REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
05-02-2024	Final Report	15-May-2018 - 14-May-2023

4. TITLE AND SUBTITLE
Final Report: Explainable Multi-Source Fusion in Deep Learning for Explosive Hazard Detection

5a. CONTRACT NUMBER
W911NF-18-1-0153

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
111111

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of Missouri - Columbia
115 Business Loop 70W
Mizzou North, Room 501
Columbia, MO 65211 -0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
73158-MI.14

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Derek Anderson
UU	UU	UU	UU		19b. TELEPHONE NUMBER 662-325-3530

RPPR Final Report
as of 07-Feb-2024

Agency Code: 21XD

Proposal Number: 73158MI **Agreement Number: W911NF-18-1-0153**
INVESTIGATOR(S):

Name: Derek Anderson
Email: andersondt@missouri.edu
Phone Number: 6623253530
Principal: Y

Organization: **University of Missouri - Columbia**
Address: 115 Business Loop 70W, Columbia, MO 652110001
Country: USA

DUNS Number: 153890272 EIN: 436003859
Report Date: 14-Aug-2023 Date Received: 05-Feb-2024
Final Report for Period Beginning 15-May-2018 and Ending 14-May-2023
Title: Explainable Multi-Source Fusion in Deep Learning for Explosive Hazard Detection
Begin Performance Period: 15-May-2018 **End Performance Period:** 14-May-2023
Report Term: 0-Other

Submitted By: Derek Anderson Email: andersondt@missouri.edu
Phone: (662) 325-3530

STEM Degrees: 1 **STEM Participants:** 7

Major Goals: Explosive hazards (EH) pose severe threats as remnants of past and current conflict zones, endangering the safety of both civilian and military personnel. The U.S. Army is actively engaged in the exploration of innovative methods for the detection and removal of these hazardous devices. Over the last few decades, our groups focus has encompassed various approaches, including hand-held and ground vehicle-based deployment, such as forward- and side-looking EH detection (EHD). Our previous efforts and objectives are detailed in the initial ARO proposal. Throughout the duration of this grant, we collaborated closely with the Army DEVCOM RTI, at Fort Belvoir, to delve into specific areas pertinent to a novel EH deployment platform -- using drones equipped with multi-sensor payloads. The list of topics from our initial proposal include:

- Detection & localization – Automatic identification of EH threats from aerial sensors.
- Scene understanding – Incorporation of platform meta-data and environmental meta-data for context to drive advanced EHD reasoning from an aerial platform.
- Multi-sensor fusion – Machine learning and data fusion in and across aerial sensors (RGB, infrared, multi-spectral, polarized cameras, etc.).
- Data augmentation – EHD is generally plagued by limited data and a need to operate in a range of environments. In return, we are focused on algorithms that can learn from class imbalanced small data sets and techniques to artificially create training data.
- Explainability – The above theories need to be understandable. Black box solutions cannot be trusted nor expected to generalize.

In the remainder of this report, we summarize our overall efforts. Specifically, we discuss progress in terms of open published academic research. If ARO would like additional details, they can be found in our biweekly presentations. Matt Aeillo and Clare Yang at RTI are our technical points of contact (TPOC). We work closely with RTI to discuss our theories and it is evaluated frequently (e.g., monthly) on real data furnished by the DoD. Furthermore, our results are independently evaluated by Institute for Defense Analysis (IDA). This process is vital as it helps us refine our basic theory relative to the ever evolving and adaptive EHD landscape.

Accomplishments: Overall, we accomplished the following:

- Created a research & development environment that resulted in a real-time prototype, which is used by DEVCOM at internal events and soldier touchpoints.
- Created and explored different real-time data-driven deep neural network-based approaches for localization, detection, and classification of above ground and buried EH that can be deployed on limited resource embedded

devices. This includes pre-screeners and second stage detectors for increased discrimination.
- Developed advanced simulation techniques to train, test, and evaluate our EHD methods. This includes partially synthetic data (synthetic EHs in real imagery) and full synthetic.
- Investigated explainable fusion for multi-sensor fusion.
- Developed graphical XAI to understand and improve our EHD algorithms and data.
- Developed linguistic XAI to understand and communicate EH results to end users (scientists, program managers, and end users).
- Created a formal language for scene (LSCENE) specification and data capture (LCAP).
- Used these languages to procedurally generate synthetic aerial datasets.
- Maintained constant contact (bi-weekly and off-week meetings) with stakeholders.
- Routinely delivered prototypes of algorithms, models, and tools for the Army to use, explore, showcase, and secure future pathways.
- Constant evaluation of our algorithms internally and externally with IDA.
- Routine code transfers to be used at demonstrations, evaluation, and feedback.
- Publications that facilitated knowledge transfer, student graduations, and in one case, a student (Matt Deardorff) who transitioned to becoming and employee with DEVCOM.

See our publications and report for more details

Training Opportunities: We did a number of "training" in this grant. This spans

conference publications and presentations

educating our students on the problem domain, both at Mizzou and trips and sponsor field activities

tech transfer and training of sponsor of our theories and tools

Results Dissemination: See the publications and biweekly reports to sponsor (DEVCOM RTI), which can be provided upon request, but with varying level of sensitivity.

Honors and Awards: Nothing to Report

Protocol Activity Status:

Technology Transfer: We transferred a number of things, documented in our report:

people (a student) to our sponsor (DEVCOM)

codes/libraries (for AI/ML but also simulation)

data sets

prototypes

reports

PARTICIPANTS:

Participant Type: PD/PI
Participant: Derek Anderson
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Brendan Alvey

Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Jeffrey Kerley
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Charlie Veal
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Co-Investigator
Participant: James Keller
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Co PD/PI
Participant: Grant Scott
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Undergraduate Student
Participant: Aaron Fuller
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: High School Student
Participant: Madeline Kovaleski
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Undergraduate Student
Participant: Peter Popescu
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:

RPPR Final Report
as of 07-Feb-2024

National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Matthew Deardorff
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

ARTICLES:

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: Sensors
Publication Identifier Type: DOI Publication Identifier: 10.3390/s23156879
Volume: 23 Issue: 15 First Page #: 6879
Date Submitted: 2/5/24 12:00AM Date Published: 8/1/23 5:00AM
Publication Location:
Article Title: Linguistic Explanations of Black Box Deep Learning Detectors on Simulated Aerial Drone Imagery
Authors: Brendan Alvey, Derek Anderson, James Keller, Andrew Buck
Keywords: artificial intelligence
Abstract: The modern era of machine learning is focused on data-driven solutions. While this has resulted in astonishing leaps in numerous applications, explainability has not witnessed the same growth. The reality is, most machine learning solutions are black boxes. Herein, we focus on data/information fusion in machine learning. Specifically, we explore four eXplainable Artificial Intelligence (XAI) questions relative to Choquet integral; (i) what is the quality of our inputs and their interactions, (ii) how is the information being combined, (iii) what is the quality of our training data (and thus our learned models), and (iv) what trust do we place in an output? Previously, we derived an initial set of indices for (i)-(iv) on the premise of perfect knowledge. Herein, we make XAI more accurate by taking into consideration what the machine learned. A combination of synthetic data and real-world experiments from remote sensing for fusing deep learners in the context of classification are explor
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: IEEE Transactions on Emerging Topics in Computational Intelligence
Publication Identifier Type: DOI Publication Identifier: 10.1109/TETCI.2020.3005682
Volume: 5 Issue: 4 First Page #: 520
Date Submitted: 2/5/24 12:00AM Date Published: 8/1/21 3:00PM
Publication Location:
Article Title: Explainable AI for the Choquet Integral
Authors: Bryce J. Murray, Muhammad Aminul Islam, Anthony J. Pinar, Derek T. Anderson, Grant J. Scott, Timoth
Keywords: XAI , explainable artificial intelligence, data fusion, information fusion, deep learning
Abstract: The modern era of machine learning is focused on data-driven solutions. While this has resulted in astonishing leaps in numerous applications, explainability has not witnessed the same growth. The reality is, most machine learning solutions are black boxes. Herein, we focus on data/information fusion in machine learning. Specifically, we explore four eXplainable Artificial Intelligence (XAI) questions relative to Choquet integral; (i) what is the quality of our inputs and their interactions, (ii) how is the information being combined, (iii) what is the quality of our training data (and thus our learned models), and (iv) what trust do we place in an output? Previously, we derived an initial set of indices for (i)-(iv) on the premise of perfect knowledge. Herein, we make XAI more accurate by taking into consideration what the machine learned. A combination of synthetic data and real-world experiments from remote sensing for fusing deep learners in the context of classification are explor
Distribution Statement: 3-Distribution authorized to U.S. Government Agencies and their contractors
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 07-Feb-2024

CONFERENCE PAPERS:

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV
Date Received: 19-Jan-2021 Conference Date: 27-Apr-2020 Date Published:
Conference Location: Online Only, United States
Paper Title: Doing more with less: similarity neural nets and metrics for small class imbalanced data sets
Authors: C. Veal, J. Schulz, A. Buck, D. T. Anderson, J. Keller, M. Popescu, G. Scott, D. Ho, T. Wilkin
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV
Date Received: 19-Jan-2021 Conference Date: 27-Apr-2020 Date Published:
Conference Location: Online Only, United States
Paper Title: Extending deep learning to new classes without retraining
Authors: J. Schulz, C. Veal, A. Buck, D. T. Anderson, J. Keller, M. Popescu, G. Scott, D. Ho, T. Wilkin
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications
Date Received: 05-Feb-2024 Conference Date: 30-Apr-2023 Date Published:
Conference Location: Orlando, United States
Paper Title: How should simulated data be collected for AI/ML and unmanned aerial vehicles?
Authors: Jeffrey Kerley, Derek Anderson, Brendan Alvey, Andrew Buck
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Geospatial Informatics XIII
Date Received: 05-Feb-2024 Conference Date: 30-Apr-2023 Date Published:
Conference Location: Orlando, United States
Paper Title: Simulated gold-standard for quantitative evaluation of monocular vision algorithms
Authors: Jack Akers, Andrew Buck, Raub Camaioni, Derek T. Anderson, Robert H. Luke III, James M. Keller, Ma
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XXIII
Date Received: 05-Feb-2024 Conference Date: 03-Apr-2022 Date Published:
Conference Location: Orlando, United States
Paper Title: Procedurally generated simulated datasets for aerial explosive hazard detection
Authors: Jeffrey Kerleya, Aaron Fullera, Madeline Kovaleski, Peter Popescu, Brendan Alvey, Derek T. Anderson,
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 07-Feb-2024

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XXIII
Date Received: 05-Feb-2024 Conference Date: 03-Apr-2022 Date Published:
Conference Location: Orlando, United States
Paper Title: Explosive hazard pre-screener based on simulated data with perfect annotation and imprecisely labeled real data
Authors: Madeline Kovaleskia, Aaron Fullera, Jeffrey Kerleya, Brendan Alveya, Peter Popescua, Derek T. Anders
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXVI
Date Received: 05-Feb-2024 Conference Date: 12-Apr-2021 Date Published:
Conference Location: Online Only, United States
Paper Title: Metadata enabled contextual sensor fusion for unmanned aerial system-based explosive hazard detection
Authors: Matthew Deardorf, Brendan Alvey, Derek T. Anderson, James M. Keller, Grant Scott, Dominic Ho, Andr
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2021 IEEE Symposium Series on Computational Intelligence (SSCI)
Date Received: 05-Feb-2024 Conference Date: 05-Dec-2021 Date Published:
Conference Location: Orlando, FL, USA
Paper Title: Characterization of Deep Learning-Based Aerial Explosive Hazard Detection using Simulated Data
Authors: Brendan Alvey, Derek T. Anderson, Clare Yang, Andrew Buck, James Keller, Ken Yasuda, Hollie Ryan
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)
Date Received: 05-Feb-2024 Conference Date: 11-Oct-2021 Date Published:
Conference Location: Montreal, BC, Canada
Paper Title: Simulated Photorealistic Deep Learning Framework and Workflows to Accelerate Computer Vision and Unmanned Aerial Vehicle Research
Authors: Brendan Alvey, Derek T. Anderson, Andrew Buck, Matthew Deardorff, Grant Scott, James M. Keller
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXVI
Date Received: 05-Feb-2024 Conference Date: 12-Apr-2021 Date Published:
Conference Location: Online Only, United States
Paper Title: Improving explosive hazard detection with simulated and augmented data for an unmanned aerial system
Authors: Jeffrey Kerley, Derek Anderson, Brendan Alvey, Andrew Buck
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
Date Received: 05-Feb-2024 Conference Date: 11-Jul-2021 Date Published:
Conference Location: Luxembourg, Luxembourg
Paper Title: Earth Mover's Distance as a Similarity Measure for Linear Order Statistics and Fuzzy Integrals
Authors: Matthew Deardorff, Derek T. Anderson, Timothy C. Havens, Bryce Murray, Siva K. Kakula, Timothy Wilk
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 07-Feb-2024

Partners
DEVCOM RTI/NVESD
belvoir, VA USA
4
data, experiments, feedback, domain knowledge

I certify that the information in the report is complete and accurate:
Signature: Derek Anderson
Signature Date: 2/5/24 11:14AM

Explainable Multi-Source Fusion in Deep Learning for Explosive Hazard Detection

Final Report for:

W911NF-18-1-0153

PI: Dr. Derek T. Anderson

Overview

Explosive hazards (EH) pose severe threats as remnants of past and current conflict zones, endangering the safety of both civilian and military personnel. The U.S. Army is actively engaged in the exploration of innovative methods for the detection and removal of these hazardous devices. Over the last few decades, our groups focus has encompassed various approaches, including hand-held and ground vehicle-based deployment, such as forward- and side-looking EH detection (EHD). Our previous efforts and objectives are detailed in the initial ARO proposal. Throughout the duration of this grant, we collaborated closely with the Army DEVCOM RTI, at Fort Belvoir, to delve into specific areas pertinent to a novel EH deployment platform -- using drones equipped with multi-sensor payloads. The list of topics from our initial proposal include:

- *Detection & localization* – Automatic identification of EH threats from aerial sensors.
- *Scene understanding* – Incorporation of platform meta-data and environmental meta-data for context to drive advanced EHD reasoning from an aerial platform.
- *Multi-sensor fusion* – Machine learning and data fusion in and across aerial sensors (RGB, infrared, multi-spectral, polarized cameras, etc.).
- *Data augmentation* – EHD is generally plagued by limited data and a need to operate in a range of environments. In return, we are focused on algorithms that can learn from class imbalanced small data sets and techniques to artificially create training data.
- *Explainability* – The above theories need to be understandable. Black box solutions cannot be trusted nor expected to generalize.

