last, Experiment 4 (E4) is an extreme version of E3. The goal is to abstract our data into a more artistic realm such that we can see how far the trained models work.

In summary, E1 demonstrates the utility of the proposed tools for exploring specific questions that impact EHD, e.g., time of day. Experiments E2-E4 are more open ended. Questions include, "is it possible to use

(a)



(b)



(c)

(d)

Figure 11: Example of simulated stylized or artistic imagery: (a) ink and paper shader, (b) combination of GEQ and cel shader, (c) sketch filter, and (d) quantized color cel shader.

Table 3: Description of datasets.

Data Set	Parameters	Description
DS1	(**GEQ**, **Solar Noon**, **Arid**)	Solar noon, minimal shadows, peak lux
DS2	(**Early & Late Day**, GEQ, Arid)	Lower lux, harsh shadows, [7,9]am and [5,7]pm
DS3	(GEQ, Arid)	Collect across [7am,7pm]
DS4	(**TOON**, Solar Noon, Arid)	Like DS1, but TOON vs. GEQ
DS5	(Early & Late Day, **TOON**, Arid)	Like DS2, but TOON vs. GEQ
DS6	(**TOON**, Arid)	Like DS3, but TOON vs. GEQ
DS7	(**Artistic**)	Post processing to make more artistic imagery

simpler data to train models that work on more complex data" and "does a model trained on more complex data abstract to simpler data." The first question is important because it can help us unlock what features are important and our simulation pipeline can focus on generating such data to improve performance and make more trustworthy models. The second question is important because we suggest that one should be wary of a machine (ML model) that cannot generalize in a fashion similar to humans. Lack of abstraction suggests memorization. How much trust should one place in a ML model that simply memorizes the test data? Ultimately, experiments E2-E4 help us better understand what features are important in EHD and how can we restrict our procedural generation pipeline. We are not in search of simulating an infinite amount of data across all variables and contexts. Nor are we convinced that such a philosophy is productive. No research has proven or demonstrated that ML performance continuously improves as one provides more data. Instead, we are interested in knowing and simulating or directing resource effective real world data collections to train a trustworthy machine that operates 24-7 and performs similar, if not better, than a human.

Table 4: Description of experiments.

Data Set	Parameters	Underlying Question
E1	(Train DS1, Test DS3), (Train DS4, Test DS6), (Train DS3, Test {DS1, DS2}), and (Train DS6, Test {DS4, DS5})	Experiments to assess the impact of operating in and out of context (time of day).
E2	(Train DS3, Test DS6)	Does a TOON model generalize to GEQ?
E3	(Train DS6, Test DS3)	Does a GEQ model abstract to TOON?
E4	(Train DS3, Test DS7) and (Train DS4, Test DS7)	Do models generalize to art?

Figure 12: Example GEQ imagery from morning (left), solar noon (center), and late afternoon (right). This imagery illustrates time of day differences in lux and shadows.

4.1 E1 : Sensitivity to Changes in Time of Day

The goal of E1 is to demonstrate and explore preliminary results for a procedural EHD pipeline to collect data and understand the impact of time of day on an EHD model. These results are not comprehensive and they are based on random generation of a diverse set of variable parameters vs. a highly controlled experiment. For example, we are not outlining results relative to a specific location on Earth, type of soil, emplacement context, exact time of day, etc. Since we cannot brute force nor grid search the object, environment, and platform/sensor variable space, random generation was selected to obtain the most diverse model possible for operating across contexts, versus operating in a specific context. While it is possible to use the proposed tools to study such a task, it is not the focus of our article. This section uses both TOON and GEQ. The aim is to observe of trends persist or contradict.

Figure 13 shows the quantitative performance difference between a TOON model trained for a specific time of day versus a TOON model trained across different times of day. Figure 16 shows corresponding example U-Net outputs for the TOON model. Figure 14 is GEQ ROC performance and Figure 15 are U-Net outputs for the GEQ model.

The ROCs illustrate that time of day has a big impact on our U-Net. Specifically, changes in time of day, manifested in illumination changes and shadows have an impact. We were primarily interested in understanding the impact of restricting real world data collection to a narrow window of time, e.g., if we just collect around solar noon will our models generalize? The reader could extend our process and explore questions like train across different times but evaluate at specific times. The result would be a characterization of algorithm performance at a specific time of day. The reader can see from the U-Net outputs that the majority of mistakes are correlated illumination differences and shadows. In future work we will export shadow layers from UE to measure shadow umbra, penumbra and antumbra. This per-pixel information will help us better measure and assess specific failure cases. The idea being, we can take all error objects or pixels and calculate statistics around those locations to observe and understand to what degree factors like occlusion, shadow, specific illumination, or illumination differences played in a single failure, or failure across a collect.

