Improving Explosive Hazard Detection with Simulated and Augmented Data for an Unmanned Aerial System

Brendan Alvey[a], Derek T. Anderson[a], James M. Keller[a], Andrew Buck[a], Grant Scott[a], Dominic Ho[a], Clare Yang[b], and Brad Libbey[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

Modern supervised machine learning for electro-optical and infrared imagery is based on data-driven learning of features and decision making. State-of-the-art algorithms are largely opaque and questions exist regarding their interpretability and generalizability. For example, what are the learned features, what contexts do they work in, and are the algorithms simply memorizing observations and exploiting unwanted correlations or has it learned an internal representation and causal associations that generalize to new environments? Under the hood, current convolutional neural networks (CNN) are sophisticated *curve fitters* that are sensitive to sampling (volume and variety). This is problematic as collecting data from real systems is often expensive and time consuming. Furthermore, labeling and quality checking of that data can also be prohibitive. As a result, many are looking to augmentation and simulation to efficiently generate more samples. Herein, we focus on ways to combine augmentation and simulation to improve explosive hazard detection. Specifically, we use the Unreal Engine to produce ray traced simulated data sets of environments and emplacements not captured in real data. We also present a new technique, coined altitude modulated augmentation (AMA), that inserts simulated objects into real world background imagery based on metadata to augment new training data. Thus, the goal of AMA is to increase sampling of observed environments. Preliminary results show that the combination of all techniques is best, followed by augmentation, simulation, then real world data.

Keywords: augmentation, drone, explosive hazard, simulation, unmanned aerial vehicle, Unreal Engine, YOLO

1. INTRODUCTION

Explosive hazard detection (EHD) has been a problem for a long time. The task of detecting EHs is one better suited for a drone than a precious human being. Drones are replaceable and they offer a way to keep humans at a safe standoff distance. They can be equipped with high resolution cameras in different portions of the electromagnetic spectrum as well as position sensors (e.g., GPS and IMU). EHs vary in size, shape, and composition. They can be placed in a variety of environments and contexts, e.g., orientations, occlusion, etc. Furthermore, imagery collected from an unmanned aerial vehicle (UAV) at an altitude of 30 meters looking straight down (nadir) looks vastly different from a UAV at 10 meters looking straight ahead. The point is, UAV-based EHD has great potential, but it is a complex technology full of sub-challenges.

EHD approaches to date vary drastically. 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 explosive 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,[1] downward looking,[2] and even side looking.[3] Herein, we focus on the use of UAVs, which can operate in each of the above modalities. This method of delivery has the potential to help us search areas faster, especially in the case of a swarm of UAVs, and dynamically interrogate regions of interest.

Obtaining real world data sets from UAVs is often time consuming and expensive. An enormous amount of effort is required to plan and coordinate data collects. Each collect requires a skilled pilot, charged batteries, flight plans, a working UAV, and acceptable weather for flying. In addition, each target emplacement should be well documented with accurate GPS positions and pictures of the surrounding context recorded. For training supervised learning approaches and for scoring purposes, annotations are required. Accurately labeling images of EHs frequently requires the assistance of skilled experts who painstakingly draw bounding boxes and assign class labels to objects. Furthermore, the global COVID-19 pandemic has made coordinating physically with one another difficult. Meeting in person for data collections, at the moment, requires extra safety precautions in addition to being subject to delays more often. Each of these requirements and conditions puts a bottleneck on the volume of quality annotated data available for training and testing.

Modern data-driven machine learning algorithms, e.g., convolutional neural networks (CNNs), are sophisticated curve fitters that are dictated by sampling statistics. In the context of EHD, we are generally not blessed with high volume and variety data sets. Instead, we tend to work with a limited number of examples and each data collection only contains a limited number of target emplacements. This is not a favorable situation for modern feature and decision making learning. Furthermore, we would like our EHD algorithms to detect targets regardless of the background they are placed in. Our model should not learn that some targets are always near a particular rock or bush because it has only seen that target near a particular rock or bush. Similarly, our model should be robust to variations in brightness, contrast, focus, noise and other realistic collection parameters. In this paper, we explore two ways to combat the challenges above. First, we explore the use of modern simulation, using the Unreal game engine, to generate EH data for new environments and emplacements. These are situations that we do not expect to see in our real world collected data sets. Second, we explore the use of simulated target EH templates and real background images. The goal here is to make better use out of the data that we do have available. Overall, the best way to characterize our article is data augmentation.

The remainder of the article is organized as follows. In Section 2, we describe the coupling of simulation and real data. In Section 3, we discuss the use of simulation without real data. Last, in Section 4 we present experiments and preliminary results for each approach individually, and their combination relative to a control algorithm on a real annotated EH data set.

2. ALTITUDE MODULATED AUGMENTATION (AMA)

The point of this section is to combine the best of both worlds, simulation and real data. To facilitate the training of a more robust EHD model and to better leverage precious existing data, simulated target templates are inserted into real UAV imagery with random variations based on context provided by metadata. By using simulated target templates from the Unreal Engine,[4] shadow and target masks are automatically obtained. The shadow mask is used to implement a simple shadowing by darkening background pixels near an inserted target. The UAV that collected our data was flown over targets at fixed altitudes but there are still small variations in relative altitude due to turbulence and changes in ground height. By taking advantage of the on-board GPS unit, we use the relative altitude of the UAV to intelligently scale target templates before inserting them. We generate higher than required resolution simulated templates, which allows us to create training and validation examples at any reasonable operational altitude automatically via down sampling.

AMA has several advantages over using real data alone. As mentioned above, labeling data is a major bottleneck. With this approach we can generate perfect ground truth labels for every image almost instantly. If a new target type is to be added for detection, we would normally have to wait for a new collection with the new target included. We would also then have to wait for the data to be annotated before we could add it to our model. All that is needed with this method is a suitable target template or set of templates. Only approximately one in ten images are annotated. As a result, segments of collections flying over target emplacements are not suitable for generating training examples as they may contain unlabeled targets. Using existing imagery is in

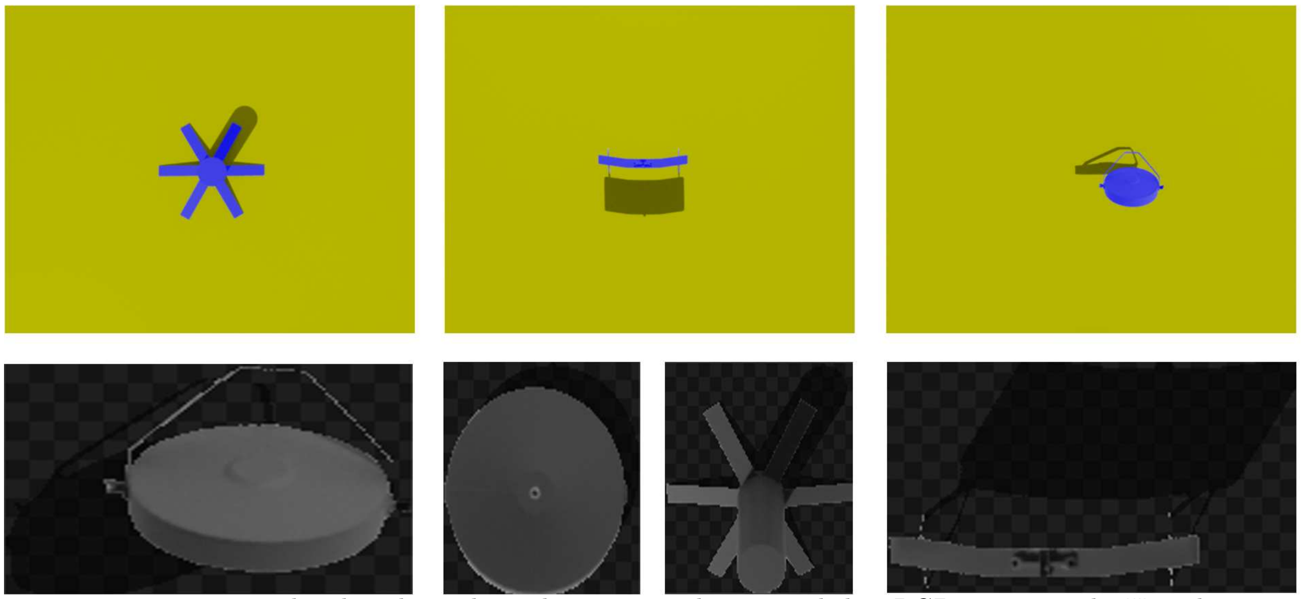


Figure 1: Top row: simulated explosive hazard target templates encoded as RGB images in the Unreal Engine. Bottom row: extracted grayscale target templates.

someways a double edged sword. On one hand, we know that the background images are realistic and should closely resemble environments found in target images. However, we are limited to only the environments for which data has already been collected.