In the remainder of this report, we summarize our overall efforts. Specifically, we discuss progress in terms of open published academic research. If ARO would like additional details, they can be found in our biweekly presentations. Matt Aeillo and Clare Yang at RTI are our technical points of contact (TPOC). We work closely with RTI to discuss our theories and it is evaluated frequently (e.g., monthly) on real data furnished by the DoD. Furthermore, our results are independently evaluated by Institute for Defense Analysis (IDA). This process is vital as it helps us refine our basic theory relative to the ever evolving and adaptive EHD landscape.

Accomplishments

Overall, we accomplished the following:

- Created a research & development environment that resulted in a real-time prototype, which is used by DEVCOM at internal events and soldier touchpoints.
- Created and explored different real-time data-driven deep neural network-based approaches for localization, detection, and classification of above ground and buried EH that can be deployed on limited resource embedded devices. This includes pre-screeners and second stage detectors for increased discrimination.
- Developed advanced simulation techniques to train, test, and evaluate our EHD methods. This includes partially synthetic data (synthetic EHs in real imagery) and full synthetic.
- Investigated explainable fusion for multi-sensor fusion.

- Developed graphical XAI to understand and improve our EHD algorithms and data.
- Developed linguistic XAI to understand and communicate EH results to end users (scientists, program managers, and end users).
- Created a formal language for scene (LSCENE) specification and data capture (LCAP).
- Used these languages to procedurally generate synthetic aerial datasets.
- Maintained constant contact (bi-weekly and off-week meetings) with stakeholders.
- Routinely delivered prototypes of algorithms, models, and tools for the Army to use, explore, showcase, and secure future pathways.
- Constant evaluation of our algorithms internally and externally with IDA.
- Routine code transfers to be used at demonstrations, evaluation, and feedback.
- Publications that facilitated knowledge transfer, student graduations, and in one case, a student (Matt Deardorff) who transitioned to becoming and employee with DEVCOM.

The remainder of this report is organized as chapters, each one a publication.

Chapter 1

Title: Linguistic Explanations of Black Box Deep Learning Detectors on Simulated Aerial Drone Imagery

Venue: MDPI Sensors: Special Edition on Deep learning methods for Aerial Imagery

Date Published: August 3, 2023

sensors

MDPI

Article

Linguistic Explanations of Black Box Deep Learning Detectors on Simulated Aerial Drone Imagery

Brendan Alvey *, Derek Anderson, James Keller and Andrew Buck

Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, MO 65211, USA; andersondt@missouri.edu (D.A.); kellerj@missouri.edu (J.K.); buckar@missouri.edu (A.B.)
* Correspondence: bja3md@umsystem.edu

Abstract: Deep learning has become increasingly common in aerial imagery analysis. As its use continues to grow, it is crucial that we understand and can explain its behavior. One eXplainable AI (XAI) approach is to generate linguistic summarizations of data and/or models. However, the number of summaries can increase significantly with the number of data attributes, posing a challenge. Herein, we proposed a hierarchical approach for generating and evaluating linguistic statements of black box deep learning models. Our approach scores and ranks statements according to user-specified criteria. A systematic process was outlined for the evaluation of an object detector on a low altitude aerial drone. A deep learning model trained on real imagery was evaluated on a photorealistic simulated dataset with known ground truth across different contexts. The effectiveness and versatility of our approach was demonstrated by showing tailored linguistic summaries for different user types. Ultimately, this process is an efficient human-centric way of identifying successes, shortcomings, and biases in data and deep learning models.

Keywords: aggregation; black box; deep learning; evaluation; Explainable AI; fuzzy; linguistic; object detection; simulation

Citation: Alvey, B.; Anderson, D.; Keller, J.; Buck, A. Linguistic Explanations of Black Box Deep Learning Detectors on Simulated Aerial Drone Imagery. *Sensors* **2023**, *23*, 6879. https://doi.org/10.3390/s23156879

Academic Editor: Biswajeet Pradhan

Received: 15 June 2023
Revised: 21 July 2023
Accepted: 27 July 2023
Published: 3 August 2023

1. Introduction

A trained object detection model may perform well under circumstances that are similar to its training and validation data. However, current generation artificial intelligence (AI) and machine learning (ML) do not yet behave well in new and novel scenarios that deviate from training. Identifying the limitations and capabilities of a black-box AI/ML model is an important, timely, relevant, and challenging task. Our aim was two-fold. First, we desired a human-centric process for extracting tailored linguistic statements for eXplainable AI (XAI) and domain knowledge discovery. Second, we desired a human-free solution that leverages recent advancements in text/natural language-based generative AI and procedural simulation. The objective was a closed loop process for automatic model search and refinement. Figure 1 presents an overview of these two complementary yet competing goals.

Recent developments in large language models (LLMs) have led to groundbreaking new tools. LLMs are AI models trained on vast amounts of text data which output human-like text responses. Many of the highest performing models use extensions of the Transformer neural network architecture [1]. Although language models have existed for quite some time, the relatively recent drastic increase in scale, along with techniques for zero-shot learning and reinforcement learning from human feedback have resulted in unprecedented breakthroughs [2–4]. Modern LLMs have been shown to preform exceedingly well in a number of tasks including translation, writing essays, creating poetry, code generation, and summarization [5–9]. While the output from an LLM is far from perfect, it is sometimes preferred over human generated responses [10]. In the current article, we demonstrate a hybrid approach which combines more traditional machine generated

linguistic explanations of an object detector with an LLM to yield an improved user-friendly report. Although LLMs are being researched for understanding and explaining neural networks themselves [11], these methods are not yet mature enough to be trusted in many contexts. Often, LLMs can "hallucinate" poor responses [12] that are difficult to explain and result in reduced trust in a system. It is also not currently clear how exactly LLM provides explanations and, hence, whether they should be trusted. As a direct result, we opted for a more transparent and explainable approach that seeks to reduce the weaknesses associated with both approaches. Linguistic summarization is fully explainable but it results in a relatively complete set of explanations in a somewhat artificial language format. On the other hand, an LLM cannot yet compute all explanation tasks nor do we entirely trust the explanations it yields. Herein, linguistic summarization produces a set of user tailored explanations, and an LLM can reduce these explanations into a succinct natural language description.

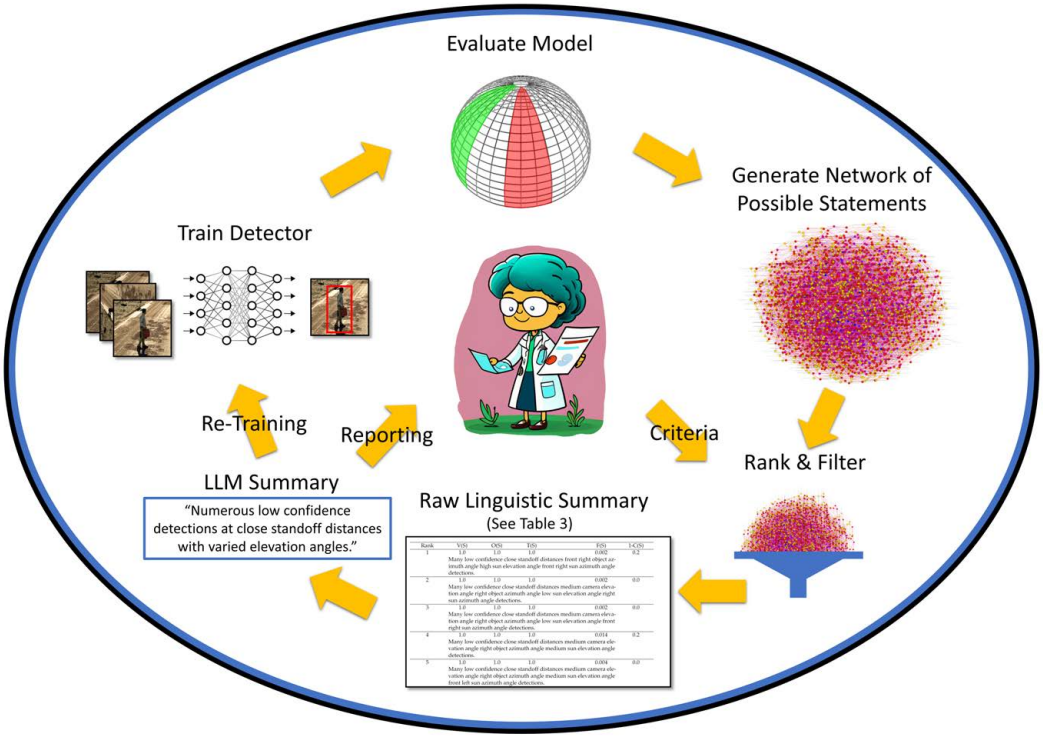


Figure 1. Overview of the process for generating and using linguistic summaries. First a detector is trained, then evaluated. The red box indicates an object declaration from a trained detector. A network of possible statements describing the model is generated and filtered based on user specific criteria. Linguistic qualities are computed and either a final report is given to the user or these statements are used to generate new data and train a new improved model.

LLMs can also be used in combination with other techniques to generate novel photo-realistic imagery from relatively simple, natural language text prompts [13,14]. Language models can be integrated with sophisticated rendering systems, such as the Unreal Engine, to create diverse photorealistic imagery with a simple set of instructions [15,16]. In this paper, we demonstrated an XAI system for generating textual descriptions of an object detection model. A future goal is to directly use the output from our system when tuned to focus on failure modes to generate novel imagery for training. This forms a closed loop learning system, capable of identifying shortcomings and addressing them to the best of its ability.

Many datasets, e.g., for unmanned aerial vehicles (UAV)/drones, are becoming larger in size and more detailed in terms of attributes. By recording camera, environmental, and target parameters associated with each image, we can automatically conduct a more detailed analysis of model performance than we could with only images and object locations.

As the number of data attributes increases, so too does the number of possible descriptions and ways of analyzing the data. Herein, we explored a new and novel process to automatically generate a summarized set of linguistic statements of a black box AI/ML model evaluated on data with many underlying data attributes. The objective was to produce a reduced and tailored set of target explanations surrounding successes, failures, and possible data and model biases. To demonstrate the proposed ideas, an AI/ML model that was trained on real-world data for person detection was evaluated on simulated photorealistic aerial data produced by the Unreal Engine. By using a virtual drone and the Movie Render Queue feature in Unreal Engine, we generated a simulated person dataset. An advantage of simulation is it has access to dense and accurate truth. Figure 2 shows an example scene with a person, a simulated drone, and an image that is captured from the low altitude aerial drone's point of view.

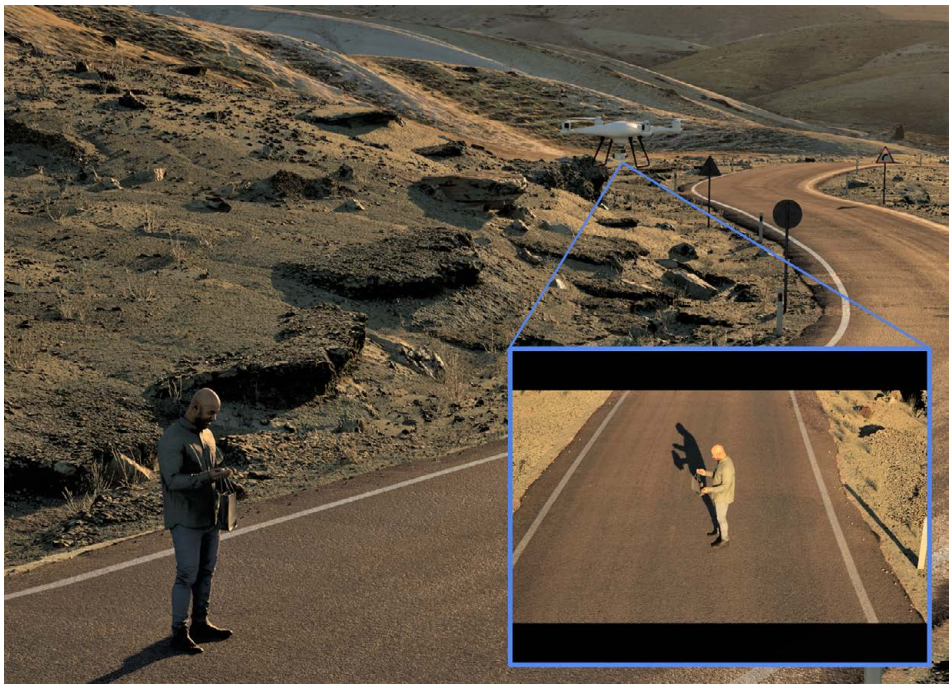


Figure 2. Example generated in Unreal Engine 5 of a virtual drone collecting imagery for a target object (person). This image illustrates one instance of collecting data in a hemisphere pattern at different relative poses, standoff distances, and environmental conditions (time of day, fog, etc.). The result is a virtual dataset (complete with ground truth) that is passed to a black-box deep learning object detector trained using real-world data. Finally, the detectors output is passed to our process for generating linguistic summaries of performance.

Linguistic summaries are a family of approaches which automatically create human-interpretable descriptions of datasets. They have been used to detect falls, improve the treatment of elders, forecast time series data, and in many other applications [17–24].