On a final note, we focused primarily on our procedural EHD pipeline and subsequent ML training and evaluation. In future work, we will expand our research to study if common data augmentation methods can

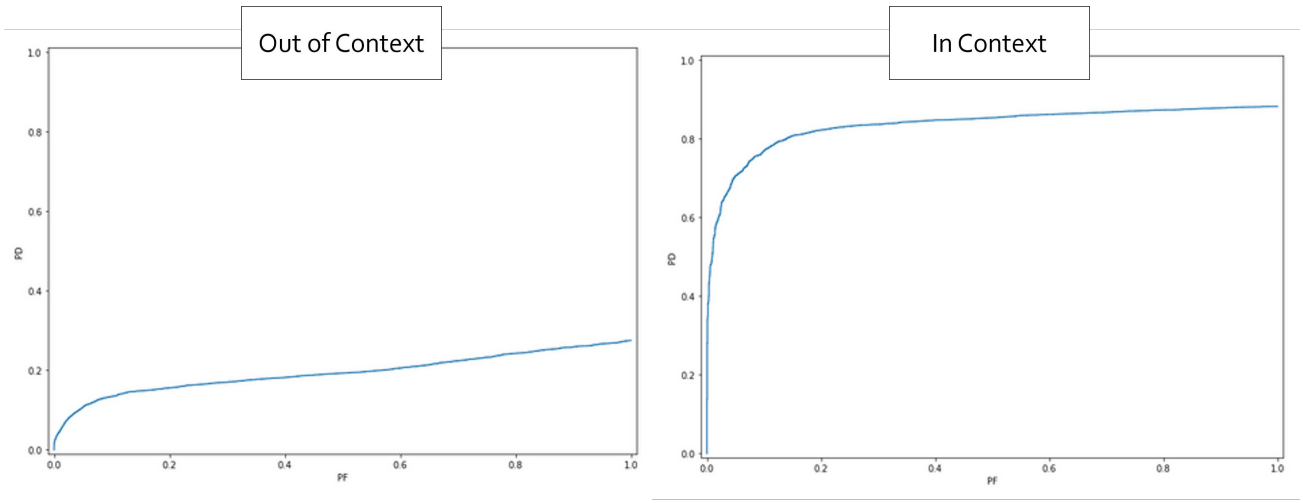


Figure 13: ROC curve results for a TOON model in "out of context" (train on solar noon and test across different times of day) and "in context" (train and test across different times of day). The y-axis shows probability of detection while the x-axis shows probability of false alarm. We do not report exact PF due to the sensitivity of EHD. However, the reader can observe the relative differences across models and datasets.

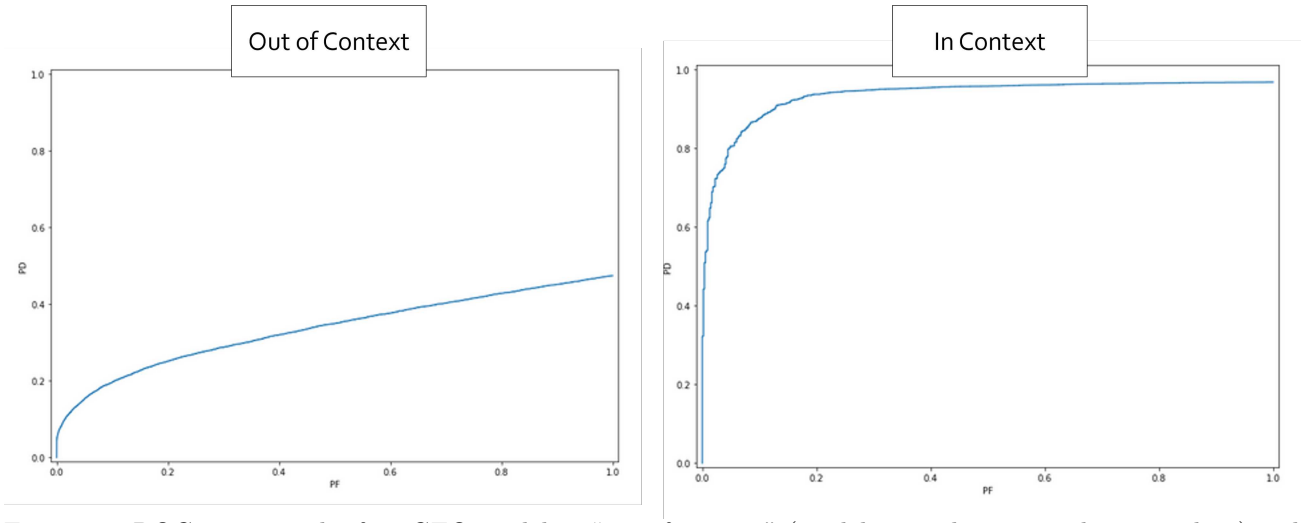


Figure 14: ROC curve results fora GEQ model in "out of context" (model trained on just solar noon data) and "in context" (model trained and tested across times of day).

help raise performance. Data augmentation (contrast, rotation, noise, etc.) has been a major performance boost in our modern era of deep learning; e.g., YOLOv5.[39] In summary, we suggest that one could likely reduce the time of day performance gap by enabling augmentation. However, it should be noted that these methods are not physically accurate. That is, while they can adjust relatively simple statistics like mean intensity and contrast, they do not insert realistic shadows.

4.2 E2 : Do TOON Models Generalize to GEQ Data?

In this subsection, we explore if a TOON model can generalize to GEQ data. This subsection is rooted in qualitative analysis. This is driven by the fact that we had little intuition initially regarding what to expect and how does one "prove" such a concept? These experiments are a starting point for us to learn from and help frame more well-structured future experiments.