The remainder of this section is a detailed description of the algorithms used to generate training and validation examples. The examples shown were generated from real images collected from a UAV. The template images were created by taking 3D models in the Unreal Engine and placing them on benign backgrounds. A custom shader with material properties and stencil buffers was used to export the templates as false color images from the Unreal Engine. Examples of these target templates can be found in Figure 1. The templates are encoded into color images with channels one and two containing the gray scale image and channel three containing a target mask. In this article, we focus on gray scale imagery-based detection. If color imagery is desired, then RGB imagery can be produced with the stencil packed into the alpha channel. Herein, shadow masks are extracted by determining which pixels are not part of the target mask but do have gray scale values below the background intensity. To facilitate easy insertion, target templates are loaded as images with an alpha transparency channel set to 0.5 for the shadow region, 1 for the target region and 0 everywhere else. The shadow region is set to 0 in the gray scale image. Combined with the transparency this has the effect of darkening the background to mimic shadows when inserted. There are a number of steps which the target templates, background images, and the combined target images undergo to add variation. Algorithm 1 describes the processes. Figure 2 shows examples of inserting our explosive hazard target templates into a different background images collected from a UAV.

The current article is focused on first steps in using simulation to augment real data. As such, we have only presented a few augmentations. In future work we will expand our procedures to consider additional features like motion blur, realistic shadows based on solar position, and more based on metadata. Furthermore, we will extend the algorithms and consider any additional data labeling and/or processing of UAV data (e.g., 3D mapping and semantic segmentation of a scene) to more intelligently determine emplacement context, e.g., generate targets that are partially occluded by nature or man made objects.

3. SIMULATION

In this section, we discuss the use of state-of-the-art modeling and simulation tools for data augmentation to advanced AI/ML. Specifically, we focus on modeling and producing high quality visual spectrum (aka RGB) imagery via ray tracing in the Unreal Engine.[4] To this end, we use a combination of free and for purchase

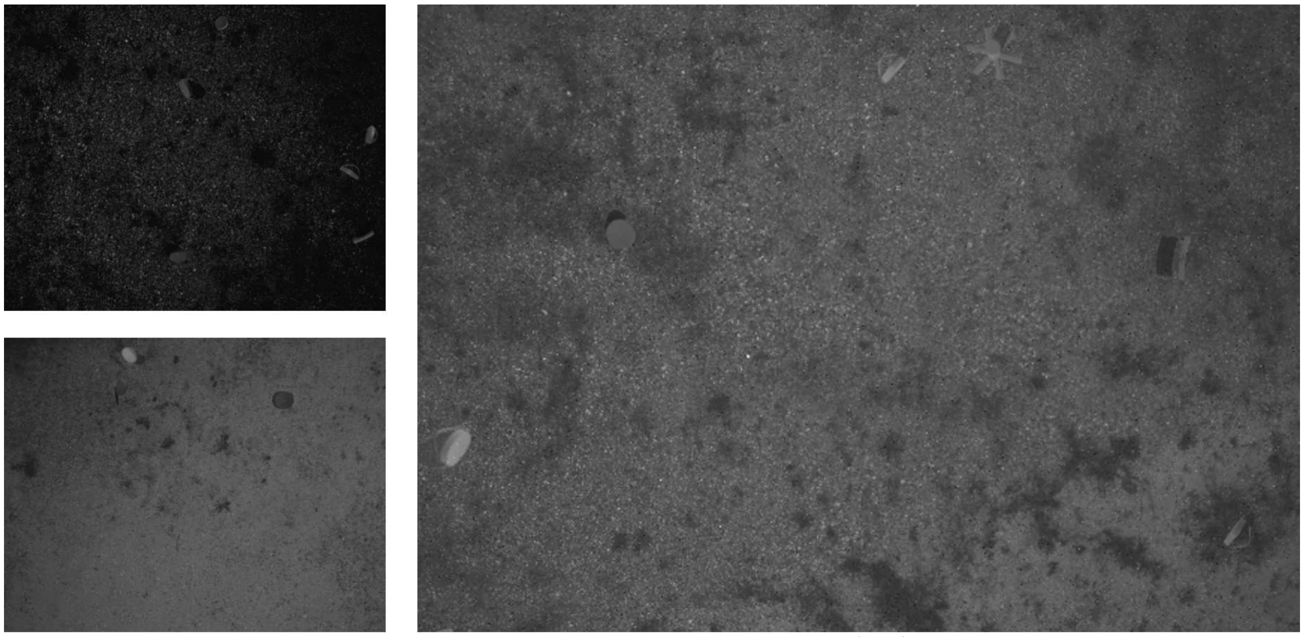


Figure 2: Three examples generated by AMA.

Algorithm 1 Altitude Modulated Augmentation (AMA)

```
 1: for Each generated image do
 2:     Randomly choose a background frame, $f$, to use.
 3:     Load background image, $I$ and corresponding altitude, $a$
 4:     Randomly choose number of targets to insert, $n$
 5:     for $n$ from 1 to $n$ do
 6:         Randomly choose a target template, $T$, to insert.
 7:         Gaussian blur $T$ with maximum random $\sigma$ of 1.5.
 8:         Rotate $T$ by a randomly chosen angle.
 9:         Apply hot-cold modification to $T$ with probability 0.2 described by Algorithm 2
10:         Add noise to 25 random pixels in $T$ and to 1000 pixels in $I$.
11:         Randomly scale target by $\pm 10\%$.
12:         Scale target by $s = 2.5/a$
13:         Randomly vary $T$ and $I$ brightness by up to 25 in either direction.
14:         Insert $T$ into $I$ at a random location, creating the target image, $I_T$
15:         Gaussian blur $I_T$ with maximum random $\sigma$ of 1.0.
16:         Apply compression modification to $I_T$, described by Algorithm 3.
17:         Save image and add entries to ground truth.
18:     end for
19: end for
```

Algorithm 2 Hot Cold Modification

```
1: Given an image, $I$:
2: Set $max\_brightness = 75$.
3: Set $min\_brightness = 25$.
4: Generate a random number, $r$ between 0 and 1.
5: Determine $bright\_shift = (max\_brightness - min\_brightness) * r + min\_brightness$
6: Generate another random number, $r_2$ between 0 and 1.
7: if $r_2 > 0.5$ then
8:     $bright\_shift = -1.0 * bright\_shift$
9: end if
10: $I = clip(I + bright\_shift, 0, 255)$
```

Algorithm 3 Compression Modification

```
1: Given an image, $I$:
2: Determine $img\_max = max(I)$.
3: Determine $img\_min = min(I)$.
4: Set $max\_compression = 25$.
5: if $max\_compression * 2 > img\_max - img\_min$ then
6:     $max\_compression = 0.5 * (img\_max - img\_min)$
7: end if
8: Generate a random number, $r$ between 0 and 1.
9: $compression\_level = r * max\_compression$
10: Re-scale values in $I$ from $img\_min + compression\_level$ to $img\_max - compression\_level$
```

content (3D models and textures) from the Unreal Marketplace[5] and TurboSquid.[6] 3D targets and textures were authored in house. Figure 3 shows example content in the Unreal Editor and Figure 4 are example environments for our application that we composed for drone imaging from available content. It is worth noting that Unreal's ray tracing, and real-time enabled by NVIDIA hardware like the GeForce RTX 3090, provide access to high fidelity rendering capable of mimicking real cameras; e.g., motion blur, fstop, focal distance, FOV, pixel resolution, noise, and much more. This provides flexibility in mimicking different systems, making simulated data more like real-world data.