One of the largest challenges in effectively analyzing AI/ML model behavior has been simply obtaining the data. It is very expensive and time consuming to collect and annotate dense and accurate real-world aerial imagery. In many applications, there is a training and/or evaluation bottleneck. Thankfully, due to tools like the Unreal Engine, Unity [25], Apple's Hypersim [26], and NVIDIA's Isaac Gym [27], we can now quickly produce large and diverse sets of photorealistic imagery with dense and accurate truth. While these tools are not a perfect simulation or a replacement for the real-world, their level of realism is high and multiple works are already showing how they can be used as is to train and evaluate AI/ML algorithms [28–33]. In [34], we showed how this imagery can be used to systematically evaluate model performance. Through simulation, we are able to output very detailed descriptions for each image in our data set. However, this is initially

a double edged sword with respect to linguistic summaries. For each new data attribute (e.g., camera elevation angle, solar azimuth angle, altitude, etc.), the number of possible statements describing the data increases exponentially. It is important to note that, while simulation is a large focus of the proposed article, it is not a necessity. If the reader can collect truth in the real-world, that information can be used instead. We primarily focused on simulation because it is a controlled environment within which o study the proposed methods, and exploring its possible role in AI/ML is an open research question.

Let a linguistic statement, S, be a "Linguistic Protoform Summary" of the type "Q R Ys are P", where Q is a quantifier, R is a qualifier, Ys are data samples and P is a summarizer. A quantifier designates the amount of samples supported by the qualifier and summarizer. One attribute is reserved as the qualifier. It can be a performance metric such as the intersection over union (IOU) or confidence, combined with a fuzzy predicate, F_R," (e.g., low, medium, high). Summarizers are constructed from the power set of attributes and their fuzzy-predicates. Let $Y = \{y_1, y_2, \cdots, y_N\}$ be a dataset of N samples. Objects are represented as a set of attribute–value pairs. As mentioned above, one attribute is reserved as the qualifier while the other K attributes are used to construct the summarizer. An example attribute–value pair for an image would be ["altitude", 10 m"]. Each attribute, k, has a list of F_k fuzzy predicates associated with it. The value of the kth attribute for the nth data sample is denoted as $A_k(y_n)$. Each attribute and fuzzy-predicate pair has a fuzzy membership function associated with it, $\mu_{k,j}$. This function maps the value that corresponds to each attribute to a fuzzy membership value between 0 and 1. An example of such functions is shown in Figure 3. Use Table 1 as a reference for the notation used in this paper.

Table 1. List of notation used in this paper.

Notation	Description
Q	Quantifier (e.g., few, some, many)
R	Qualifier (fuzzy predicate + attribute)
y_n	nth data sample
P	Summarizer
A_k	kth data attribute (e.g., altitude)
$F_{k,j}$	jth fuzzy predicate for the kth data attribute
μ	fuzzy membership function
S	statement (Q R Ys are P)

The number of possible statements as a factor of the number of quantifiers, J_Q, and fuzzy-predicates is $J_Q J_R \prod_{k=1}^{K}(J_k + 1)$. This means moving from three quantifiers and three fuzzy-predicates for the qualifier and three other attributes to four quantifiers with four fuzzy-predicates for the qualifier and five other attributes is an increase from 576 to 50,000 possible statements. Making sense of the huge number of possible descriptions for a system is a major topic of this paper. We factor in linguistic qualities such as truth, focus, complexity, and a user-specified filter we call operational relevancy.

Expressing the state and behavior of a system using natural human language is the goal of linguistic summaries. Yager first introduced an approach to generating linguistic summaries for relational databases using the theory of fuzzy subsets [35]. Statements were created of the form, "Q objects in Y are P", where Q is known as a "Quantity in Agreement" such as "Few" or "Many". Y is the dataset being summarized, and P is a description such as "Altitude" or "Camera Elevation Angle". Fundamental to this approach is the use of fuzzy set theory. For each possible statement, Yager computed a "Truth" value, τ. This truth value is computed by first calculating the degree to which each sample satisfies each summarizer. That value is then passed through the fuzzy membership function for the

quantity in agreement of the statement. The "Truth" value is just one of many criteria we can consider when trying to analyze linguistic summaries. Often, we may prefer a few short statements that can efficiently communicate important information, rather than many lengthy, highly specific statements. The complexity of a statement can be measured by either counting the number of linguistic components, or simply counting the number of words or characters in a complete statement to determine its length.

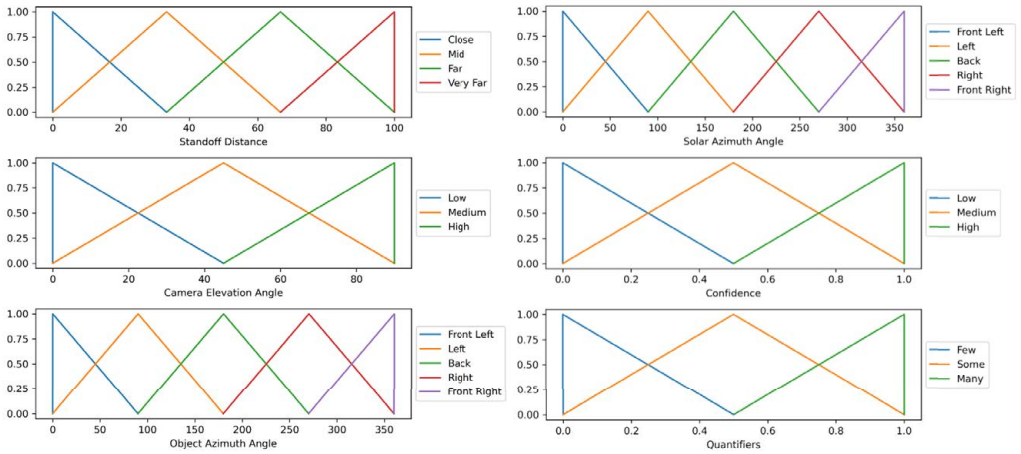


Figure 3. Plot of the membership functions used to calculate the quality measures and final ranking for each linguistic statement. Each attribute has a number of predicates. For each attribute + predicate pair, there is a membership function that can be tailored to match heuristic knowledge. Triangular functions were used to create a more decisive model. While we chose these functions for this paper, a user can choose whichever functions they desire.

Kacprzyk [36] proposed using five "quality" measures for filtering linguistic summaries—truth, imprecision, covering, appropriateness, and length. Imprecision is a value computed based on how vague or precise each statement is. For example, "On some of the days its somewhat hot" is not as precise as "On every day it is cold". Coverage is a measure of how much of a dataset is actually represented by a particular statement. A statement such as "High confidence detections at high altitudes" may happen to have a high truth measure but be poorly supported by few samples of data. If we use more than one criteria for filtering linguistic summaries, then we must also consider how to best aggregate them together. It is not a trivial task choosing which combination is best; there are many options [37]. One may use a pessimistic intersection like operator (t-norm) or an optimistic union like operator (t-conorm). An approach proposed by Popek and Katarzyniak is to use an interval-based aggregation [38]. Their method aims to build an internal representation of linguistic concepts to relate different statements. For this application, we chose to use a different approach, described in Section 2.3.

In summary, the current article was focused on generating tailored linguistic statements of black box AI/ML model performance. A simulation was used to build datasets with truth relative to object detection on a low altitude aerial drone. The following sections detail these steps. In Section 2, we outline the structure and process for extracting summaries. In Section 3, experiments and results are provided. Last, we summarize our findings and discuss future work.

2. Theoretical Background

In this section, we formally define our linguistic summary process. Let $Y = \{ y_1, y_2, \cdots, y_N \}$ be a dataset of N samples. Each sample in our dataset is an image that comes with associated metadata: a list of $K + 1$ key-value data attributes that describe the sample. (These attributes are either produced automatically as truth in simulation (dense and accurate) or they are approximated in the real-world (a gold standard).) Each of the different data attributes A_k also has associated with it a list of J_k number of

fuzzy-predicates. For all experiments in this paper, we used the data attributes and fuzzy predicates in agreement shown in Table 2. We assumed that an object detector has been trained and evaluated on a set of test samples. For each of the data attributes, we have a set of fuzzy membership functions, $\mu_{k,j}(A_k(y_n))$, which map the values associated with each of the K attributes to each of the fuzzy-predicates for each test sample.

Table 2. Data attribute and predicate pairs considered in this paper.

Data Attribute (A_k)	J_k	Fuzzy Predicates ($F_{k,j}$)
Altitude	4	Low, Medium, High, Very High
Camera Elevation Angle	3	Low, Medium, High
Camera Azimuth Angle	5	Back, Left, Front Left, Front, Right, Front Right
Solar Elevation Angle	3	Low, Medium, High
Solar Azimuth Angle	5	Back, Left, Front Left, Front, Right, Front Right
Confidence	3	Low, Medium, High

2.1. Generating Statements

Our linguistic summaries are functionally equivalent to the form, "Q R Ys are P", where Q is a quantifier, e.g., "Few", "Some", "Many", Y is a data set, R is a qualifier, e.g., "Low confidence", and P is a summarizer, e.g., "Low elevation angle, high altitude". You may use Figure 4 to serve as an examples statement and its parts. To compute metrics on each summary, we first developed a scheme to enumerate and keep track of our linguistic statements. Let each statement, S, be represented by a fixed length sequence of integers, $I(S)$. Each value in the sequence corresponds to a data attribute. If the value is 0, then that data attribute is not present in the current statement. Otherwise, the integer corresponds to an index in the list of fuzzy-predicates for that data attribute. For example, using the data attributes in Table 2, the sequence, [2, 1, 0, 0, 3, 0] corresponds to the linguistic statement, "Medium confidence, Low Altitude, and High Solar Elevation Angle images". With this structure, we are able to enumerate all of the possible statements in a compact list of integers. Importantly, we are able to represent complex (e.g., [3, 2, 1, 2, 3, 2] = "High confidence, Medium Altitude, Low Camera Elevation Angle, Left Camera Azimuth Angle, Medium Solar Elevation Angle, Left Solar Azimuth Angle images"), highly specific statements just as easily as compact, simpler statements (e.g., [1, 0, 0, 0, 0, 0]→"Low confidence images"). Likewise, we can keep track of the quantifier and qualifier fuzzy-predicate index by appending two more values to the array. Using this framework, all of the possible statements are encoded as a 2D array of integers. The combination of all possible index values and thus statements can be efficiently generated by computing the Cartesian product of the set of index values for each attribute.

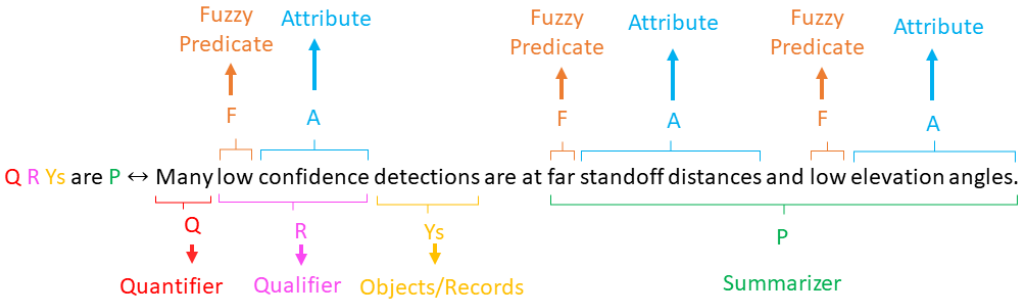


Figure 4. The parts breakdown of a linguistic summary of the form "Q R Ys are P". Q represents a quantifier. R represents a qualifier. Ys are the data samples in the data set. P is a summarizer. Summarizers and qualifiers are made up of attribute + fuzzy predicate pairs.

2.2. Linguistic Qualities

For each possible linguistic statement used to describe the performance of our detector, there are many different measures of quality we can compute. Depending on the use case, we may care more about statements which are highly accurate but are also relatively simple, or we may want statements that cover most of our data but are not very specific. Tuning the different criteria can help filter and guide results to be more useful. In this paper, we only considered four measures of quality: truth, focus, complexity, and operational relevance.

Probably the most popular measure is the "Truth Value". This value represents how accurate a statement is with respect to a given dataset. The truth value, $\tau(S)$, for each statement is computed according to Equation (1). Statements, which are composed of data attributes and fuzzy-predicate pairs along with a quantifier, can be thought of as composing a fuzzy query on the dataset. Each sample, n, has an aggregate membership value for a given fuzzy query, which is the minimum across the fuzzy membership values to each of the data attributes and fuzzy-predicates defined by the statement, S. We use $\mu_P(y_n)$ to denote the membership of the nth sample, to the summarizer, P,

$$\mu_P(y_n) = \min_{k,j \in I(s)} \mu_{k,j}(A_k(y_n)), \tag{1}$$

$$\tau(\text{Q R Ys are P}) = \mu_Q\left(\frac{\sum_{n=1}^{N} \mu_R(y_n) \wedge \mu_P(y_n)}{\sum_{n=1}^{N} \mu_P(y_n)}\right). \tag{2}$$

Another common measure is called "Focus". The focus value, sometimes called "coverage", measures how much of the dataset supports a given statement. This can be useful when considering statements which are represented by very few samples. For example, consider the statement, "Many high confidence detections at low elevation angles" and a dataset where only 1 of the 10,000 images has a high membership value to "Low Elevation Angle". If the image which has a high membership value to "Low Elevation Angle" also has a high membership value to "High Confidence", then that statement will achieve a high truth value. This creates a problem because the statement explicitly states "Many $\cdots$ detections $\cdots$" even though only one image was used to compute the high truth value. Instead, we can also consider the "Focus", weighing the fact that, in this example, the statement was only supported by a single image out of ten thousand. The focus value, $\mathcal{F}(S)$, for each statement is computed as,

$$\mathcal{F}(\text{Q R Ys are P}) = \frac{1}{N} \sum_{i=1}^{N} \mu_Q(\mu_P(y_i) \wedge \mu_R(y_i)). \tag{3}$$

As described above, the number of possible statements increases rapidly with the number of data attributes and quantities in agreement. In addition, the maximum length of a statement also increases, thought not as fast. Cognitive load refers to the amount of working memory required to perform a task [39]. Long statements composed of many attributes are likely to cause a higher cognitive load for a user to understand by requiring more working memory to process than shorter statements. Instead of reporting each and every place that the detector failed, we can summarize the performance in just one simple statement, "Many low confidence detections from the front". As Kacprzyk stated, through linguistic summaries we hope to provide "intuitively appealing and comprehensible results to the human user" [40]. As such, one goal of linguistic summaries is to provide simple, accurate statements that demand a minimum cognitive load on the reader to understand. To measure the complexity of a summary, $C(S)$, we chose to count the number of components via

$$C(\text{Q R Ys are P}) = \frac{\|I(S)\|_0}{K}, \tag{4}$$

where the numerator is the L0 norm, simply counting the number of attribute and fuzzy predicate pairs that are in a given statement. The denominator is the maximum number of attributes in a statement.