The reader can see (Figure 17) that the TOON model appears to generalize to some degree. Specifically, there are a good number of PDs, less than on TOON, and more FAs. Why does the TOON model work? The TOON

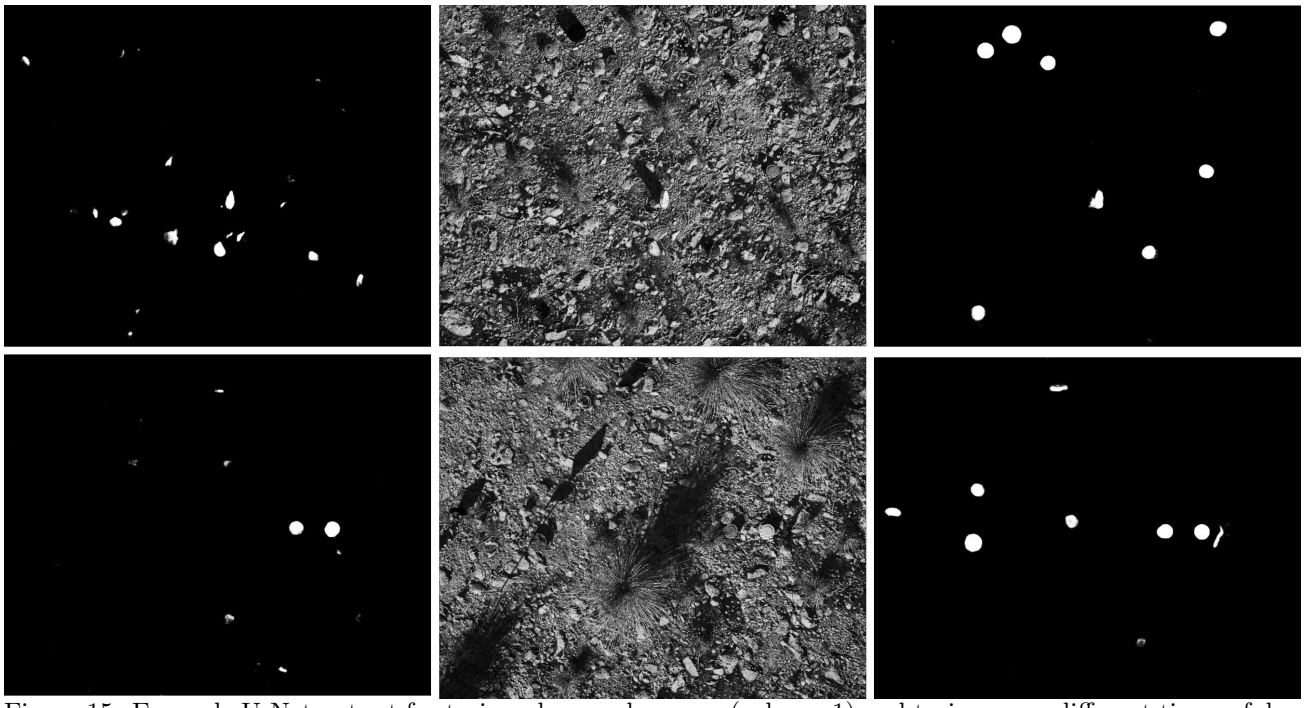


Figure 15: Example U-Net output for train only on solar noon (column 1) and train across different times of day (column 3) for a GEQ model. The middle column is example GEQ images and rows show corresponding data.