The majority of data that we collect for training is based on making a Camera object in Unreal. While we use a pinhole camera model, the reader can refer to Ref. 7 for an Unreal Engine AirSim plugin with more advanced camera modeling features. We then create a Cinematic Track and last, use the Movie Render Queue to achieve a pre-scripted flight pattern like a drone. This process is extremely controlled and is pristine. The advantage of this route is that rendering can be done with higher quality ray tracing offline, versus approximated in real-time using hardware like the GeForce RTX 3090. This gives us great control, e.g., use of Deferred Rendering (via Path Tracer), multiple spatial and temporal samples for anti-aliasing, specification of maximum number of ray bounces, etc. This procedure has allowed us to achieve the desired level of quality of imagery for our experiments. Figure 5 shows our rendering pipeline for automated ground truthing. We produce both RGB images and also false color imagery where channels are grayscale imagery, depth information, and object IDs, per pixel. This allows us to automatically generate a tremendous amount of training data for semantic segmentation (e.g., a U-Net[8]) and/or bounding boxes (by running connected components on the object ID channel) that can be directly fed to well-known detection and localization algorithms like YOLO.[9,10]

However, it is worth mentioning that we also have used the Unreal Engine extension AirSim.[11] AirSim is a plug in for Unreal for mimicking aerial and ground vehicles. Furthermore, we use the Robotic Operating System (ROS)[12] to communicate between the Unreal Engine, AirSim, and our custom algorithms. This coupling enables the real-time simulation of drones flying around in environments with support for sensors like GPS, IMU, LiDAR, and RGB. An example is shown in Figure 6. Like others, we have extended this framework and created a bridge between 3D modeling and ROS to achieve sensors like RGBD (where D is either active or passive). The reader can refer to Ref. 13 for an AirSim extension for infrared imagery. In summary, while we use cinematic

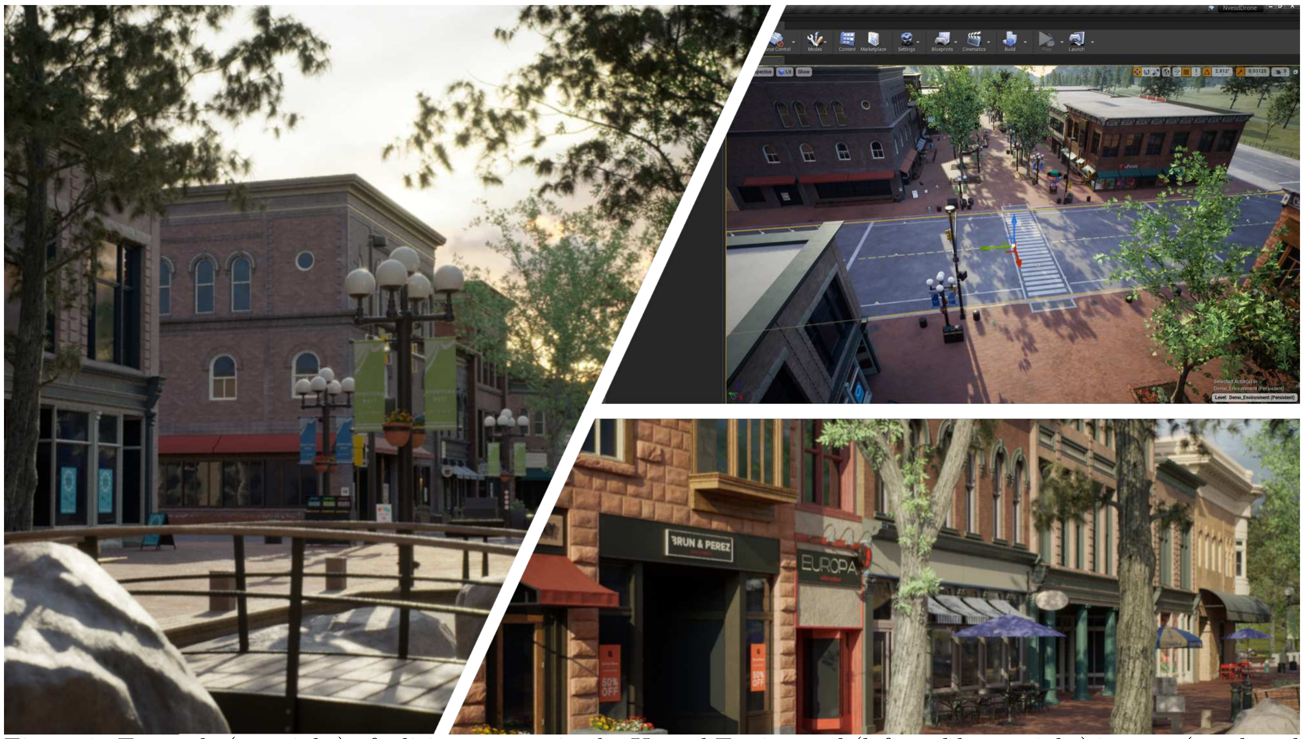


Figure 3: Example (top right) of editing a scene in the Unreal Engine and (left and lower right) images (produced using ray tracing and the Movie Render Queue).

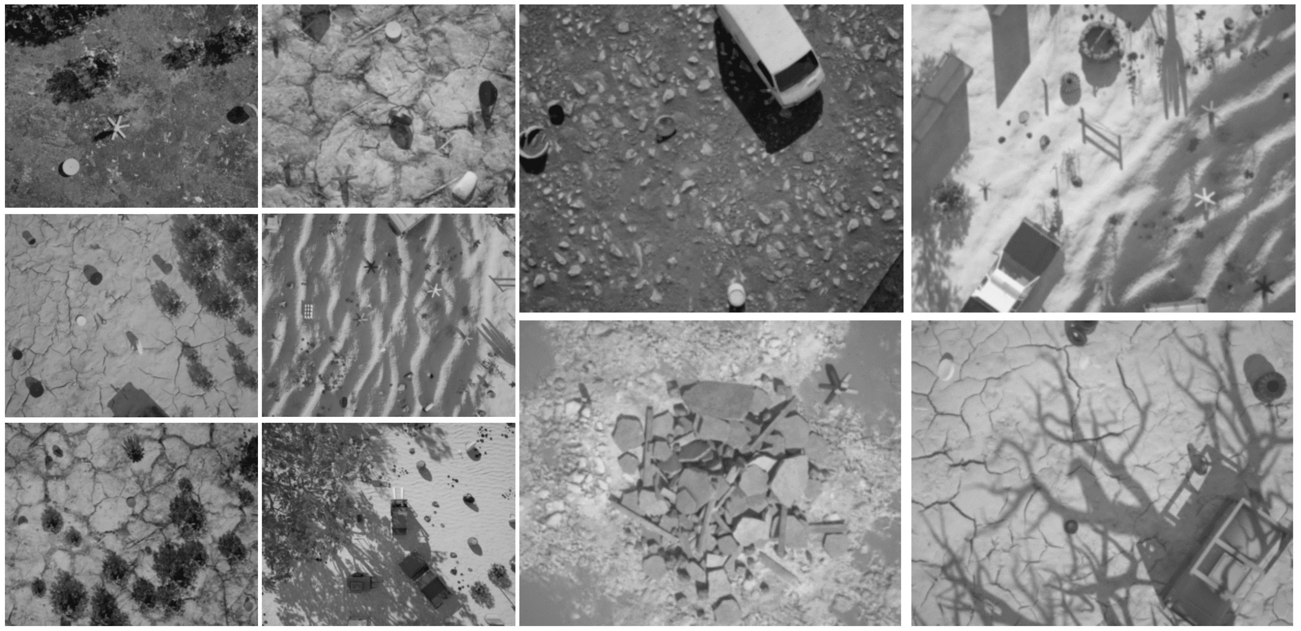


Figure 4: Arbitrarily selected Unreal Engine imagery for some of our modeled environments, whose content (3D models and textures) were authored by others. The imagery is generated from a simulated UAS with a specified RGB camera (e.g., particular f-stop, FOV, etc.) at different altitudes, sensor look angles, and environment variation (e.g., position of sun and shadows, time of day, etc.).