Finally, we considered a measure that functions mostly as a linguistic filter, which we call operational relevancy, $O(S)$. For all possible statements of the linguistic summaries, we can define a weight which represents how important that statement is to us. For example, if we are mostly concerned with the failure modes, then we may want to place a relatively high weight on "Many" and "Low Confidences" so that statements describing failures are valued higher. Likewise, we could tune our results to focus mostly on medium and high altitudes by placing weights of [0.2, 1.0, 0.8, 0.1] for altitudes of "Low", "Medium", "High", and "Very High", respectively. To compute the relevancy for a given summary,

$$w_P = \min_{k,j \in I(s)} w_{k,j}, \tag{5}$$

$$O(\text{Q R Ys are P}) = w_Q \wedge \frac{1}{N} \sum_{n=1}^{N} w_R \wedge w_P, \tag{6}$$

where $w_{k,j}$ is the relevancy weight for the jth fuzzy-predicate for the kth attribute.

2.3. Aggregating Linguistic Qualities

To combine the different measures described above, there are many options. Ultimately, it is up to the user to define which characteristics are most important to them. We proposed one method of combining measures, with some flexibility to balance the measures as needed. For each summary, S, we have four measures: Truth = $\tau(S)$, Focus = $F(S)$, Complexity = $C(S)$, and Operational Relevancy = $O(S)$. Before each of the measures we scale their values to be between 0 and 1 across all statements. We can then express relative simplicity as $1 - C(S)$. To combine the measures, we used

$$V(S) = \frac{O(S)\tau(S)[1 + w_F \mathcal{F}(S) + w_C(1 - C(S))]}{1 + w_F + w_C}, \tag{7}$$

and chose w_F and w_C to weight statements by how well supported by data or how simple the statements are. These variables tend to be highly correlated in terms of their outcome on resulting summaries. The statements are then ranked according to their final value, $V(s)$, ordered greatest to least. A threshold could be applied to remove summaries which do not meet a minimum value. This aggregation strategy allows us to only present relevant, true statements while taking into account how supported by data and how simple each statement is.

3. Experiments and Results

3.1. Experimental Design, Model, and Dataset

In the preceding sections, a systematic process was outlined to generate linguistic summaries. Now, in this section, this method was applied to summarize the performance of an object detection model (In prior work [41], we showed how an earlier version of our approach can be applied to a publicly available object detection model). This model was trained using real data and is being used to evaluate simulated data with known truth. The training data for this model consisted of a relatively large number of labeled domain specific images captured by a drone at various low altitudes, depicting people at different stand-off distances and off-nadir slant angles. Overall, the resulting model is enigmatic, functioning as a black box. It takes a new image as input and produces predictions in the form of axis-aligned bounding boxes (AABBs), box centers, and associated confidences for the locations of people in the image.

While our primary emphasis lay in a person detector, we also ventured into exploring the training, testing, and evaluation of more unique and infrequent objects like explosive

hazards (EH) [16,42–44] using both single and multiple modalities (RGB and infrared). It is essential to note that the specific object category, whether it is a person, EH, car, or other, is not the central focus of this article. Rather, our key objective was to showcase the process of gathering and evaluating an object-centric dataset acquired from a low altitude drone. To achieve this, we conducted simulations with different camera, platform, and environmental specifications. The drone equipped with a camera is maneuvered in a hemisphere pattern around the object, capturing data under diverse and varying conditions. This approach provides us with an ideal and controlled means to comprehensively study our AI/ML model. By manipulating and studying individual factors and their combinations, we can gain valuable insights into the model's performance. Alternatively, if the reader can obtain accurate platform metadata and environment ground truth, real-world data can be used instead of simulated data.

To evaluate model performance, we created a simulated person dataset. We leveraged the Unreal Engine 5.1 and used content from the Epic Marketplace to construct a realistic scene depicting a person in a desert environment. Through the movie render queue, high-quality photorealistic images were obtained, covering a diverse range of conditions. Specifically, four distinct character models were used: light-skinned male, light-skinned female, dark-skinned male, and dark-skinned female. For each character, we generated imagery from eleven different camera azimuth angles, three camera elevation angles, six solar azimuth angles, three solar elevation angles, and ten standoff distances. This setup allows us to capture a wide array of scenarios for evaluation.

Figure 5 shows example imagery from our simulated person dataset. While we chose a relatively empty scene in an arid environment, primarily due to free high-quality photometrically scanned assets, we acknowledge the potential for a more diverse and robust dataset with varied scene conditions in the future. We selected this dataset to limit our analysis to a few variables and to minimize confounding factors. Our objective was a controlled and tractable experiment that can be used to study our linguistic summaries. The reader can clearly add additional objects, introduce occlusion, and other factors relative to their intended goal and application.

Figure 6 shows detector results, showcasing examples of the most confident hits, false alarms, and missed targets. For a more comprehensive understanding, Figure 7 shows the distribution of detection performance on our simulated dataset relative to our different data attributes. This plot serves the dual purpose of helping us validate the linguistic summaries and it acts as a summary in itself.

Figure 7 provides context for the following linguistic statements. The reader can see that the model has a peak average performance somewhere around a standoff distance of 53 m to the object (human). Performance decreases as distance increases, which could be expected as a function of the number of pixels on our target. However, performance drops off faster as the object is closer (thus larger and more pixels) to the camera/drone. This model behavior is more likely a function of the attributes of the labeled data versus a core deficiency in the object detection model. That is, there are little to no observations of this object at close distances to the drone. Therefore, we can derive that the model is sensitive to object scale. Next, Figure 7 indicates performance variation with respect to the relative camera azimuth angle, e.g., we see the human from their front, side, back, etc. Our training dataset had different human poses, but this variables distribution was not uniformly distributed. As such, performance variation could be an indicator of the data or difficulty in recognizing people from different poses. Further analysis would need to be sought to determine the culprit more deeply. Furthermore, solar elevation and azimuth angle create different performances, which changes the overall image intensity and shadows, but its effect looks negligible. Last, and perhaps most dramatically, camera elevation angle has a big impact, with the model performing better at lower angles and dropping off at/near Nadir (overhead). Again, these graphical summarizations of our model's performance relative to our camera, platform, and environment attributes will help set the stage for understanding our linguistic statements.

Figure 5. Example renders for each of the different character models used in our simulated person dataset. Each model was placed in the same location in an arid landscape environment. Images were captured using the Movie Render Queue in Unreal Engine under a variety of different conditions. Each data attribute that we vary such as standoff distance, camera angle, and solar angle, are recorded for each sample. They are directly used to produce the linguistic summaries.

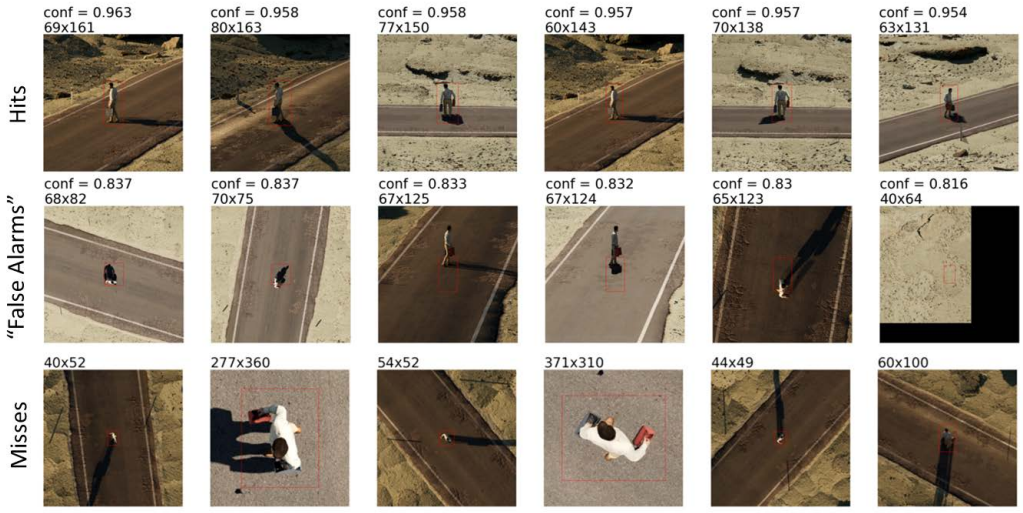


Figure 6. Plots visualizing the detection results of the object detector on the light-skinned male subset of our simulated person dataset. The confidence is given for the "Hits" and "False Alarms", and the bounding box size (width × height) is given for each declaration shown. (**Top**): Highest confidence true positive detections. (**Middle**): Highest confidence false alarms. Declarations over a target that are too large were scored as incorrect, as well as declarations that did not meet a minimum IOU (intersection over union) threshold. (**Bottom**): Examples of missed targets, randomly shuffled.

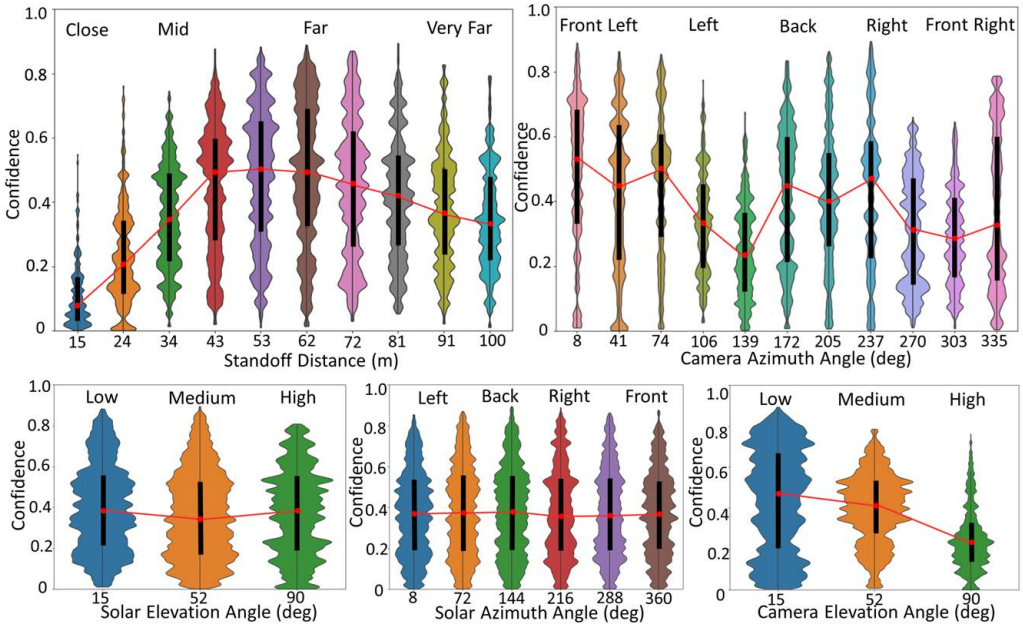


Figure 7. Plots visualizing the performance of the object detector as a factor of the different attributes. The width of each of the "violins" is determined by the number of occurrences of a particular attribute (x-axis) at a particular confidence. The thick black line in the middle of each unique data attribute value represents the interquartile range. The small red dot in the center of each thick black line represents the median. Although it is labeled only as "Confidence" for simplicity, the y-axis is actually IOU * confidence. These visualizations of performance are dictated by scored results, not just raw detector confidence. (**Top Left**): Standoff distance. (**Top Right**): Camera azimuth angle. (**Bottom Left**): Solar elevation angle. (**Bottom Middle**): Solar azimuth angle (**Bottom Right**): Camera elevation angle.

3.2. Tailored Summaries

To showcase the flexibility and adaptability of our approach in accommodating individual user preferences, we produced distinct sets of results tailored for two specific users. This demonstration exemplifies how summaries can be customized to meet the unique needs and criteria of different users. By presenting diverse summaries for each user, we underscored the versatility of our method in delivering relevant and personalized information to various stakeholders.

"User A" embodies the role of a research scientist, assigned with the crucial task of comprehending the existing model's limitations and devising strategies for enhancement. For this user, the primary emphasis lies in scrutinizing the failure cases characterized by numerous low confidence statements. They are entirely at ease with receiving lengthy, detailed statements ($w_C = 0$), provided they are precise and accurate. To optimize each improvement iteration, User A prefers to concentrate solely on the most valid statements, showing little concern for focus and complexity ($w_F = 0$ as well). Their objective is to extract the most relevant and truthful information from the summaries, enabling them to efficiently prioritize and address the model's weaknesses in subsequent iterations.

"User B" serves as a non-technical marketing designer, responsible for presenting the most favorable aspects of the current model to stakeholders. Their key focus lies in success cases distinguished by numerous high confidence statements. As a marketing professional, User B prefers simple and easily understandable statements (high w_C), prioritizing clarity and accessibility in the summaries. In this context, User B is willing to accept a trade-off where some true and operationally relevant statements may not make up a substantial portion of the dataset (low w_F). Their objective is to showcase the model's strengths in the most user-friendly manner, highlighting the positive outcomes and confidently conveying the model's capabilities to stakeholders.

3.3. Results

To offer in-depth insights into the evaluations tailored for each user, we present the results of running the linguistic summary separately for User A and User B in Tables 3 and 4, respectively. These tables provide a comprehensive breakdown of the linguistic summaries specifically generated for each user, precisely catering to their individual preferences and objectives. By presenting user-specific outcomes side by side, we demonstrate the customized nature of our approach, showcasing how linguistic summaries can be fine-tuned to suit the unique needs of different stakeholders.