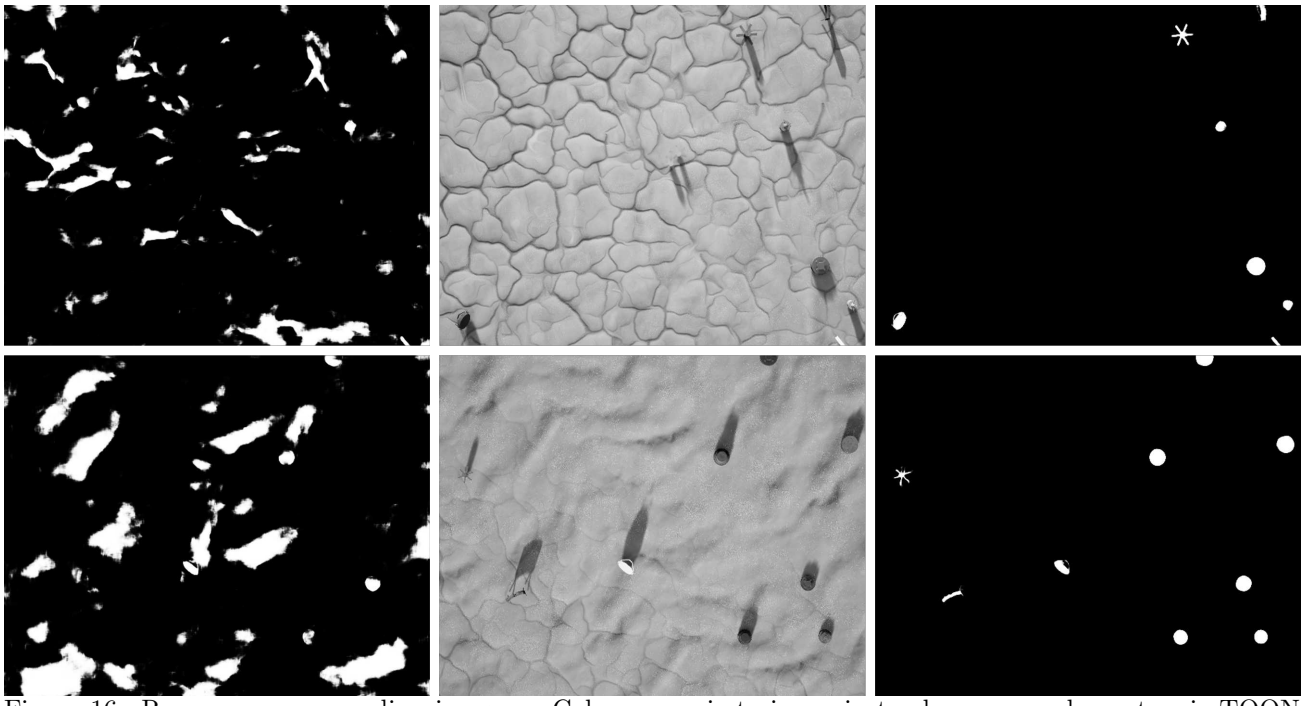


Figure 16: Rows are corresponding imagery. Column one is train on just solar noon, column two is TOON imagery, and column three is train and test across all times of day for a TOON model.

and GEQ imagery are clearly *different*. Our visual inspection has highlighted that most correctly identified targets are primarily shape and contrast based. The majority of missed targets appear to be correlated with regions of rich texture. When the machine is presented with additional information, specifically higher frequency

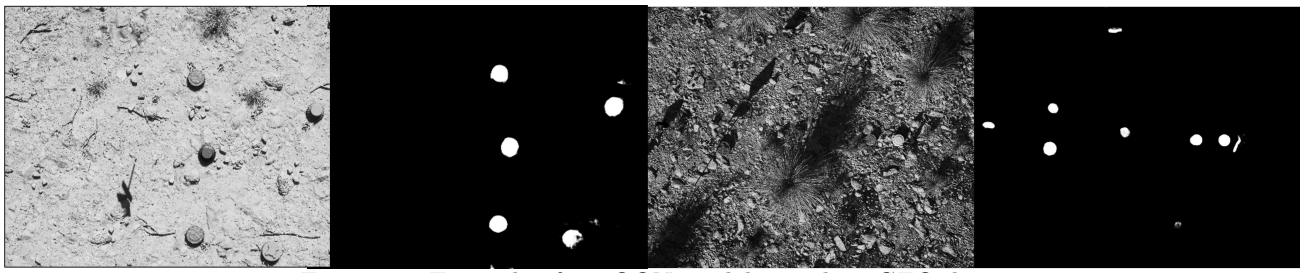


Figure 17: Example of a TOON model tested on GEQ data.

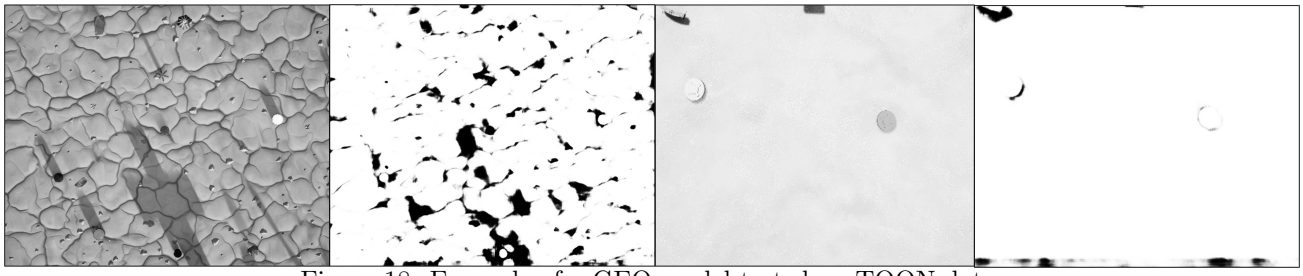


Figure 18: Example of a GEQ model tested on TOON data.

texture features versus lower frequency shape features, the U-Net declares pixels as non-target versus target. The model works well on TOON, which is shape and contrast rich. The model falters in texture rich GEQ areas. Does this imply that the model has learned reliable features that we believe are important to EHD? Furthermore, does this imply that the model is poor at the decision making level in light of these new high energy texture features? This is interesting because if it is true then it suggests that we should investigate locking TOON features and training the higher decision making levels on a combination of TOON and GEQ data. Alternatively, it suggests exploring style augmentation. In Geirhos et al.,[22] used artistic style augmentation of data to emphasize shape versus texture. While they used a *generative adversarial network* (GAN) called StyleGAN, a work like ours could do it directly via simulation in controlled ways. Performance gain using StyleGAN appears to be based on the principle that if a machine is presented with the task of memorizing hundreds or thousands of variations of the same image, with changes in features, that shape is the most consistent feature and the machine would do wise to learn it. In essence, style augmentation for shape learning is a trick during learning to make a machine go against its natural tendencies and learn features that we find important.