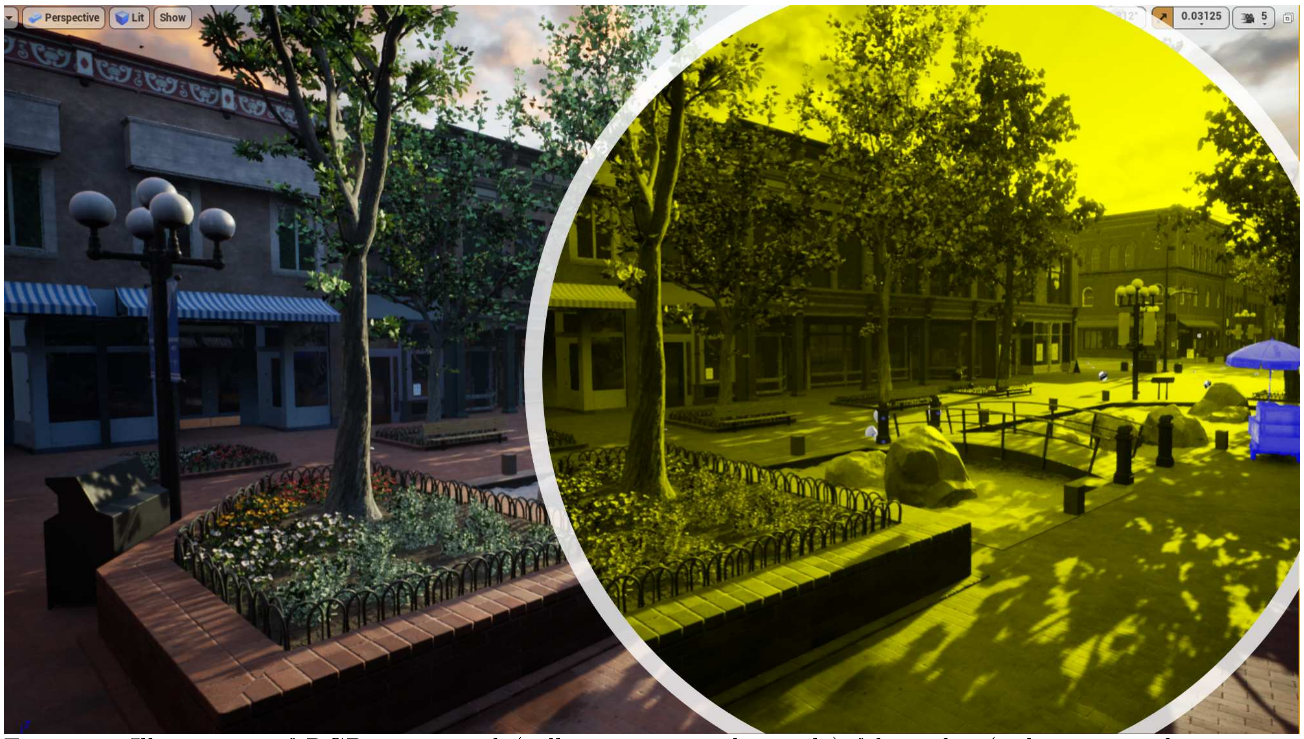


Figure 5: Illustration of RGB image and (yellow image in the circle) false color (red = grayscale, green = grayscale, blue = object ID) stencil buffer for per pixel ground truth.

rendering to produce training data, our coupling of Unreal, AirSim, ROS, and our custom algorithms (detection, tracking, mapping, and autonomy) enables two things. The first is the ability to fly around in simulation and test models. Instead of building full data sets, we often fly around changing factors and watch the algorithms to get a feel for their breaking points. Second, which we discuss in our paper on a Hardware and Simulation Platform for Visually Aware Drone Autonomy Research (VADER),[14] our algorithms and real-time codes, from detection to mapping and autonomy, are an abstract pipeline that run in both simulation and on a real drone (via technologies like an NVIDIA Jetson and MAVLink messages to the flight controller in ROS). The point is, simulation provides a bridge for rapid research that can be more quickly transitioned to a real-world solution.

Herein, we discuss the following factors relative to generating useful simulated training data. Factor 1 (F1) considers purposeful scenes or random configuration of objects (see Figure 7). Factor 2 (F2), what viewing or drone flight pattern to use is shown in Figure 8. Factor 3 (F3), which we call purposeful scenes or “floating islands” is shown in Figure 9. Last, Factor 4 (F4), environmental factors is shown in Figure 10. We have experimented with different configurations of each of these factors. Through experimentation, aka we do not have a proof, we have observed the following. For F1, we prefer “random” (arbitrary is probably a better term) scenes in order to get as many looks on targets and scene objects possible. The research outlined in this paper is closer to what we call *level 1 visual intelligence*. That is, we are interested in training AI/ML algorithms that can detect objects in and across sensors/spectra. Higher, and subsequent, AI/ML is needed to reason about data/information, likely in context provided by environment and platform/drone metadata. Our experiments have likely preferred F1 because it provides more looks at low-level visual features in different contexts.

Next, our experiments have taken us towards random viewing conditions, preferably uniformly sampled in space and angle (e.g., governed by uniformly spaced Poisson sampling), within the bounds of operation of a given system. Again, this is likely we are concerned here with low level visual intelligence and maximizing the number of looks on an object is akin to increasing our sampling in the underlying feature space. With respect to F3, our experiments have the best performance for mini-islands. That is, placement of objects, specifically duplicated, on different islands, which are alterations of background, e.g., rocks, sand, grass, etc. The last factor considered

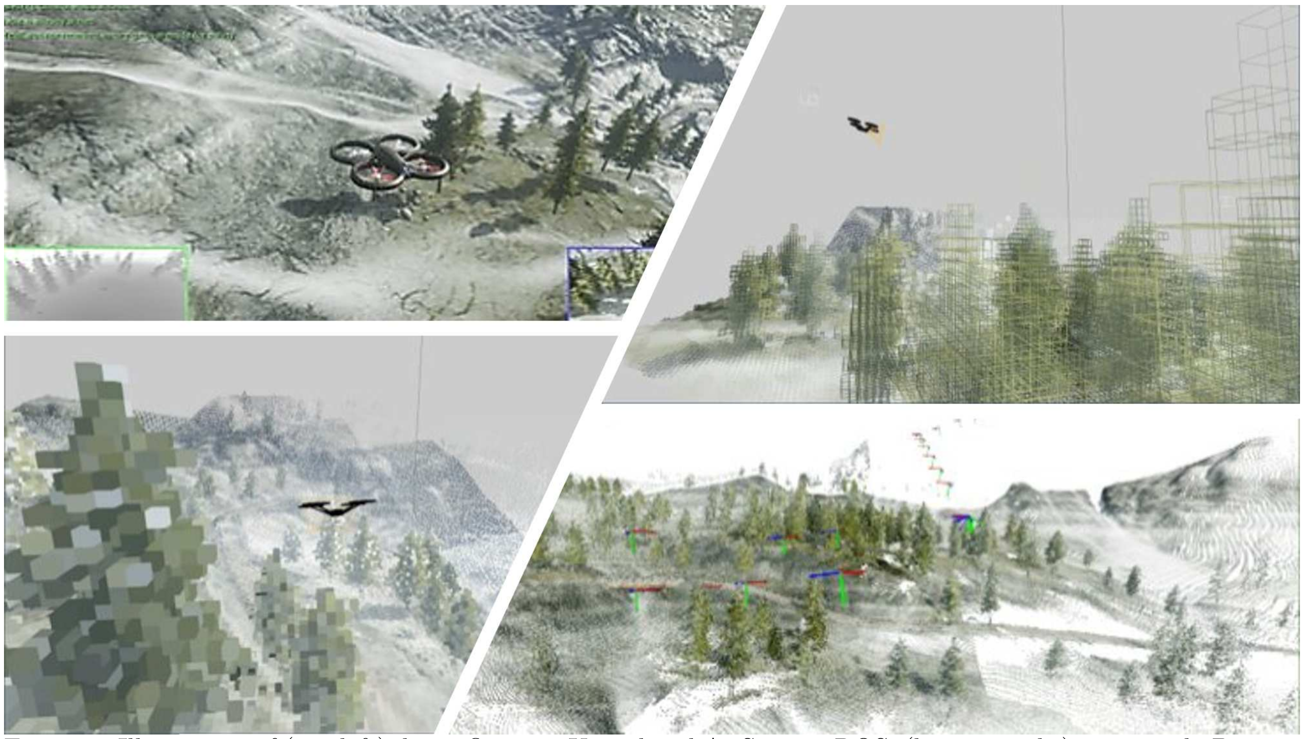


Figure 6: Illustration of (top left) drone flying in Unreal and AirSim via ROS, (bottom right) generated 3D point cloud, (upper right) voxelization, and (bottom right) resultant voxel downsampled internal representation.