Table 3. Linguistic summary for User A. This user is focused primarily on where the model fails (many low confidence). This user is mostly concerned with statements that are true and relevant, with $w_c = 0$ and $w_F = 0$. They are comfortable with long, detailed statements. This summary contains the 10 statements with the highest aggregate score.

Rank	V(S)	O(S)	T(S)	F(S)	$1 - C(S)$
1	1.0	1.0	1.0	0.002	0.2
	Many low confidence close standoff distances front right object azimuth angle high sun elevation angle front right sun azimuth angle detections.				
2	1.0	1.0	1.0	0.002	0.0
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle low sun elevation angle right sun azimuth angle detections.				
3	1.0	1.0	1.0	0.002	0.0
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle low sun elevation angle front right sun azimuth angle detections.				
4	1.0	1.0	1.0	0.014	0.2
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle medium sun elevation angle detections.				
5	1.0	1.0	1.0	0.004	0.0
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle medium sun elevation angle front left sun azimuth angle detections.				
6	1.0	1.0	1.0	0.005	0.0
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle medium sun elevation angle left sun azimuth angle detections.				
7	1.0	1.0	1.0	0.005	0.0
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle medium sun elevation angle back sun azimuth angle detections.				
8	1.0	1.0	1.0	0.005	0.0
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle medium sun elevation angle right sun azimuth angle detections.				
9	1.0	1.0	1.0	0.004	0.0
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle medium sun elevation angle front right sun azimuth angle detections.				
10	1.0	1.0	1.0	0.002	0.0
	Many low confidence close standoff distances medium camera elevation angle right object azimuth angle low sun elevation angle back sun azimuth angle detections.				

Table 4. Linguistic summary for User B. This user is focused primarily on where the model succeeds (many high confidence). This user is mostly concerned with simple statements that are true, relevant, and make up a significant portion of the data with $w_c = 1$ and $w_F = 1$. This summary contains the 10 statements with the highest aggregate score.

Rank	V(S)	O(S)	T(S)	F(S)	1 − C(S)
1	0.34	1.0	0.782	0.08	0.6
	Many high confidence far standoff distances low camera elevation angle detections.				
2	0.334	1.0	0.732	0.131	0.6
	Many high confidence far standoff distances medium camera elevation angle detections.				
3	0.312	1.0	0.756	0.029	0.6
	Many high confidence low camera elevation angle front left object azimuth angle detections.				
4	0.311	1.0	0.576	0.128	0.8
	Many high confidence low camera elevation angle detections.				
5	0.307	1.0	0.952	0.018	0.4
	Many high confidence far standoff distances low camera elevation angle front left object azimuth angle detections.				
6	0.299	1.0	0.522	0.203	0.8
	Many high confidence medium camera elevation angle detections.				
7	0.294	1.0	0.701	0.047	0.6
	Many high confidence medium camera elevation angle front left object azimuth angle detections.				
8	0.292	1.0	0.674	0.078	0.6
	Many high confidence medium camera elevation angle back object azimuth angle detections.				
9	0.291	1.0	0.67	0.079	0.6
	Many high confidence medium camera elevation angle left object azimuth angle detections.				
10	0.289	1.0	0.514	0.178	0.8
	Many high confidence far standoff distances detections.				

The summary for User A was tuned to focus on "Many low confidence · · · " statements. This user wanted statements that are accurate and relevant to them, even if they were very detailed and specific. As a result, the summary produces statements that on average have a simplicity value of only 0.1, but a very high minimum truth value of a perfect 1.0. By analyzing the top ranked statement, "Many low confidence close standoff distances front right object azimuth angle high sun elevation angle front right sun azimuth angle detections" against the violin plots and the membership functions in Figure 3, one can confirm that this is an accurate statement, given the information our summarizer had.

The summary for User B was tuned to focus on "Many high confidence · · · " statements. This user was interested in an accurate and relevant statements as well. However, this user very much preferred compact summaries supported by a relatively large proportion of the data. This resulted in an average simplicity score of 0.64. However, the minimum truth value was now 0.514 for the lowest ranked statement. The top ranked statement, "Many high confidence far standoff distances low camera elevation angle detections" is accurate. If we look back at the plots, we see that, at far standoff distances, the detector preforms relatively poorly at far standoff distances and there is a large amount of low confidence low camera elevation angle detections as well. Both summaries are valid but differ in important ways based on the user inputted criteria.

An intriguing perspective on the structure of the statements generated by our method is to envision the summary hierarchically. Beginning with the simplest statements and

progressively building up to more complex ones, a directed graph can be constructed to represent this hierarchy. It is important to recall that each summarizer is composed of attribute and fuzzy predicate pairs. In our constructed graph, edges connect nodes containing the current node's string as a sub-string, with only one additional attribute. For example, "Many high confidence detections" would be connected to "Many high confidence detections at medium standoff distances". This connection scheme ensures that the graph captures the relationships between statements as they become increasingly elaborate. Visualizing this topology in two dimensions is challenging due to the richness and intricacy of the linguistic summaries. As we delve deeper into the hierarchical tree, creating more complex statements, we discover new connections between branches that were not previously evident. This exploration encompasses all possible statements and connections based on their complexity, resulting in a complex and multi-dimensional structure. To help navigate and comprehend this linguistic summary graph, we utilized the Gephi tool, which offers valuable visualization capabilities [45]. See Figure 8 for a visualization of the linguistic summaries for User A. Nevertheless, we are actively exploring alternative methods to construct and visualize the graph in a manner that maximizes its utility and facilitates a better understanding of the summarization process.

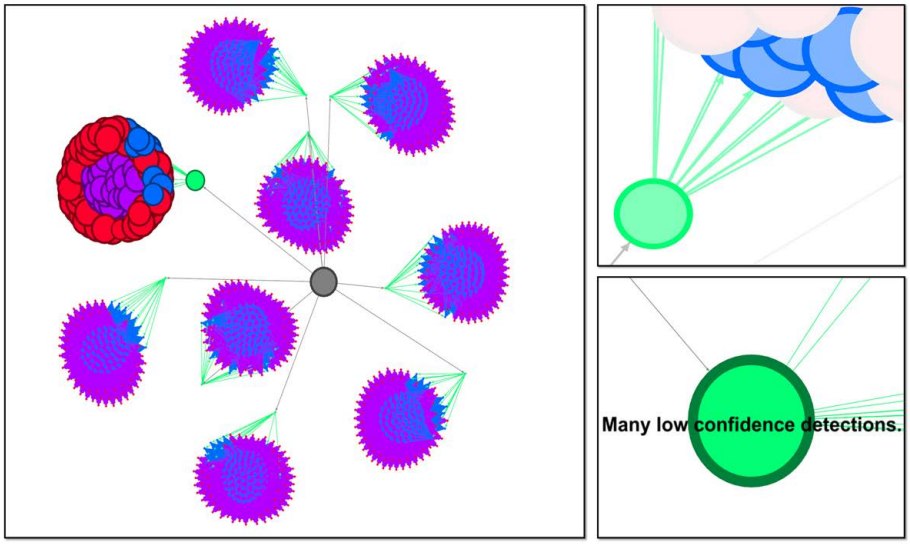


Figure 8. Graphic visualization of linguistic summaries for User A. Each node is a statement, sized by the final value assigned to that statement. Each edge is a directed connection from a simple statement to a more complex statement. The color of each node and edge is determined by the complexity of the statement. Each unique complexity value gets a random, unique color. (**Left**): Overview of all of the possible statements. Most of the statements have a zero value (no operational relevancy for User A, so they are set to the minimum node size. The "Many low confidence $\cdots$" branch is the only one with non zero values. (**Top Right**): Zoomed in view of a few nodes and their edges for the relevant branch of this summary. The green node is selected so only its immediately connected nodes are highlighted. (**Bottom Right**): Zoomed in view of a single node, its connections, and its corresponding statement.

3.4. LLM Summaries of Lingustic Protoform Summaries

In recent times, large language models (LLMs) like ChatGPT [46] have gained unparalleled popularity due to their versatility in various applications, including summarization. However, the linguistic summaries produced by our method might still contain redundant phrases. Combining several statements into a simpler yet equally accurate summary could be beneficial. An existing challenge with LLMs is their sensitivity to the exact "prompt" or input text, as evident in studies such as [47,48]. As a quick demonstration of leveraging an LLM to enhance summaries, we fed our statements into ChatGPT using GPT-4 as the base model. Below are the results obtained via the prompt: "The following statements were

produced by a linguistic summarizer of an aerial object detection model. Can you please analyze them and reduce any redundancy to produce a more succinct report? Can you summarize it in one sentence while maintaining as much of the content as you can?"

> User A: *The aerial object detection model identified numerous low confidence detections at close standoff distances, with varied camera elevations, object azimuths, and sun angles, primarily focusing on medium camera elevation and right object azimuth scenarios.*
>
> User B: *The aerial object detection model demonstrates high confidence in detecting objects at far standoff distances with low and medium camera elevation angles, as well as various object azimuth angles, including front left, back, and left.*

This simple experiment merely scratches the surface of the vast possibilities that arise from harnessing LLMs for model summarization. The summaries produced by ChatGPT are remarkably accurate and impressive, especially considering the relatively unstructured input data of natural language followed by machine-produced results. Imagine the potential if we were to run an LLM like ChatGPT offline, enabling us to automate the generation process using highly tailored prompts that incorporate a broader range of input data beyond the ten statements provided here. This approach presents an appealing prospect as it combines the strengths of principled approaches, yielding true and relevant statements, with the added benefit of simple, well-articulated, and user-friendly results, making it a compelling choice for effective summarization.

4. Conclusions and Future Work

In conclusion, this article presented a novel Explainable AI (XAI) approach designed to generate linguistic summarizations of black box models, offering tailored explanations to suit diverse user needs. We applied this approach in the context of explaining an object detector model, e.g., automatic target recognition (ATR), utilized in a low-altitude aerial drone setting. Two very different types of users were demonstrated and example summaries were provided. Moreover, we delved into the potential of large language models (LLM) to further refine and simplify our summaries for human consumption. To demonstrate the efficacy, capabilities, and flexibility of our approach, we conducted an experiment using simulation with known ground truth. Through this experiment, we successfully showcased how our methodology empowers users to gain insights into the inner workings of an AI/ML model and its decision-making process. By producing comprehensive and accountable user-specific linguistic summaries, we have taken a step towards bridging the gap between complex black box models and human interpretable explanations.

With respect to future work, we are keen on further investigating the impact of the underlying evaluation dataset's distribution on our generated linguistic summaries. It is crucial to approach the dataset selection meticulously, ensuring that the attributes are uniformly distributed throughout the dataset to prevent any artificial biases in the summaries. To address this challenge, we propose two potential solutions. Firstly, one can thoughtfully curate the evaluation dataset, as we have done in our study, ensuring that it encompasses a well-balanced representation of attributes to foster unbiased summaries. Alternatively, we can adapt our technique to handle knowledge gaps or biases that may exist in the distribution of the evaluation dataset. This adaptation would enable us to produce reliable and accurate summaries even under varying dataset conditions.

Moreover, we are interested in exploring alternative aggregation operators and linguistic qualities in pursuit of user multi criteria decision making. By expanding our range of options, we can exercise greater control over the characteristics of the resulting summaries, tailoring them to specific user requirements more effectively.

The graph-based hierarchical structure and the corresponding visualizations have proven invaluable in understanding the data in ways that text summaries alone cannot achieve. These visualizations offer a more intuitive and comprehensive grasp of the summarization process, providing valuable insights into the complex relationships between

statements, which may otherwise remain obscured in textual form. Future work looking into interactive processes or combined graphical and text explanations is important.

A highly promising use case that our method opens up for exploration is its integration into a closed-loop machine learning and data generation system. By feeding our linguistic summaries into procedural generation tools, we can generate photorealistic imagery, particularly in scenarios where our model's performance is subpar. This approach allows us to delve into the inherent difficulty of certain scenarios and gain a deeper understanding of their impact on object detection. For instance, scenarios with less contrast between the target and the background are expected to pose increased challenges in detection, irrespective of the training data. By exploring such scenarios, we can shed light on the underlying factors affecting our model's performance and develop strategies to overcome these challenges. Our ultimate goal is to develop an automated system capable of achieving exceptional object detection proficiency across diverse scenarios. This system would be self-improving, continuously learning and adapting to changing operating conditions. By combining our linguistic summarization approach with procedural data generation and machine learning, we aim to create an intelligent and agile system that can dynamically enhance its capabilities, staying at the forefront of object detection performance.

Last, we employed simulation to acquire imagery and precise metadata. While our process and results hold valid within this context, we recognize the significance of investigating the impact of metadata errors on the summary generation process. Ensuring robustness in the presence of potential inaccuracies in metadata is an important avenue for further exploration. While our use of fuzzy set theory naturally absorbs some level of error, a precise study and possible extensions might be warranted. Additionally, to maintain accuracy and minimize confounding factors, we deliberately restricted the complexity of our environment. This approach allowed us to focus on producing accurate summaries without overwhelming complexities. However, for future work, it will be essential to broaden our scope and include a more diverse set of variables. By introducing more complex scenes, we can gain a deeper understanding of how our method performs under varied and challenging conditions. This expansion will enable us to assess the generalizability and adaptability of our approach across a wider range of scenarios and applications.

Author Contributions: Conceptualization, All authors; methodology, B.A., D.A. and J.K.; software, B.A.; validation, B.A. and D.A.; writing—original draft preparation, B.A.; writing—review and editing, All authors; visualization, B.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.U.; Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Granada, Spain, 2017; Volume 30.
2. Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T.B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; Amodei, D. Scaling Laws for Neural Language Models. *arXiv* **2020**, arXiv:2001.08361.
3. Larochelle, H.; Erhan, D.; Bengio, Y. Zero-Data Learning of New Tasks. In Proceedings of the 23rd National Conference on Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008; Volume 2, pp. 646–651.
4. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training language models to follow instructions with human feedback. *arXiv* **2022**, arXiv:2203.02155.
5. Zhu, W.; Liu, H.; Dong, Q.; Xu, J.; Huang, S.; Kong, L.; Chen, J.; Li, L. Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis. *arXiv* **2023**, arXiv:2304.04675.