4.3 E3 : Do GEQ Models Abstract to TOON?

This subsection is about GEQ model performance on more abstract TOON data. In particular, the GEQ model classifies nearly the entire image incorrectly as target (see Figure 18). Why? First, the GEQ model makes mistakes in nearly all locations of the image and it was trained across times of day and performed well on GEQ data. TOON data possesses simpler geometry and shape, simplified illumination and shadows, and simpler or no texture. However, error does not seem to be localized to objects, shadows, or a simple change in intensity or contrast. While future experiments are needed to narrow and experimentally back our intuition, the fact that error is distributed across the image suggests that the change or absence of texture is perhaps the biggest factor. As shown in,[22] modern neural networks learn and are largely driven by texture versus shape. It is logical to believe that a model trained on GEQ data is biased towards texture. These features are abundant and have relatively large magnitudes (energies) that can fool a machine. These false correlations are dangerous. While they perhaps work well on training and test data that are similar, memorizing these features is not a reliable way to generalize. A machines learned features dictate its "vector span space" – what statements it can form in a language – and ultimately what set of functions it is capable of approximating. Further analysis is needed to determine if the learned GEQ model features are capable of abstracting to TOON or if blame resides in how the GEQ model has learned to use these features. Regardless, this experiment suggests that a machine may tend to overly listen to texture and miss other important features present in the training data.

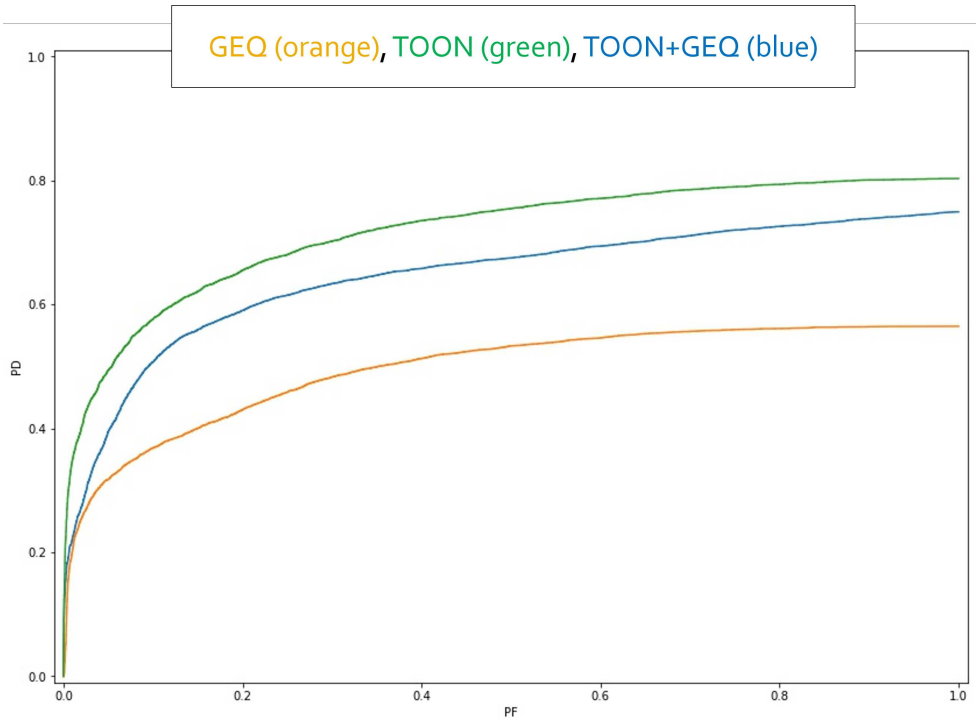


Figure 19: Quantitative ROC curve results showing how well TOON, GEQ, and TOON+GEQ trained models abstract to highly stylized or artistic imagery.

4.4 E4 : Do {TOON, GEQ} Models Abstract to Highly Stylized Imagery?

This subsection describes an open ended set of experiments centered around simulated stylized EH imagery (see Figure 11). The experiment is exploratory in nature and we do not know what types of stylization to apply. Our idea is to study the results and relate our algorithms performance back to underlying core features like color, texture, contrast, shape, and intensity.

In general, we have observed similar TOON model behavior to Sections 4.2 and 4.3. Namely, a TOON models *positive detection rate* (PDR) is decreased by there is not a noteworthy increase in the FAR. The TOON models mistakes do not appear to occur around points of stylization, e.g., artistic specular highlights or cel shaded edges or color saturation. Instead, our manual analysis has revealed that the largest source of error is regions with texture. The stylization with the lowest PDR is dithering (row four in Figure 20). On the other hand, while the GEQ model appears to abstract in a few scenarios, across the board it results in a large FAR increase. We take this result with a grain of salt as GEQ imagery is "further away" from stylized imagery than TOON.

ROCs were generated for three models trained using TOON, GEQ, and a combination of TOON and GEQ data (TOON+GEQ). In Figure 19, the reader can clearly see that the TOON model does best, followed by TOON+GEQ then GEQ. Again, while preliminary, this is interesting for a number of reasons. GEQ has more detailed information regarding shape, texture, illumination, and shadows. However, the artistic imagery considered here does not possess more detailed shape, they are often absent of texture or possess texture that does not look real, and illumination and shadows are discretized. Can a decrease in model performance be explained by saying that the TOON+GEQ model learned a new set of features that are too far away from art? From an information theoretic standpoint, are GEQ features more like the real world but a distraction with respect to abstraction? A data-driven machine can be discussed in terms of what features and decision making functionality it can compute and what it learned from a set of data and training algorithm. More structured future experiments are needed to determine if the TOON+GEQ model learned useful features that can be applied to both TOON and GEQ, but not how to successfully exploit these features in a new and novel situation.