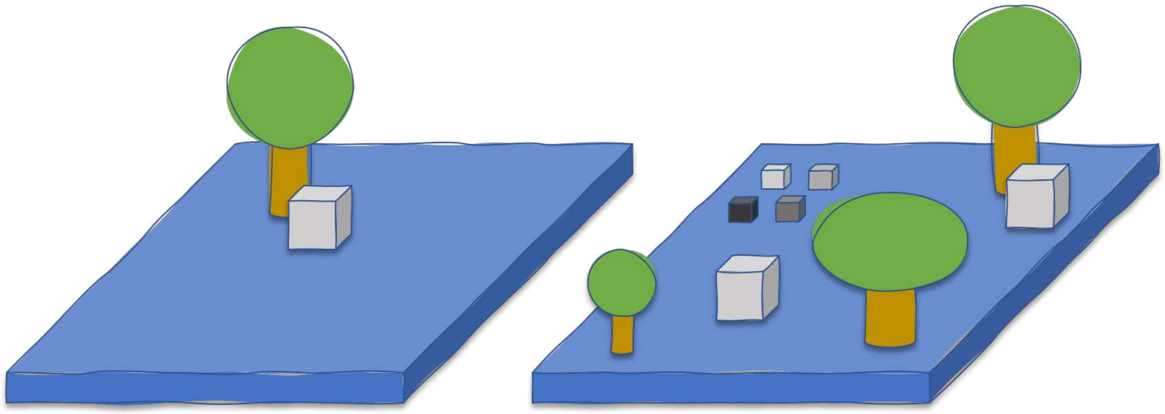


Figure 7: Illustration (left) of a scene with a target (the cube) and environment object (tree). Example (right) with randomly placed and scaled trees and targets.

here is F4. We run simulation multiple times, each time with different solar locations and angles (simulating different times of day), which change underlying visual effects like shadows, scene illumination, and etc. In the future we will expand our set to include factors like scattering (Unreal supports Mie and Rayleigh and etc.).

4. EXPERIMENTAL RESULTS

In this section, we outline a set of EHD experiments for low altitude UAVs. We intentionally do not discuss specific targets nor performance relative to metadata, e.g., what times of day, emplacement contexts, or environments. The goal is demonstration of relative performance so the reader can understand potential benefits of the data augmentation ideas expressed herein.

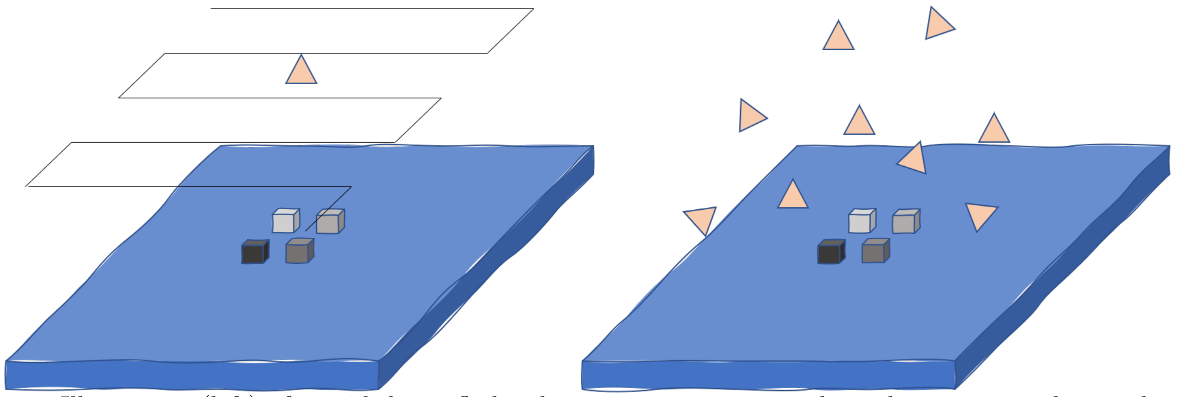


Figure 8: Illustration (left) of a grid drone flight observing a scene; triangle is the camera and view direction and cubes are targets. Example (right) showing randomly selected viewing positions.

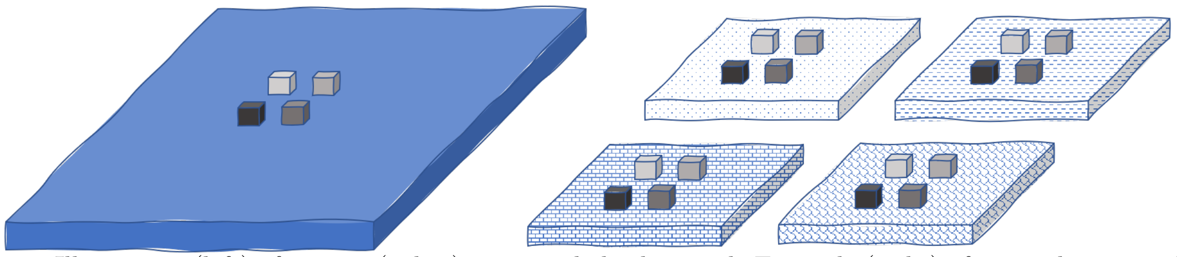


Figure 9: Illustration (left) of targets (cubes) on a single background. Example (right) of same objects on "mini islands" with different textures.

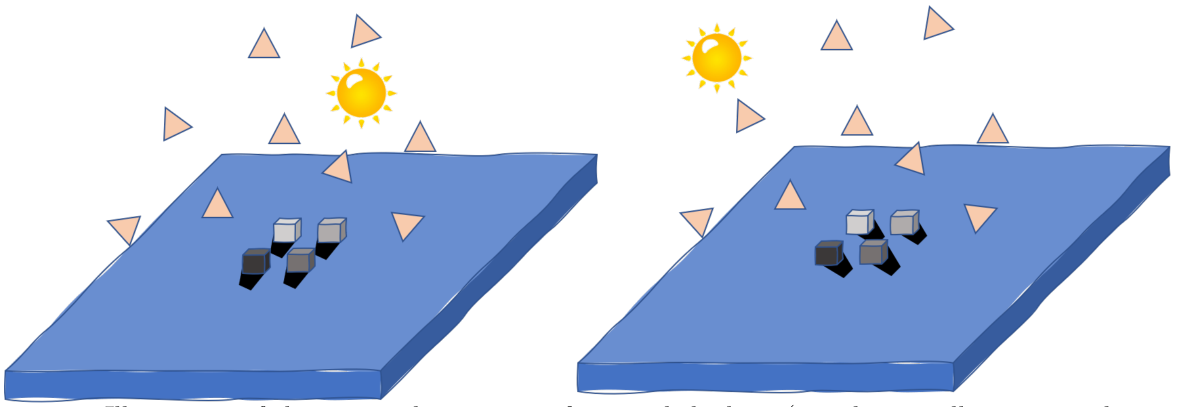
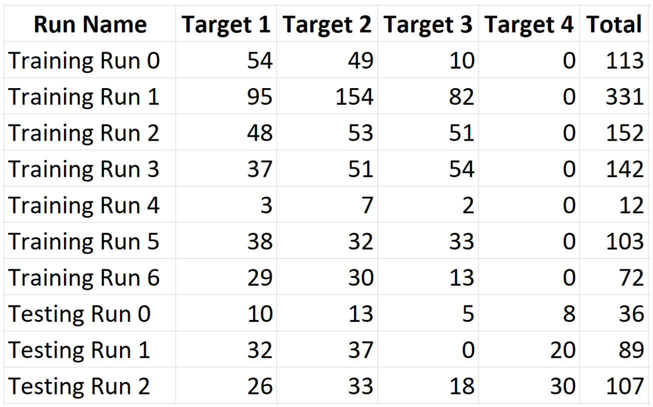


Figure 10: Illustration of changes in the position of sun and shadows (not showing illumination changes).