6. Franceschelli, G.; Musolesi, M. On the Creativity of Large Language Models. *arXiv* **2023**, arXiv:2304.00008.
7. Li, R.; Allal, L.B.; Zi, Y.; Muennighoff, N.; Kocetkov, D.; Mou, C.; Marone, M.; Akiki, C.; Li, J.; Chim, J.; et al. StarCoder: May the source be with you! *arXiv* **2023**, arXiv:2305.06161.
8. Zhang, T.; Ladhak, F.; Durmus, E.; Liang, P.; McKeown, K.; Hashimoto, T.B. Benchmarking Large Language Models for News Summarization. *arXiv* **2023**, arXiv:2301.13848.
9. Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Zhu, K.; Chen, H.; Yang, L.; Yi, X.; Wang, C.; Wang, Y.; et al. A Survey on Evaluation of Large Language Models. *arXiv* **2023**, arXiv:2307.03109.
10. Liu, Y.; Fabbri, A.R.; Liu, P.; Radev, D.; Cohan, A. On Learning to Summarize with Large Language Models as References. *arXiv* **2023**, arXiv:2305.14239.
11. Bills, S.; Cammarata, N.; Mossing, D.; Tillman, H.; Gao, L.; Goh, G.; Sutskever, I.; Leike, J.; Wu, J.; Saunders, W. Language Models Can Explain Neurons in Language Models. 2023. Available online: https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html (accessed on 3 May 2023).
12. Mündler, N.; He, J.; Jenko, S.; Vechev, M. Self-contradictory Hallucinations of Large Language Models: Evaluation, Detection and Mitigation. *arXiv* **2023**, arXiv:2305.15852.
13. Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; Sutskever, I. Zero-Shot Text-to-Image Generation. In Proceedings of the 38th International Conference on Machine Learning, Virtual Event, 18–24 July 2021; Meila, M., Zhang, T., Eds.; PMLR: Cambridge, MA, USA, 2021; Volume 139, pp. 8821–8831.
14. Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; Ommer, B. High-Resolution Image Synthesis with Latent Diffusion Models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–20 June 2022.
15. Epic Games. *Unreal Engine*; Epic Games: Cary, NC, USA, 25 April 2019. Available online: https://www.unrealengine.com (accessed on 14 June 2023).
16. Kerley, J.; Fuller, A.; Kovaleski, M.; Popescu, P.; Alvey, B.; Anderson, D.T.; Buck, A.; Keller, J.M.; Scott, G.; Yang, C.; et al. Procedurally generated simulated datasets for aerial explosive hazard detection. In *Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XXIII*; Guicheteau, J.A., Howle, C.R., Eds.; International Society for Optics and Photonics, SPIE: San Diego, CA, USA, 2022; Volume 12116, p. 1211611. [CrossRef]
17. Wilbik, A.; Keller, J.M.; Alexander, G.L. Linguistic summarization of sensor data for eldercare. In Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics, Anchorage, AK, USA, 9–12 October 2011; pp. 2595–2599. [CrossRef]
18. Anderson, D.T.; Luke, R.H.; Keller, J.M.; Skubic, M.; Rantz, M.J.; Aud, M. Linguistic summarization of video for fall detection using voxel person and fuzzy logic. *Comput. Vis. Image Underst.* **2009**, *113*, 80–89.
19. Kaczmarek-Majer, K.; Hryniewicz, O. Application of linguistic summarization methods in time series forecasting. *Inf. Sci.* **2019**, *478*, 580–594. [CrossRef]
20. Jain, A.; Popescu, M.; Keller, J.; Rantz, M.; Markway, B. Linguistic summarization of in-home sensor data. *J. Biomed. Inform.* **2019**, *96*, 103240. [CrossRef] [PubMed]
21. Jain, A.; Jiang, T.; Keller, J.M. Impact of the Shape of Membership Functions on the Truth Values of Linguistic Protoform Summaries. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*; Carvalho, J.P., Lesot, M.J., Kaymak, U., Vieira, S., Bouchon-Meunier, B., Yager, R.R., Eds.; Springer: Cham, Switzerland, 2016; pp. 204–213.
22. Jain, A.; Keller, J.M. On the computation of semantically ordered truth values of linguistic protoform summaries. In Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Istanbul, Turkey, 2–5 August 2015; pp. 1–8. [CrossRef]
23. Jain, A.; Keller, J.M. Textual summarization of events leading to health alerts. *Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.* **2015**, *2015*, 7634–7637. [CrossRef] [PubMed]
24. Jain, A.; Keller, J.M.; Bezdek, J.C. Quantitative and qualitative comparison of periodic sensor data. In Proceedings of the 2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), Las Vegas, NV, USA, 24–27 February 2016; pp. 37–40. [CrossRef]
25. Juliani, A.; Berges, V.P.; Vckay, E.; Gao, Y.; Henry, H.; Mattar, M.; Lange, D. Unity: A General Platform for Intelligent Agents. *arXiv* **2018**, arXiv:1809.02627.
26. Roberts, M.; Ramapuram, J.; Ranjan, A.; Kumar, A.; Bautista, M.A.; Paczan, N.; Webb, R.; Susskind, J.M. *Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding*; ICCV: Vienna, Austria, 2021.
27. Liang, J.; Makoviychuk, V.; Handa, A.; Chentanez, N.; Macklin, M.; Fox, D. *GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning*; PMLR: Cambridge, MA, USA, 2018.
28. Truong, J.; Rudolph, M.; Yokoyama, N.H.; Chernova, S.; Batra, D.; Rai, A. Rethinking Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation. In Proceedings of the 6th Annual Conference on Robot Learning, Auckland, New Zealand, 14–18 December 2022.
29. Chebotar, Y.; Handa, A.; Makoviychuk, V.; Macklin, M.; Issac, J.; Ratliff, N.; Fox, D. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8973–8979. [CrossRef]
30. Chen, X.; Hu, J.; Jin, C.; Li, L.; Wang, L. Understanding Domain Randomization for Sim-to-real Transfer. *arXiv* **2022**, arXiv:2110.03239.

31. Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. Learning from Simulated and Unsupervised Images through Adversarial Training. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2242–2251. [CrossRef]
32. Akers, J.; Buck, A.; Camaioni, R.; Anderson, D.; Luke, R.; Keller, J.; Deardorff, M.; Alvey, B. *Simulated Gold Standard for Quantitative Evaluation of Monocular Vision Algorithms*; SPIE: San Diego, CA, USA, 2023.
33. Georgakis, G.; Mousavian, A.; Berg, A.C.; Kosecka, J. Synthesizing Training Data for Object Detection in Indoor Scenes. *arXiv* **2017**, arXiv:1702.07836.
34. Alvey, B.J.; Anderson, D.T.; Yang, C.; Buck, A.; Keller, J.M.; Yasuda, K.E.; Ryan, H.A. Characterization of Deep Learning-Based Aerial Explosive Hazard Detection using Simulated Data. In Proceedings of the 2021 IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, FL, USA, 5–7 December 2021; pp. 1–8. [CrossRef]
35. Yager, R.R.; Ford, K.M.; Cañas, A.J. An approach to the linguistic summarization of data. In *Uncertainty in Knowledge Bases*; Bouchon-Meunier, B., Yager, R.R., Zadeh, L.A., Eds.; Springer: Berlin/Heidelberg, Germany, 1991; pp. 456–468.
36. Kacprzyk, J.; Zadrożny, S. Linguistic database summaries and their protoforms: Towards natural language based knowledge discovery tools. *Inf. Sci.* **2005**, *173*, 281–304. [CrossRef]
37. Xu, Z. *Linguistic Aggregation Operators: An Overview*; Springer: Berlin/Heidelberg, Germany, 2007. [CrossRef]
38. Popek, G.; Katarzyniak, R.P. Interval-based aggregation of fuzzy-linguistic statements. In Proceedings of the 2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Shenyang, China, 23–25 July 2013; pp. 510–514. [CrossRef]
39. de Jong, T. Cognitive load theory, educational research, and instructional design: Some food for thought. *Instr. Sci.* **2010**, *38*, 105–134. [CrossRef]
40. Kacprzyk, J.; Zadrożny, S. Linguistic Data Summarization: A High Scalability through the Use of Natural Language? In *Scalable Fuzzy Algorithms for Data Management and Analysis: Methods and Design*; IGI Global: Hershey, PA, USA, 2009; pp. 214–237. [CrossRef]
41. Alvey, B.; Anderson, D.; Keller, J. Explainable AI via Linguistic Summarization of Black Box Computer Vision Models. In Proceedings of the IEEE Conference on Artifical Intelligence (CAI), Santa Clara, CA, USA, 5–6 June 2023.
42. Smith, R.E.; Anderson, D.T.; Ball, J.E.; Zare, A.; Alvey, B. Aggregation of Choquet integrals in GPR and EMI for handheld platform-based explosive hazard detection. In *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXII*; Bishop, S.S., Isaacs, J.C., Eds.; International Society for Optics and Photonics, SPIE: San Diego, CA, USA, 2017; Volume 10182, p. 1018217. [CrossRef]
43. Alvey, B.; Ho, D.K.C.; Zare, A. Fourier features for explosive hazard detection using a wideband electromagnetic induction sensor. In *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXII*; Bishop, S.S., Isaacs, J.C., Eds.; International Society for Optics and Photonics, SPIE: San Diego, CA, USA, 2017; Volume 10182, p. 101820E. [CrossRef]
44. Veal, C.; Dowdy, J.; Brockner, B.; Anderson, D.T.; Ball, J.E.; Scott, G. Generative adversarial networks for ground penetrating radar in hand held explosive hazard detection. In *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIII*; Bishop, S.S., Isaacs, J.C., Eds.; International Society for Optics and Photonics, SPIE: San Diego, CA, USA, 2018; Volume 10628, p. 106280T. [CrossRef]
45. Bastian, M.; Heymann, S.; Jacomy, M. Gephi: An Open Source Software for Exploring and Manipulating Networks. In Proceedings of the International AAAI Conference on Weblogs and Social Media, San Jose, CA, USA, 17–20 May 2009.
46. Shahriar, S.; Hayawi, K. Let's have a chat! A Conversation with ChatGPT: Technology, Applications, and Limitations. *arXiv* **2023**, arXiv:2302.13817. https://doi.org/10.48550/arXiv.2302.13817.
47. Lu, Y.; Bartolo, M.; Moore, A.; Riedel, S.; Stenetorp, P. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. *arXiv* **2022**, arXiv:2104.08786.
48. White, J.; Fu, Q.; Hays, S.; Sandborn, M.; Olea, C.; Gilbert, H.; Elnashar, A.; Spencer-Smith, J.; Schmidt, D.C. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. *arXiv* **2023**, arXiv:2302.11382.

Chapter 2

Title: Explainable AI for the Choquet Integral

Venue: IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE

Date Published: August 4, 2021

Explainable AI for the Choquet Integral

Bryce J. Murray, *Student Member, IEEE*, Muhammad Aminul Islam, *Member, IEEE*, Anthony J. Pinar, *Member, IEEE*, Derek T. Anderson, *Senior Member, IEEE*, Grant J. Scott, *Senior Member, IEEE*, Timothy C. Havens, *Senior Member, IEEE*, and James M. Keller, *Life Fellow, IEEE*

***Abstract*—The modern era of *machine learning* is focused on data-driven solutions. While this has resulted in astonishing leaps in numerous applications, explainability has not witnessed the same growth. The reality is, most machine learning solutions are black boxes. Herein, we focus on data/information fusion in machine learning. Specifically, we explore four eXplainable Artificial Intelligence (XAI) questions relative to Choquet integral; (i) what is the quality of our inputs and their interactions, (ii) how is the information being combined, (iii) what is the quality of our training data (and thus our learned models), and (iv) what trust do we place in an output? Previously, we derived an initial set of indices for (i)–(iv) on the premise of perfect knowledge. Herein, we make XAI more accurate by taking into consideration what the machine learned. A combination of synthetic data and real-world experiments from remote sensing for fusing deep learners in the context of classification are explored. Our approach leads to performance gain, insights into what was learned, and it helps us realize better future solutions.**

***Index Terms*—XAI, explainable artificial intelligence, data fusion, information fusion, deep learning.**

I. Introduction

SINCE the dawn of computing, machines have been designed to carry out lists of deterministic operations given to them. In the last few decades, factors like Big Data and machine learning (ML) have given rise to a so-called data-driven era of *artificial intelligence* (AI). On one hand, we have data to help approximate parameters. On the other hand, the vast majority of algorithms are black box solutions. Sometimes, it might not be imperative to have an explanation as to what was learned; the solution in itself is all that is required. However, in other settings it may be vital to understand why a decision was reached. This need is a driving factor behind *explainable artificial intelligence* (XAI). Another advantage of XAI is finding gaps in current AI to accelerate the field.

The big question in XAI is, what is the question! Specifically, what do we want from XAI and what constitutes an explanation? Common inquiries include: "what is the worth of our inputs?," "why did an AI make a particular decision?," "what evidence and process(es) were used?," "what confidence does our machine have in its decision?," "does our AI and/or data have a bias? (age, race, sex, etc.)," "what trust do we place in the machine and its decision," etc. There are many questions that make up XAI, some of which are specific to a particular tasks or tool, e.g. what features were learned by a convolutional neural net (CNN)?

At a high-level, XAI can be divided into two classes: methods that are naturally explainable from the ground up and methods that elicit an explanation post-learning. An example of "XAI by design" in computational intelligence is fuzzy logic. Often, IF-THEN rules are formed with meaningful linguistic variables whose values are modeled by fuzzy sets over appropriate domains. These rules are fired, aggregated, and mapped into linguistic expression. However, it is often the case that fuzzy logic learned from data might not be interpretable [1]–[3]. Another example of explainability in the fuzzy set community is linguistic summarization [4], [5]. Linguistic summaries are a high-level description provided for the user of the system. However, not all AI is created with explainability in mind. For instance, neural networks are not developed in this manner; this AI requires extracting any explanations after the model has been learned from data. Specifically, various methods have been developed to visually understand what a CNN is "looking at" in an image [6]–[8]. There are numerous works and discussions about XAI in the literature. The reader can refer to [9] and [10] for a high-level overview and review of recent work.