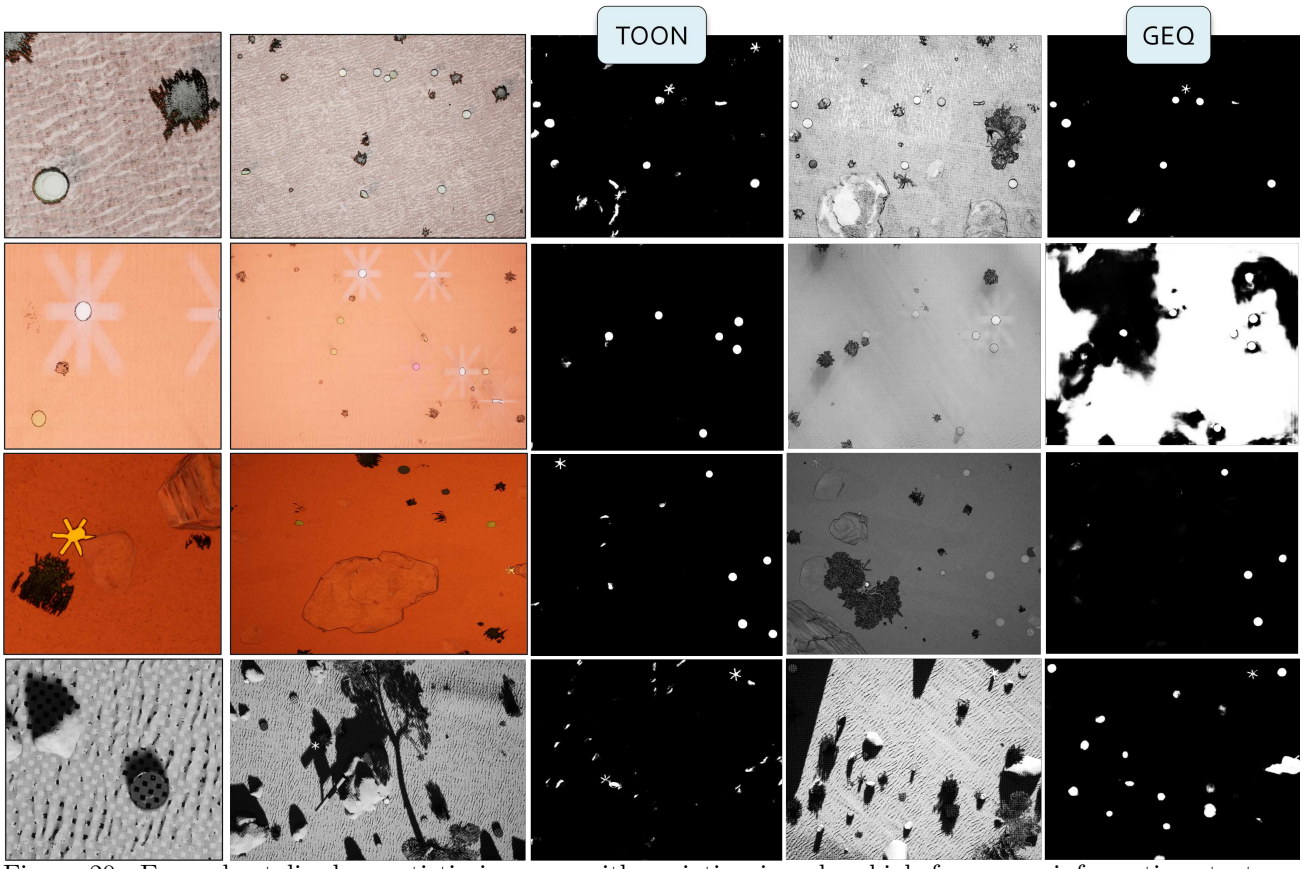


Figure 20: Example stylized or artistic imagery with variation in color, high frequency information, texture, specular highlights, etc. Column two are example effects and column one is zoomed in regions of interest. Columns three, four, and five are corresponding U-Net result pairs. Column three is TOON model, column four is the input image, and column five is GEQ model.

5. CONCLUSIONS AND FUTURE WORK

In this article, we presented an initial constrained procedural scene and data collection framework and workflows in the Unreal Engine for simulation of data across the reality spectrum. The reader should note that the signatures being exploited here derive from line of sign spatial features, the targets under consideration are surface-deployed, and the solution space being worked is electro-optical/infrared against line-of-sight EH only. Example imagery from current game engine quality (GEQ) to cartoon (TOON) and highly stylized or artistic imagery was generated. A variety of experiments relative to U-Net-based pixel-level semantic segmentation were performed in and across the visual categories. Specifically, we found that both GEQ and TOON models are highly sensitive to the time of day that data is collected. Both experiments followed similar processes and found similar degradation trends. We also showed that TOON models generalize to GEQ data, while GEQ does not abstract well to TOON imagery. Furthermore, TOON appears to abstract to highly stylized or artistic imagery, while GEQ not so much.