Run Name	Target 1	Target 2	Target 3	Target 4	Total
Training Run 0	54	49	10	0	113
Training Run 1	95	154	82	0	331
Training Run 2	48	53	51	0	152
Training Run 3	37	51	54	0	142
Training Run 4	3	7	2	0	12
Training Run 5	38	32	33	0	103
Training Run 6	29	30	13	0	72
Testing Run 0	10	13	5	8	36
Testing Run 1	32	37	0	20	89
Testing Run 2	26	33	18	30	107

Figure 11: Table of the number of annotations for each target type in each training and testing run.

4.1 Data Description

Experiments were preformed on images of EHs collected from an UAV. The data set consists of 10 flights, denoted as runs. Four of the runs were collected on one day, three a month later, and the remaining three were collected some months later. The most recent three runs were selected as hold-out testing runs to be used for cross-validation. Testing runs 1 and 2 are representative of the rest of the data. Run 0, however, is an intentionally difficult run. Images in the test set were collected in a subset of altitudes encountered in training. The camera was pointed nadir or near nadir for all runs. A variety of target types are found in the images. The targets vary meaningfully in size, shape, color, and composition. Targets are found in a variety of orientations and are sometimes approached from multiple angles. We choose to focus on four common EHs of interest. Target type 1 is a is a flat circular object. Target type 2 is a larger version of the target type 1. Thus, these two targets are simple and are primarly driven by shape. Target type 3 has a vertical cylindrical body. It is supported by 6 flat rectangular metal feet. The final target, target type 4 is is a small rectangular, slightly concave object. This target type is not found in any of the training runs. Target type 4 is also very difficult to detect in Run 2 due to lighting conditions and altitude. In total there are 925 training labels and 232 testing labels. The number of labels for each target type and in total can be found in Figure 11. It is important to take into account the number of labels provided per run when interpreting the results. Runs with very few labels of easy to detect targets will result in perfect receiver operating characteristic (ROC) curves, but will not give much insight into the actual performance of a model. All of the runs were collected during the day. Each run consists of a takeoff sequence, followed by an approach and search of a site containing EHs, then finally a return and landing sequence. In this article, we limit the scope of our analyis to grayscale vs RGB imagery. The resolution of all of the images is 640 x 512. We generated 3 simulated runs in the Unreal Engine with 240 images each. It should be noted, in simulation we found the best results for densely populated scenes on random islands with changes in environmental conditions (e.g., changes in solar position and therefore shadows). What this means is, even though we only use 240 simulated images, its variety is greater than what we encountered in our real training data. Likewise, we generated 7 AMA runs with 250 images each.

4.2 Results

This article is focused on data augmentation, not a specific EHD algorithm. As such, we choose to use the YOLOv5 object detector.[10] The point is to use a single algorithm and to evaluate performance relative to keeping that variable fixed. The YOLOv5 detector is an open source tool that directly predicts bounding box locations and sizes. Building on the original ”You Only Look Once” (YOLO) algorithm,[9] YOLOv5 has a number convenient features and improvements. At the cost of detection performance, we opted to use the small weights network configuration for a single class to speed up inference time as we are concerned with real-time performance on a UAV. We trained four sets of custom models. Each set consisted of 10 models, trained with identical parameters to assess the variability due to the stochastic nature of the learning algorithm. While YOLOv5 is capable of learning classification models, for these experiments we learn one model to detect all of the target types we are interested in. For all of the training sets, data was randomly split into 80% training and

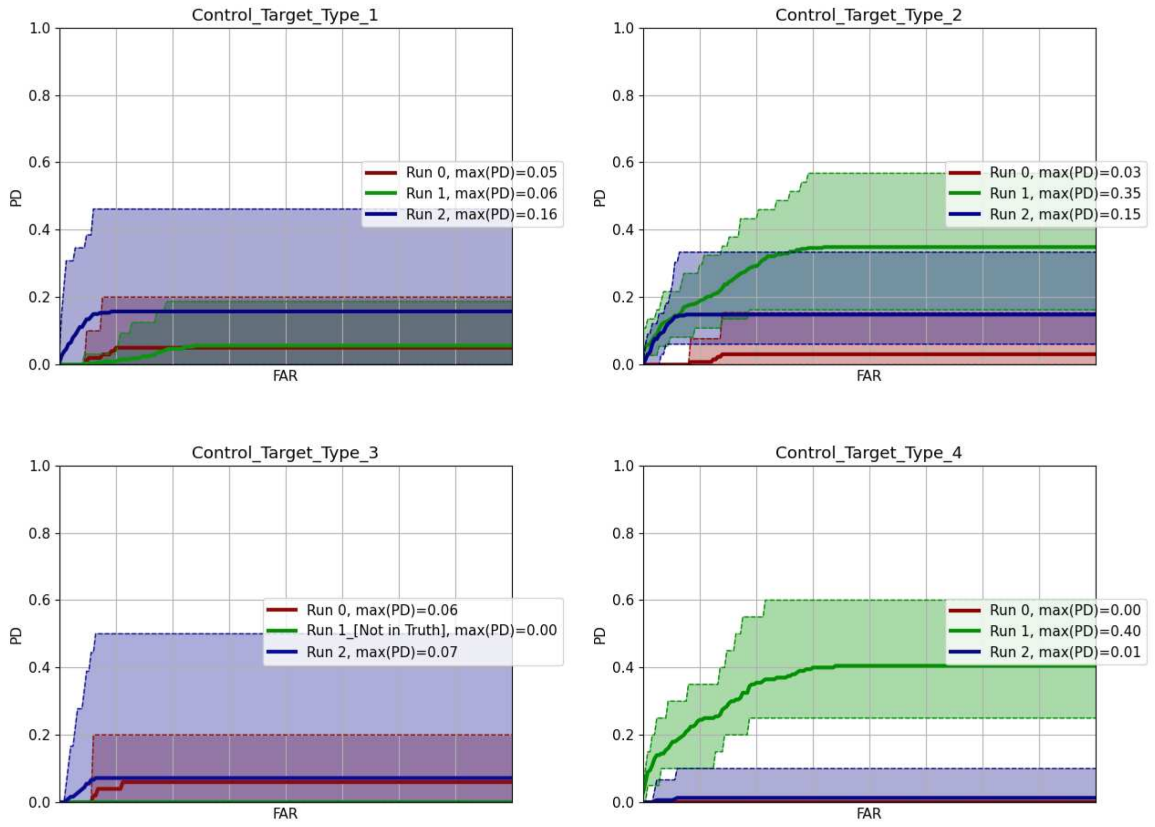


Figure 12: Receiver operator curve for the model trained only on real data.

20% validation. The first set of models were trained on all runs except the three hold-out testing runs. This model is intended to act as a control. The second set of models are trained on data generated by the AMA algorithm described in Section 2. For each of the training runs, 250 AMA images were generated with a variety of image modifications and simulated targets inserted. The third set of models were trained on images generated from three different simulated runs. There were 240 images generated per simulated run. The final set of models were trained on simply the union of the training sets for the other three models. This is a naive approach to combining models which does not take into account any sort of balancing and is just a first approach. More work could certainly be done to better setup the combined model. There are 1750 training images from AMA, 720 from simulation. Each image contains at least one labeled explosive hazard target. There are 925 real labels used in the control. The ROC curve in figs. 12 to 15 shows a measure of targets detected compared to false alarms for each of the target types. For each set of parameters, we train 10 models. To visualize the results, we shade the region from the minimum to the maximum detection results. The darker line represents the average detection results across the 10 models.

The results from adding simulated and augmented data to EHD models looks extremely promising. Through simulation (Figure 14) we are able to generate realistic imagery which can be used to train EHD models. Training a model from these images alone shows substantial improvement in detection performance over the control (Figure 12). While there is still a lot of room for improvement, we are encouraged to see that purely simulated imagery can be used to generate models which preform better than the relatively limited labeled real imagery. On that note, the simulated environments were abstract EH scenes. This means that three was very little effort spent to make the simulated objects, environments, UAV flight conditions, and camera model match the test conditions. We anticipate further gain if effort was exerted to make simulation match the real operating conditions.