Herein, we examine data/information fusion, a piece of the AI/ML puzzle. Namely, we focus on the *fuzzy integral* (FI). The FI is selected because it has been successfully applied across a variety of applications like computer vision [11], multi-criteria decision making [12], [13], multi-sensor fusion [14], remote sensing [15], neural networks [16], etc. The FI is a parametric function that yields a wide class of operators used in practice; the max, min, linear order statistics, and more. The FI aggregates data from *sources*–people, sensors, algorithms, and combinations thereof. Let $X = \{x_1, \ldots, x_N\}$ be N sources and $h(x_i) \in \mathbb{R}$ is the input from source i. In general, an aggregation function is a mapping of data, $\mathbf{h} = \{h_1, \ldots, h_N\}$ (where $h_i = h(x_i)$),

Manuscript received August 7, 2019; revised November 13, 2019 and May 26, 2020; accepted June 14, 2020. Date of publication July 27, 2020; date of current version July 22, 2021. *(Corresponding author: Bryce Murray.)*

Bryce J. Murray is with the Electrical Engineering and Computer Science Department, University of Missouri, Columbia, MO 65211 USA (e-mail: bmndc@mail.missouri.edu).

Muhammad Aminul Islam is with the Department of Electrical & Computer Engineering and Computer Science, University of New Haven, Connecticut, CT USA (e-mail: amin_b99@yahoo.com).

Anthony J. Pinar and Timothy C. Havens are with the Electrical Engineering and Computer Science Departments, Michigan Technological University, MI 49931 USA (e-mail: ajpinar@mtu.edu; thavens@mtu.edu).

Derek T. Anderson, Grant J. Scott, and James M. Keller are with the Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, MO 65211 USA (e-mail: andersondt@missouri.edu; grantscott@missouri.edu; kellerj@missouri.edu).

Digital Object Identifier 10.1109/TETCI.2020.3005682

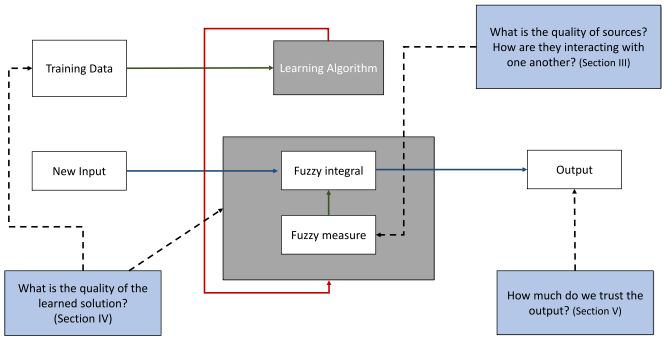


Fig. 1. Depiction of the XAI questions asked herein and their respective locations in the overall fusion process.

to data, i.e., $f_\Theta(\mathbf{h}) \rightarrow \mathbf{y}$, where Θ are model parameters. The FI is defined with respect to a capacity, which can be learned from data [17]. The focus of our article is XAI as it pertains to answering questions about aggregation.

There are many "types" of FIs; herein, we focus on the Choquet integral (ChI) [18], one of the most utilized *types* of FI. Furthermore, the ChI has many desirable properties, e.g., it recovers the Lebesgue integral for an additive/probability measure. There are many FIs, such as the Sugeno integral [19], [20], the non direct FI [21], the shape preserving FI [22], the general FI [23], the Shilkret integral [24], etc. A common thread across FIs is their underlying *fuzzy measure* (FM). In return, it is the FM that ultimately allows our proposed indices to be applicable to integrals beyond the ChI.[1] The combination of the above is why we focus on a single integral, the ChI. Exploring different integrals and observing if there are any advantages of one over there other for XAI is beyond the scope of this article. This article aims to lay a foundation for such future work.

This paper makes the following specific contributions. First, we show that it is typical that a significant number of fusion variables are often not learned from data. As such, we identify which variables are *data unsupported* (referred to hereafter as missing). On this premise, we extend the Shapley (i.e. how important is each source) and the interaction index (i.e. how do sources interact) to data-driven partially observable domains. Next, we address the question "how good is a fusion decision" based on the learned fusion model. Last, we apply our XAI tools to synthetic data and real-world benchmark data in the context of classification from remote sensing to demonstrate and highlight their utility.

An overview of the paper is as follows (see Fig. 1). In Section II we review the ChI, Sections III–V highlight existing XAI tools and our new extensions, and last, Section VI explores the fusion of different deep learners.

II. Measure and Choquet Integral

On a discrete (finite X) domain, the *fuzzy measure* (FM), $\mu : 2^X \rightarrow \mathbb{R}^+$, is a function that satisfies the following two properties: (i) (boundary condition) $\mu(\emptyset) = 0$, and (ii) (monotonicity) if $A, B \subseteq X$, and $A \subseteq B$, then $\mu(A) \leq \mu(B)$.[2] One way to think about the FM is that it models the "interactions" (e.g., statistical correlations) between subsets of sources.

Let $h(x_i) \in \mathbb{R}^{\geq 0}$ be the *input* from source i. The ChI of $\mathbf{h} = (h_1, h_2, \ldots, h_N)^t$, where $h_i = h(x_i)$, is

$$\int \mathbf{h} \circ \mu = \sum_{j=1}^{N} h_{\pi(j)}(\mu(A_{\pi(j)}) - \mu(A_{\pi(j-1)})), \quad (1)$$

for $A_{\pi(j)} = \{x_{\pi(1)}, \ldots, x_{\pi(j)}\}$, $\mu(A_{\pi(0)}) = 0$, and π such that $h_{\pi(1)} \geq h_{\pi(2)} \geq \cdots \geq h_{\pi(N)}$. Once the FM is defined, the ChI becomes a specific aggregation operator. For example, the ChI becomes the max operator for $\mu(A) = 1, \forall A \in X$, excluding $\mu(\emptyset)$ which is 0. One useful way to discuss the ChI is in terms of $N!$ *linear convex sums* (LCSs). Relative to a particular sort of the data (π_i)–of which there are $N!$ possible sorts–the ChI is

$$\sum_{j=1}^{N} h_{\pi_i(j)}(\mu(A_{\pi_i(j)}) - \mu(A_{\pi_i(j-1)})) = \mathbf{h}_{\pi_i}^t \mathbf{w}_{\pi_i}, \quad (2)$$

where $w_{\pi_i}(j) = \mu(A_{\pi_i(j)}) - \mu(A_{\pi_i(j-1)})$. For the ChI, these $N \times N!$ weights are tied to the underlying 2^N FM variables. The next section addresses one way in which these variables can be defined.

A. Optimization of the ChI

In this section, one method for determining the ChI is discussed. Most of the time it is impractical to have an expert define the FM; as the number of variables increases, the likelihood of arriving at a meaningful FM is not highly probable. Another route is to just determine the values of each of the singletons (densities) and to use a formula to calculate the remaining variables; e.g., the Sugeno λ-FM or the S-Decomposable FM [19]. While there are many methods to optimize the ChI (see [25]–[34]), we limit our scope to the following data-driven method. Full mathematical detail and experiments can be found in [17].

Let $O = \{\mathbf{h}_j, y_j\},\ j = 1, \ldots, M$, be M training examples; where $\mathbf{h}_j$ is the j-th instance with data/information from N inputs and y_j is the ground-truth (e.g., class label, regression output, etc.) for $\mathbf{h}_j$. The *sum of squared error* (SSE) for O is

$$E(O, \mathbf{u}) = \sum_{j=1}^{M} e_j^2 = \sum_{j=1}^{M} (\mathbf{c}_j^T \mathbf{u} - y_j)^2 = ||D\mathbf{u} - \mathbf{y}||_2^2, \quad (3)$$

where $\mathbf{u} = [\mu(\{x_1\}), \ldots, \mu(\{x_1, x_2\}), \mu(\{x_1, x_3\}), \ldots, \mu(X)]$ (lexiographic vector of size $2^N - 1$), $D = [\mathbf{c}_1\ \mathbf{c}_2\ \cdots\ \mathbf{c}_M]^T$ (full dataset), $\mathbf{y} = [y_1\ y_2\ \cdots\ y_M]^T$, $||\cdot||_2$ is norm-2 operation, and $\mathbf{c}_j$ holds the coefficients of $\mathbf{u}$ for observation $\mathbf{h}_j$, e.g., for $N = 3$ and $h(x_2) \geq h(x_1) \geq h(x_3)$,[3]

$$c = [0, h(x_2) - h(x_1), 0, h(x_1) - h(x_3), 0, 0, h(x_3)]\,.$$

The regularized SSE optimization problem is

$$\min_{\mathbf{u}} f(\mathbf{u}) = ||D\mathbf{u} - \mathbf{y}||^2 + \beta v(\mathbf{u}), \quad (4)$$

[1]The core principles are the same, however other integrals might result in different data observed FM variables. As a result, the formulas could change.

[2]Sometimes a normality condition is imposed such that $\mu(X) = 1$.

[3]The dimensionality of variables are as follows: $\mathbf{h}_j \in \mathcal{R}^N$, $y_j \in \mathcal{R}$, $\mathbf{u} \in \mathcal{R}^{2^N-1}$, and $c \in R^{2^N-1}$

where $\beta \in \mathbb{R}^{\geq 0}$ is a regularization constant (which balances the "cost" (or penalty) of obtaining minimum function error relative to our desire to have minimal model complexity) and $v(\mathbf{u})$ is an index of model complexity (e.g., k-additive and Mobius, Gini-Simpson, ℓ_p-norm, etc. [35]), subject to the FM boundary and monotonicity conditions (see [36] for how to pack the constraints into a linear algebra expression), which can be solved via quadratic programming. Full code and explanation (including how to build the constraint matrix C) can be found at https://github.com/B-Mur/ChoquetIntegral.

B. Data Supported and Unsupported Variables

A benefit of working with the ChI is knowing how training data maps to ChI parameters. That is, we can identify unsupported variables. In [17], we showed that the sort operation in a discrete ChI reveals which FM variables are exercised. For sort π_i, we get $w_{\pi_i}(j) = \mu(A_{\pi_i(j)}) - \mu(A_{\pi_i(j-1)})$. For example, let $N = 3$ and let $h_2 > h_3 > h_1$. The FM variables that are encountered are $\mu(\{h_2\})$, $\mu(\{h_2, h_3\})$ and $\mu(\{h_1, h_2, h_3\})$. By taking the entire training set of data into account, we can identify which of the variables are never encountered. Herein, we will exploit this property to build our XAI tools. It is our opinion that the ChI is a wonderful framework for XAI because the math is explainable by design, and as we show below, tools exist to tie these properties to what was extrapolated from data.

III. Measure Centric Indices

Numerous researchers have created indices to help explain the ChI, but on the basis that the full measure/ChI has been observed. However, as we showed, this is not often the case for data-driven fusion. Two of the most common indices are the Shapley index [37] and the interaction index [38]. The Shapley index assigns a value to each input to describe its *worth*; whereas, the interaction index computes how much two inputs *interact* (are they redundant, independent, or complimentary?). However, to date, all have assumed that the FM was fully obtainable, i.e., all variables were explicitly learned or determined. Herein, we propose new indices that do not have this inherent missing information bias.

A. Shapley Index

XAI Question: How good is a source?

The Shapley index is a way to summarize the 2^N variables in the FM. Generally, they are interpreted to be the *worth* of the different sources. The Shapley index is

$$\Phi_\mu(i) = \sum_{K \subseteq X \backslash \{i\}} \zeta_{X,1}(K) \left(\mu(K \cup \{i\}) - \mu(K)\right), \quad \text{(5a)}$$

$$\zeta_{X,1}(K) = \frac{(|X| - |K| - 1)!|K|!}{|X|!}, \quad \text{(5b)}$$

where $K \subseteq X \backslash \{i\}$ denotes all proper subsets from X that do not include source i. The Shapley values of μ is the vector $\mathbf{\Phi}_\mu = [\Phi_\mu(1), \ldots, \Phi_\mu(N)]^t$ where $\sum_{i=1}^{N} \Phi_\mu(i) = 1$. The Shapley index can be interpreted as the average amount of *contribution* of source i across all coalitions. Intuitively, it is a weighted sum of the difference between the ensemble not including source i and the ensemble including source i. To further illustrate the Shapley index, a numerical example is provided (see Table I). This example is built on the ChI presented in Fig. 2.

TABLE I
Shapley Index for FM Shown in Fig. 2

$\Phi_{\mu(1)}$	$.68 = \frac{1}{3}(.7-0) + \frac{1}{6}(.9-.1) + \frac{1}{6}(.7-.2) + \frac{1}{3}(1-.3)$
$\Phi_{\mu(2)}$	$.18 = \frac{1}{3}(.1-0) + \frac{1}{6}(.9-.7) + \frac{1}{6}(.3-.2) + \frac{1}{3}(1-.7)$
$\Phi_{\mu(3)}$	$.13 = \frac{1}{3}(.2-0) + \frac{1}{6}(.7-.7) + \frac{1}{6}(.3-.1) + \frac{1}{3}(1-.9)$

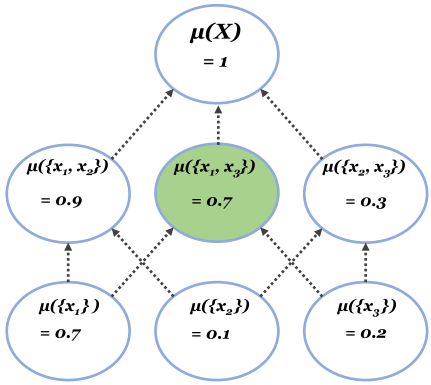


Fig. 2. Visualization of an example FM for N = 3. All variables are observed except $\mu(\{x1, x3\})$ (green).

B. Walk Centric Shapley

XAI: How good is a source relative to what we observed?