There is a large body of future work that is needed. First, this is our initial work. A greater number of models and procedural data sets need to be generated and characterized. For example, each experiment should be performed under the pseudo-random guidelines outlined and ROC curves showing average and standard deviation are needed to better understand how consistent the results are and to what degree the separation in performance is due to the underlying process versus the training of an individual model. Second, we cast a pseduo-random net to address the sheer complexity of the underlying object, environment, and platform/sensor variable space. Future focused studies with further constrained bounds can help us better understand how useful

this process is and help us address real world application domain questions. Third, further studies where we pick TOON or art effects and content and better control, and "linearly vary", visual features like shape, contrast, intensity, texture, illumination, and shadows are needed. The initial idea was to set up a sandbox and explore. However, our findings suggest that the machine is learning perhaps shape and contrast and other features from a domain like TOON. How can this be "proven", or experimentally further supported? Furthermore, if this is true, what can we do to further emphasize or encourage improved behavior in light of new information, e.g., rich texture in GEQ? As this is simulation, we can pose and explore these questions much better than in the real-world.

Last, in[42] we showed that GEQ and TOON simulated data led to a U-Net that works on real world aerial EH data. In future work, that article and this article need to be combined to develop better ways of testing simulation discoveries in the real-world. While our goal is not to build a "digital twin", there needs to be a way to verify simulation discoveries and transfer them to successful real world models or knowledge to help improve real world data collections.

6. ACKNOWLEDGMENTS

This research is funded by ARO grant number W911NF1810153 to support the U.S. Army DEVCOM C5ISR Center. This research would not be possible without our collaborators.

REFERENCES

[1] "If data is the new oil, these companies are the new Baker Hughes," (2021).

[2] Anderson, D. T., Stone, K. E., Keller, J. M., and Spain, C. J., "Combination of anomaly algorithms and image features for explosive hazard detection in forward looking infrared imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **5**(1), 313–323 (2012).

[3] Gader, P. D., Mystkowski, M., and Yunxin Zhao, "Landmine detection with ground penetrating radar using hidden markov models," *IEEE Transactions on Geoscience and Remote Sensing* **39**(6), 1231–1244 (2001).

[4] Dowdy, J., Brockner, B., Anderson, D. T., Williams, K., Luke, R. H., and Sheen, D., "Voxel-space radar signal processing for side attack explosive ballistic detection," in [*Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXII*], Bishop, S. S. and Isaacs, J. C., eds., **10182**, 421 – 435, International Society for Optics and Photonics, SPIE (2017).

[5] Havens, T., Anderson, D. T., Stone, K. E., Becker, J., and Pinar, A. J., "Computational intelligence methods in forward-looking explosive hazard detection," in [*Recent Advances in Computational Intelligence in Defense and Security*], (2016).

[6] Gaidon, A., Wang, Q., Cabon, Y., and Vig, E., "VirtualWorlds as proxy for multi-object tracking analysis," in [*2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 4340–4349 (2016).

[7] Chen, C., Seff, A., Kornhauser, A., and Xiao, J., "DeepDriving: Learning affordance for direct perception in autonomous driving," in [*2015 IEEE International Conference on Computer Vision (ICCV)*], 2722–2730 (2015).

[8] Wymann, B., Dimitrakakisy, C., Sumnery, A., and Guionneauz, C., "TORCS: The open racing car simulator," (2015).

[9] Martinez, M., Sitawarin, C., Finch, K., Meincke, L., Yablonski, A., and Kornhauser, A., "Beyond Grand Theft Auto V for training, testing and enhancing deep learning in self driving cars," (2017).

[10] Martinez-Gonzalez, P., Oprea, S., Garcia-Garcia, A., Jover-Alvarez, A., Orts-Escolano, S., and Garcia-Rodriguez, J., "UnrealROX: An extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation," *ArXiv e-prints* (2018).

[11] Müller, M., Casser, V., Lahoud, J., Smith, N., and Ghanem, B., "Sim4CV: A photo-realistic simulator for computer vision applications," *Int. J. Comput. Vision* **126**, 902–919 (Sept. 2018).

[12] Drouin, M., Fournier, J., Boisvert, J., and Borgeat, L., "Modeling and simulation framework for airborne camera systems," in [*ICPR Workshops*], (2020).

[13] Nouduri, K., Gao, K., Fraser, J., Yao, S., AliAkbarpour, H., Bunyak, F., and Palaniappan, K., "Deep realistic novel view generation for city-scale aerial images," in [*2020 25th International Conference on Pattern Recognition (ICPR)*], 10561–10567 (2021).

[14] Alvey, B., Anderson, D. T., Buck, A., Deardorff, M., Scott, G., and Keller, J. M., "Simulated photorealistic deep learning framework and workflows to accelerate computer vision and unmanned aerial vehicle research," in [*2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*], 3882–3891 (2021).

[15] Deardorff, M., Alvey, B., Anderson, D. T., Keller, J. M., Scott, G., Ho, D., Buck, A., and Yang, C., "Metadata enabled contextual sensor fusion for unmanned aerial system-based explosive hazard detection," in [*SPIE*], (2021).

[16] Alvey, B., Anderson, D. T., Keller, J. M., Buck, A., Scott, G., Ho, D., Yang, C., and Libbey, B., "Improving explosive hazard detection with simulated and augmented data for an unmanned aerial system," in [*Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXVI*], Bishop, S. S. and Isaacs, J. C., eds., **11750**, 76 – 90, International Society for Optics and Photonics, SPIE (2021).