Another advantage of using simulation is that we can now go through missed detections and tune our en-

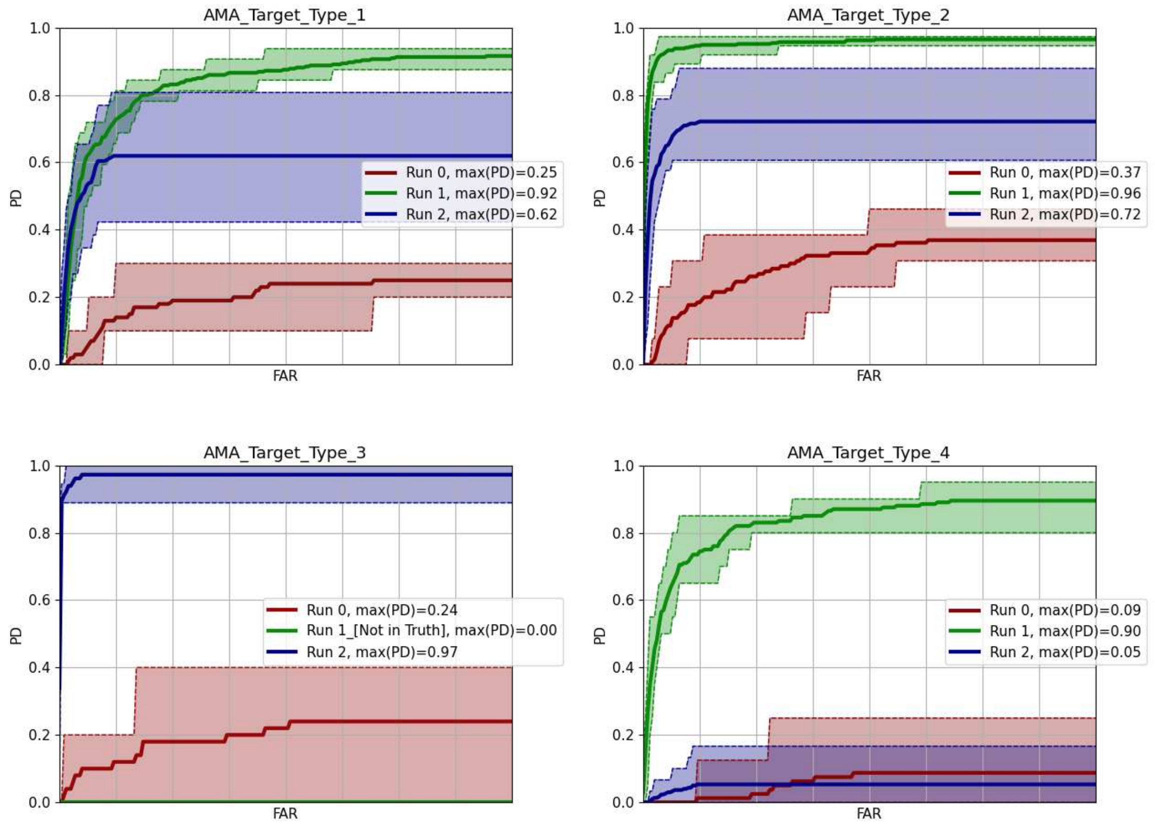


Figure 13: Receiver operator curve for the model trained only on AMA generated images.

vironments and targets to generate more robust models. Before simulation we would have requested certain collection parameters and have had to wait for the planning, collection, labeling, and transferring of the data to be complete before we could test and include new images in our models. Now we can simply modify our simulation, generate new images, and then train new and improved models. On a similar note, simulation can be used to identify contexts to aid future data collection. We do not assert that our Unreal Engine simulations are a replacement for real data. As data collection is a resource extensive process, any guidance or intuition is greatly welcomed. However, the experiments and results discussed in this article are preliminary and questions like these are subject of future work.

Next, the control models are not trained with any labels for target type 4 but they are still able to manage some detections. This is likely due to a combination of reasons. Target type 4 is relatively small and we run the YOLOv5 detector with a low detection threshold of 0.01 for scoring purposes. At low confidence levels, the model is hitting on anything with basic features that tend to make up actual targets. Features such as high contrast edges and corners, bright small anomalous patches, and certain shape segments trigger on nearly all true targets as well as many false alarms. There are some common features between targets so even without explicitly training on target type 4, at low confidence levels, features from the other target types is enough to generate some true positives.

One of the added benefits of generating simulated EH data is that it is easy to output the EH templates to be used with AMA. The results (Figure 13) show that we can use these templates, inserted into background images along with various corruptions to further improve detection performance. The models trained from AMA imagery alone appear better than the control and simulation alone. As mentioned in Section 4 we also took the union of the real training data set, the AMA generated images, and the simulated images to train and created a set of combined models. The combined models show some advantages over the others. Namely, we tend to see

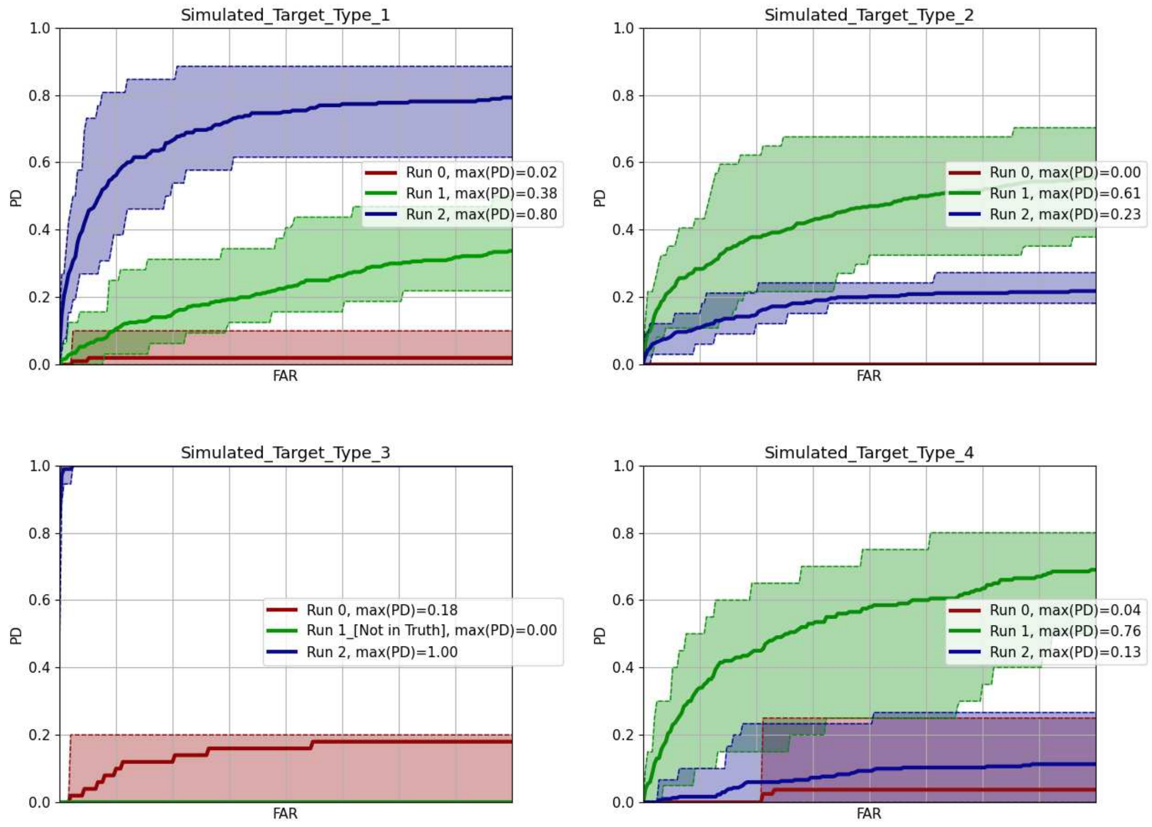


Figure 14: Receiver operator curve for the model trained only on simulated images.

that detections get pushed above false alarms. A more sophisticated approach for combining models or balancing data sets would almost certainly prove useful in this situation.