[17] Larsson, J., *Performance of Physics-Driven Procedural Animation of Character Locomotion*, PhD thesis, Blekinge Institute of Technology (2015).

[18] Olsen, J., "Realtime procedural terrain generation," (2004).

[19] Perlin, K., "An image synthesizer," in [*Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*], *SIGGRAPH '85*, 287–296, Association for Computing Machinery, New York, NY, USA (1985).

[20] "The AI Systems of Left 4 Dead." https://steamcdn-a.akamaihd.net/apps/valve/2009/ai_systems_of_l4d_mike_booth.pdf. (Accessed: 1 March 2021).

[21] "Houdinin Unreal Engine Plugin." https://www.sidefx.com/products/houdini-engine/plug-ins/unreal-plug-in/. (Accessed: 1 March 2021).

[22] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W., "Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.," in [*International Conference on Learning Representations*], (2019).

[23] Hinterstoisser, S., Lepetit, V., Wohlhart, P., and Konolige, K., [*On Pre-trained Image Features and Synthetic Images for Deep Learning: Munich, Germany, September 8-14, 2018, Proceedings, Part I*], 682–697 (01 2019).

[24] "Unity." https://unity.com/. (Accessed: 1 March 2021).

[25] Rohde, M. M., Crawford, J., Toschlog, M. A., Iagnemma, K., Kewlani, G., Cummins, C. L., Jones, R. A., and Horner, D. A., "An interactive physics-based unmanned ground vehicle simulator leveraging open source gaming technology: progress in the development and application of the virtual autonomous navigation environment (vane) desktop," in [*Defense + Commercial Sensing*], (2009).

[26] "Storytelling reimagined," (2021).

[27] "Real-time ray tracing," (2021).

[28] "UE Architecture." https://www.unrealengine.com/en-US/solutions/architecture. (Accessed: 1 March 2021).

[29] "Quixel." https://quixel.com/. (Accessed: 1 March 2021).

[30] "Free Vigilante UE Models." https://www.unrealengine.com/marketplace/en-US/profile/Vigilante?count=20&sortBy=effectiveDate&sortDir=DESC&start=0. (Accessed: 1 March 2021).

[31] "Instant Terra Unreal Engine Plugin." https://www.unrealengine.com/marketplace/en-US/item/0c08f0ce7ac04724931565b2386012d0. (Accessed: 1 March 2021).

[32] "Unreal Marketplace." https://www.unrealengine.com/marketplace/en-US/store. (Accessed: 1 March 2021).

[33] "Cel Toon Outline Post Process Material." https://www.unrealengine.com/marketplace/en-US/item/65cd54d6adcc4b47a6c383d579beda4c. (Accessed: 1 March 2021).

[34] "Post Process Toolkit." https://www.unrealengine.com/marketplace/en-US/item/94cebc0ec72945899a494cf724f96293. (Accessed: 1 March 2021).

[35] "Real City SF - Downtown Environment Mega Pack." https://www.unrealengine.com/marketplace/en-US/item/0bf112c1f34841a3b4d98a768b2d4236. (Accessed: 1 Jan 2022).

[36] "MEGASCANS TREES ARE NOW IN EARLY ACCESS ." https://quixel.com/blog/2021/12/15/megascans-trees-are-now-in-early-access. (Accessed: 1 Jan 2022).

[37] "Post Process Toolkit." https://www.unrealengine.com/marketplace/en-US/item/94cebc0ec72945899a494cf724f96293. (Accessed: 1 March 2021).

[38] Ronneberger, O., P.Fischer, and Brox, T., "U-net: Convolutional networks for biomedical image segmentation," in [*Medical Image Computing and Computer-Assisted Intervention (MICCAI)*], *LNCS* **9351**, 234–241, Springer (2015). (available on arXiv:1505.04597 [cs.CV]).

[39] Jocher, G., Stoken, A., Borovec, J., NanoCode012, ChristopherSTAN, Changyu, L., Laughing, tkianai, yxNONG, Hogan, A., lorenzomammana, AlexWang1900, Chaurasia, A., Diaconu, L., Marc, wanghaoyang0106, ml5ah, Doug, Durgesh, Ingham, F., Frederik, Guilhen, Colmagro, A., Ye, H., Jacobsolawetz, Poznanski, J., Fang, J., Kim, J., Doan, K., and Yu, L., "ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration," (Jan. 2021).

[40] Alvey, B., Anderson, D., Yang, C., Buck, A., Keller, J., Yasuda, K., and Ryan, H., "Characterization of deep learning-based aerial explosive hazard detection using simulated data," in [*SPIE*], (2021).

[41] Li, C., Tan, Y., Chen, W., Luo, X., Gao, Y., Jia, X., and Wang, Z., "Attention unet++: A nested attention-aware u-net for liver ct image segmentation," in [*2020 IEEE International Conference on Image Processing (ICIP)*], 345–349 (2020).

[42] Kovaleski, M., Fuller, A., Kerley, J., Alvey, B., Popescu, P., Anderson, D., Buck, A., Keller, J., Scott, G., Yang, C., Yasuda, K., and Ryan, H., "Combining high-quality ground truthed simulation data with imprecise real data for training a per-pixel aerial explosive hazard detection pre-screener," in [*SPIE*], (2022).