5. SUMMARY AND FUTURE WORK

In the current article we demonstrated two custom data augmentations to improve the quality of an UAV-based EHD algorithm for class imbalanced and data limited domains. The first approach relied entirely on simulated data. The second approach relied on simulated objects inserted into real UAV imagery that otherwise would likely go unused. Our preliminary results show that the combination of all techniques is best, followed by real data augmentation, then simulation, and real data last.

There are many ways to improve this paper. We used YOLOv5 as a baseline because our focus is data augmentation, not the underlying detection algorithm. In future work, we will tackle a number of avenues from tracking to multi-classifier and multi-sensor fusion,[15] smarter detection algorithms informed by shape,[16] more transparent and explainable algorithms,[17] generation and incorporation of 3D versus 2D data, smarter incorporation of metadata into the underlying detection process (beyond augmentation), and beyond.

In addition, targets are inserted with random rotations applied resulting in unrealistic shadow orientations. By analyzing the GPS and IMU data we could determine an orientation. Along with timestamps and cloud coverage information, we could determine the angle of the sun relative to the imagery and insert target templates with accurate shadow directions. Furthermore, target templates are inserted into an image at random positions. If more than one target template is inserted there is a chance of partial or even total occlusion. While adding some occlusion is likely desirable, the placements of targets directly on top of one another is not very realistic. Logic could be added to limit the maximum amount of occlusion. Similarly when targets are inserted into random

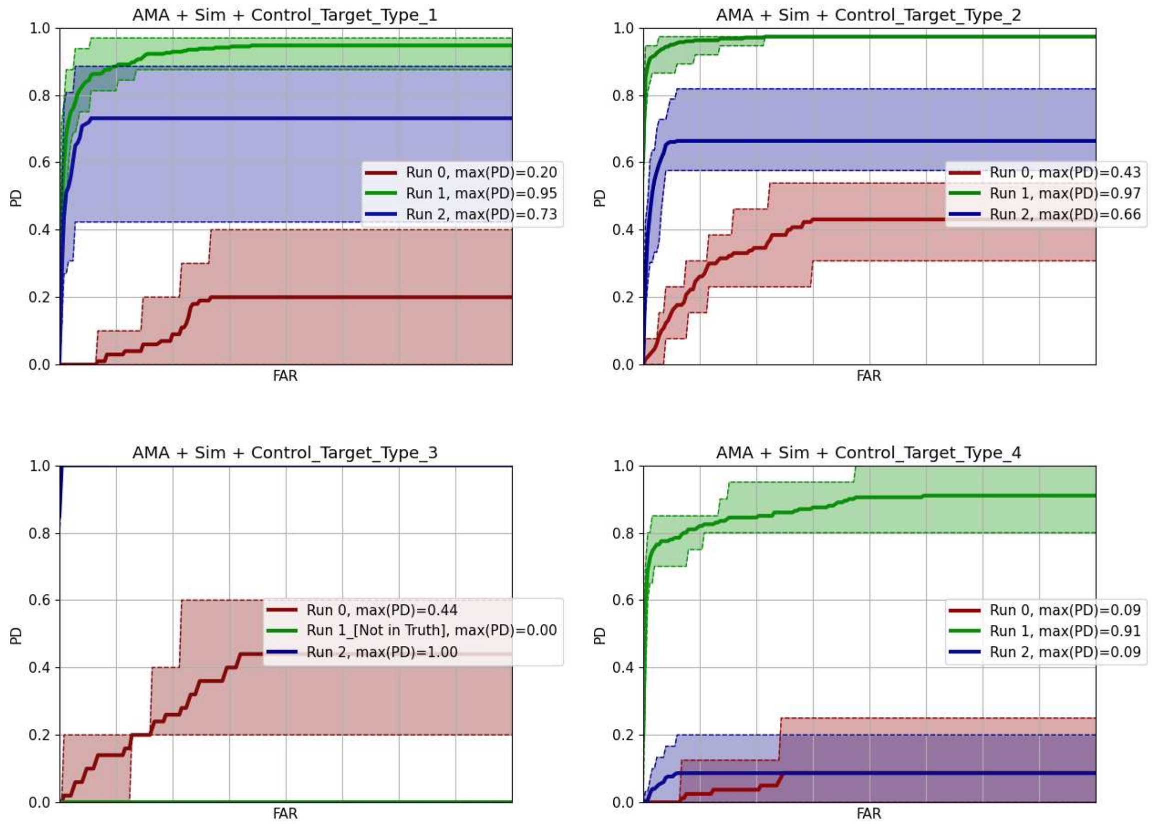


Figure 15: Receiver operator curve for the model trained on a combination of real, simulated and AMA generated images.

positions, there is no analysis of the structure of the image. Targets could be inserted right on top of trees, vehicles, or other structures unrealistically. This could be mitigated by segmenting each image first and then more intelligently inserting targets based on context. Examples include using a UNet for semantic segmentation in image space or generating 3D data from an algorithm like structure from motion (SfM) or multi view stereo (MVS). This would help us better generate and validate EHs out in the open (considered easier) vs partially obscured by man made and/or natural clutter.

With respect to simulation, we have a number of data augmentation research avenues to explore. As stated above, little effort was given to modeling realistic scenes. Further studies are needed to see if this is a benefit (does this encourage generalizability?) or drawback. There is also the question of a more rigorous analysis of which simulation "factors" (random islands, position of sun, etc.) contribute more than others. Another future research line is the aforementioned process of demonstrating an idea in simulation and then replicating it in the real world. The point being, simulation data helped our models, but there is great potential perhaps if simulation can inform real data collection. On a related note, we plan to explore the idea of a "self contained loop", in which the UAV moves in simulation looking for failures. When operating bounds are identified, that can be used to drive system specs, to inform data collection, or labeled data can be automatically generated in simulation and models can be retrained. Its an exciting idea to see if a machine can continue to self improve under such supervised conditions in simulation.

ACKNOWLEDGMENTS

Research funded by ARO grant number W911NF1810153 to support the U.S. Army DEVCOM C5ISR Center.

REFERENCES

[1] 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).

[2] 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).

[3] 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).

[4] "Unreal Engine." https://www.unrealengine.com/. (Accessed: 1 March 2021).

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

[6] "TurboSquid." https://www.turbosquid.com/. (Accessed: 1 March 2021).

[7] Pueyo, P., Cristofalo, E., Montijano, E., and Schwager, M., "Cinemairsim: A camera-realistic robotics simulator for cinematographic purposes," (2020).

[8] Ronneberger, O., Fischer, P., and Brox, T., "U-net: Convolutional networks for biomedical image segmentation," in [*Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*], Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., eds., 234–241, Springer International Publishing, Cham (2015).

[9] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You only look once: Unified, real-time object detection," in [*2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 779–788 (2016).

[10] 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).

[11] "AirSim." https://github.com/microsoft/AirSim. (Accessed: 1 March 2021).

[12] "Robot Operating System (ROS)." https://ros.org. (Accessed: 25 February 2021).

[13] Bondi, E., Dey, D., Kapoor, A., Piavis, J., Shah, S., Fang, F., Dilkina, B., Hannaford, R., Iyer, A., Joppa, L., et al., "Airsim-w: A simulation environment for wildlife conservation with uavs," in [*Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*], 40, ACM (2018).

[14] Buck, A., Deardorff, M., Anderson, D. T., Wilkin, T., Keller, J. M., Scott, G., III, R. H. L., and Camaioni, R., "Vader: A hardware and simulation platform for visuallyaware drone autonomy research," in [*SPIE*], (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 unmannedaerial system-based explosive hazard detection," in [*SPIE*], (2021).

[16] Islam, M. A., Murray, B., Buck, A., Anderson, D. T., Scott, G. J., Popescu, M., and Keller, J., "Extending the morphological hit-or-miss transform to deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems* , 1–13 (2020).

[17] Islam, M., Anderson, D. T., Pinar, A. J., Havens, T. C., Scott, G., and Keller, J. M., "Enabling explainable fusion in deep learning with fuzzy integral neural networks," *IEEE Transactions on Fuzzy Systems* **28**(7), 1291–1300 (2020).