The Shapley index can be modified in order to exclude any variables that are not data supported. Hereafter, we refer to each input sort (π_i) as a *walk*. The FM can be visualized as a Hasse Diagram, where nodes are the FM variables and edges are the monotonicity constraints. Furthermore, a layer in the Hasse Diagram is all variables of equal size cardinality, i.e., layer 1 is $\{\mu(\{x_1\}), \mu(\{x_2\}), \ldots\}$, layer 2 is all tuples, etc. In the ChI, each input sort results in selecting one variable in each layer. Equation (6a) is the Walk-Centric Shapley index,

$$\bar{\Phi}_\mu(i) = \sum_{K \subseteq X \backslash \{i\}} \zeta_{X,2}(K, i) \left(\mu(K \cup \{i\}) - \mu(K)\right), \quad \text{(6a)}$$

$$\zeta_{X,2}(K, i) = \frac{(|X| - |K| - 1)!|K|!}{|X|!} \mathbb{1}_{(K \cup \{i\})} \mathbb{1}_{(K)}, \quad \text{(6b)}$$

$$\widetilde{\Phi}_\mu(i) = \frac{\bar{\Phi}_\mu(i)}{\sum_{j=1}^{N} \bar{\Phi}_\mu(j)}. \quad \text{(6c)}$$

Extending the Shapley index required including the indicator function, $\mathbb{1}$, which takes the value of 1 if the FM value is data-supported and 0 otherwise. The Shapley index values sum to 1, which helps with our interpretation of their relative worth degrees. Herein, we normalize the Walk Centric Shapley index in order to maintain the same semantics.

The calculation of the Walk Centric Shapley index, relative to Fig. 2, is provided in Table II. The variable, $\mu(\{x_1, x_3\})$, is

TABLE II
WALK CENTRIC SHAPLEY FOR FIG. 1

$\widetilde{\Phi}_{\mu(1)}$	$.6 = \frac{1}{3}(.7-0)+\frac{1}{6}(.9-.1)+(0)+\frac{1}{3}(1-.3)$
$\widetilde{\Phi}_{\mu(2)}$	$.08 = \frac{1}{3}(.1-0)+\frac{1}{6}(.9-.7)+\frac{1}{6}(.3-.2)+(0)$
$\widetilde{\Phi}_{\mu(3)}$	$.13 = \frac{1}{3}(.2-0)+(0)+\frac{1}{6}(.3-.1)+\frac{1}{3}(1-.9)$
$\widetilde{\Phi}_{\mu}(1)$	$.74 = \frac{.6}{.81}$
$\widetilde{\Phi}_{\mu}(2)$	$.09 = \frac{.08}{.81}$
$\widetilde{\Phi}_{\mu}(3)$	$.16 = \frac{.13}{.81}$

TABLE III
INTERACTION INDEX FOR FIG. 2

$\mathcal{I}_\mu(1,2)$	$.15 = \frac{1}{2}(.9-.7-.1+0)+\frac{1}{2}(1-.7-.3+.2)$
$\mathcal{I}_\mu(1,3)$	$-.15 = \frac{1}{2}(.7-.7-.2+0)+\frac{1}{2}(1-.9-.3+.1)$
$\mathcal{I}_\mu(2,3)$	$.05 = \frac{1}{2}(.3-.1-.2+0)+\frac{1}{2}(1-.9-.7+.7)$

data-unsupported. As such, computation on any term containing this variable is not considered. For example, in $\bar{\Phi}_\mu(1)$ (first row of Table II), the third term is 0 because it contains the $\mu(\{x_1, x_3\})$. The reader can see that a single missing variable leads to a noticeable change of the second source's worth from 0.18 to 0.08. The point is, unsupported variables change the way that one thinks about worth relative to what is observable.

C. Interaction Index

XAI: How are two sources interacting?

Next, we explore the interaction index, which informs us about how well two sets of sources interact with one another. Similar of the Shapley index, the interaction index assumes a fully observable FM. The interaction index [38] between i and j is

$$\mathcal{I}_\mu(i,j) = \sum_{K\subseteq X\setminus\{i,j\}} \zeta_{X,3}(K)(\mu(K\cup\{i,j\}) - \mu(K\cup\{i\}) - \mu(K\cup\{j\}) + \mu(K)), \quad (7a)$$

$$\zeta_{X,3}(K) = \frac{(|X|-|K|-2)!|K|!}{(|X|-1)!}, \quad (7b)$$

where $\mathcal{I}_\mu(i,j) \in [-1,1], \forall i,j \in \{1,2,\ldots,N\}$. A value of 1 (respectively, -1) represents the maximum complementary (respectively, redundancy) between i and j. A numerical example is provided in Table III with respect to Fig. 2.

D. Walk-Centric Interaction

XAI: How are two sources interacting, relative to what we observed?

Next, we extend the interaction index to listen only to data supported variables.

$$\widetilde{\mathcal{I}}_\mu(i,j) = \sum_{K\subseteq X\setminus\{i,j\}} \zeta_{X,4}(K,i,j)(\mu(K\cup\{i,j\}) - \mu(K\cup\{i\}) - \mu(K\cup\{j\}) + \mu(K)), \quad (8a)$$

TABLE IV
WALK-CENTRIC INTERACTION INDEX FOR FIG. 2

$\widetilde{\mathcal{I}_\mu}(1,2)$	$.1 = (.9-.7-.1+0)+(0)$
$\widetilde{\mathcal{I}_\mu}(1,3)$	$-.1 = (0)+(1-.9-.3+.1)$
$\widetilde{\mathcal{I}_\mu}(2,3)$	$0 = (.3-.1-.2+0)+(0)$

Algorithm 1: Variable Visitation.

1: INPUT: Data set O
2: INPUT: FM μ
3: Make a vector of counters, $\mathbf{v}$, where $|\mathbf{v}| = |\mathbf{u}|$, and set each value to zero
4: **for** $j = 1$ to M **do** ▷each training sample
5: Identify the $N-1$ FM variables for sample j and increment their corresponding values in v
6: **end for**
7: set v $= \frac{v}{M}$
8: RETURN: $\mathbf{v}$

$$\zeta_{X,4}(K,i,j) = \frac{\widehat{\zeta}_{X,4}(K,i,j)}{\sum_{\widehat{K}\subseteq X\setminus\{i,j\}} \widehat{\zeta}_{X,4}(\widehat{K},i,j)} \quad (8b)$$

$$\widehat{\zeta}_{X,4}(K,i,j) = \frac{(|X|-|K|-2)!|K|!}{(|X|-1)!} \mathbb{1}_{(K\cup\{i\})}\mathbb{1}_{(K\cup\{j\})}\mathbb{1}_{(K\cup\{i,j\})}\mathbb{1}_{(K)}. \quad (8c)$$

See Table IV for a numeric example of the walk-centric interaction index with respect to Fig. 2.

Remark 1: In Section III-D we extended the interaction index to partially observable domains with respect to two sources. In [39], Grabisch extended the interaction index to $\mathcal{I}(A,B)$, where $A, B \subseteq X$. Our missing variable weighting (Equation (8a)) naturally extends to interaction between sets of sources.

IV. DATA-CENTRIC INDICES

Oftentimes computational methods are applied to a data set based on intuition of the practitioner without any type of analysis of the data. Whereas Section III produced metrics that explain characteristics in and across the sources, this section's focus is insights into the quality of a learned ChI to a set of training data. We specifically address the data, directly, to determine their *diversity*. In this section, we present a set of indices that describes a different aspect of data diversity. However, understanding the diversity is only half of the story; we also present a new index to assess the degree to which we should *trust* a new sample's fused result.

A. Variable Visitation

XAI: How often was a FM variable observed?

Each of our indices are a different piece of evidence in the XAI big picture. The idea behind the next index (see Algorithm 1) is to determine the frequency for which a FM variable is observed in data.

TABLE V
NUMERICAL EXAMPLE OF THE VARIABLE VISITATION AND PERCENTAGE OF DATA SUPPORTED VARIABLES INDICES

Samples	FM Variables	Count	$\mathbf{v}$	$\hat{\mathbf{v}}$
$\{x_1, x_2, x_3\}$	$\{x_1\}$	2	.50	1
$\{x_2, x_1, x_3\}$	$\{x_2\}$	1	.25	1
$\{x_3, x_1, x_2\}$	$\{x_3\}$	1	.25	1
$\{x_1, x_3, x_2\}$	$\{x_1, x_2\}$	2	.5	1
	$\{x_1, x_3\}$	2	.5	1
	$\{x_2, x_3\}$	0	0	0

Table V is a numeric example for $N = 3$. The first column shows the $M = 4$ samples. Column two is the different FM variables. Column 3 is the number of times each variable was encountered. The next two columns are our proposed variable visitation index and its conversion to a binary seen/not seen.

B. Percentage of Data Supported Variables

XAI: How many FM variables were observed?

While knowing the relative frequency of visibility of each FM variable per layer is useful, it may still be too much information for a user to interpret. The next index reduces the 2^N probabilities into a single value that tells us total percentage of the variables seen. To this end, our variable visitation probabilities are "hardened." First, we convert $\mathbf{v}$ into $\hat{\mathbf{v}}$ such that an index in $\hat{\mathbf{v}}$ is 0 if its corresponding value in $\mathbf{v}$ is 0, otherwise it is assigned a 1. We calculate the number of nonzero observed probabilities,

$$\delta_1 = \sum_{k=1}^{|\hat{\mathbf{v}}|} \hat{\mathbf{v}}(k).$$

The percentage of data supported variables index is

$$i_1(\mu, O) = \frac{\delta_1}{|\hat{\mathbf{v}}|} \in [0, 1]. \tag{9}$$

Following the previous example from Table V, approximately 83% of the variables are supported.

C. Walk Visitation

XAI: What contexts did we observe?

The data-centric indices proposed thus far have provided valuable insights to the inner workings of a learned FM. However, each input/sample has a corresponding walk, which corresponds to one of the $N!$ LCS operators. Algorithm 2 exploits this property to provide the *walk visitation frequency*.

Algorithm 2 sorts each instance of the training data, and then it uses a mapping function to increment the number of times each sort has been seen. Once each iteration is complete, the values are normalized by the number of training instances.

Algorithm 2: Walk Visitation Frequency.

1: INPUT: Data set O
2: INPUT: FM μ
3: Let $\mathbf{z}$ (size $N!$) be a vector of all zeros
4: **for** $j = 1$ to M **do**
5: Sort (in descending order) the values in $\mathbf{h}_{\pi_j}$. The result is $\pi_j = [\pi_{j,1}, \ldots, \pi_{j,N}]$.[4]
6: Let each sort map to a unique index in $\mathbf{z}$. As such, let $\mathbf{z}(r(\pi_j)) = \mathbf{z}(r(\pi_j)) + 1$.[5]
7: **end for**
8: set $\mathbf{z} = \frac{z}{M}$
9: RETURN: $\mathbf{z}$

D. Percentage of Walks Observed

XAI: What percentage of contexts were observed?

Algorithm 2 provides insight into the relative frequency of how often each walk is encountered. Next, we outline an index that tells us what percentage of walks are seen. However, that again is information overload as there are $N!$ walks. Similar to i_1, z is "hardened." The percentage of LCSs observed is $i_2(\mu, O) = \frac{\delta_2}{N!} \in [0, 1]$, where δ_2 is the sum of the $\hat{\mathbf{z}}$ values. This index can be calculated based on the example provided in Table V. The percentage of walks seen is $\frac{4}{6}$.

E. Dominant Walk Identification

XAI: Are there any anomalous contexts?

The next index tells us about the diversity of the data. The idea behind this index is to look for irregularly high observation walks. It is computed as

$$i_3(\mu, O) = \max_k \mathbf{z}(k). \tag{10}$$

In theory, a balanced data set would have a uniform distribution for the probability of walk frequencies. As we discuss in the Section VI, cases where $i_3(\mu, O)$ is large are suspicious and need further analysis.

F. Complexity Analysis

Here, we provide a succinct analysis of the computational complexity of the proposed indices. It is trivial to show that the computational complexity of the Shapley index is $O(N2^{N-1})$, one difference term per monotonicity constraint. In the case of the Walk-Centric Shapley, complexity simply grows by a factor of 2 (the indicator functions) and the Walk-Centric Interaction index grows by a factor of 4 (the indicator functions). With respect to the data-centric indices, computing the walks and keeping track of the data-supported variables only requires running a sort on the M data points. The point is, the proposed indices do not come at any significant computational complexity, namely relative to the fact that they are often an offline procedure.

[4]For example, let $N = 2$ and $h(x_2) \geq h(x_1)$. The sort index (walk) would therefore be [2,1].

[5]Note, $r(\cdot) \in \{1, \ldots N!\}$ is the index resolving function for sort $\mathbf{s}_j$.

Algorithm 3: Potential variability index.

INPUT: New instance $\mathbf{h}$
INPUT: FM μ
Sort $\mathbf{h}$; which results in the walk $\mathbf{W_h}$
$t = 0$ ▷*Trust value (0 is good and 1 is bad)*
$I = \emptyset$ ▷*Uncertainty interval*
for $k = 1$ to $|\mathbf{W_h}|$-1 **do**
if $\mathbf{W_h}(k)$ is data unsupported **then**
Calculate $I_k = [l_k, u_k]$, where l_k is the max of the incoming variables (for $\mathbf{W_h}(k)$) at layer $k-1$ and u_k is the min of the respective variables in layer $k+1$
$I = I \bigcup I_k$
end if
end for
Set $t = \text{length}(I)$ ▷*Sum of uncertainty intervals*
RETURN: $t = 0$

V. Trust: Potential Variability Index

XAI: How much trust do we place in a ChI decision?

The above sections focus on answering XAI questions about learning. In this section, we address the question of how much should we trust the output of a ChI? Specifically, we focus on the viewpoint of how many missing parameters were used in a particular ChI instance. Ideally, we would like this value to be low, which corresponds to a low interval of uncertainty in that instance. To the best of our knowledge, this has not been tackled before with respect to the FI. Let $\mathbf{h}_{\pi_i}$ be the sorted input and $\mathbf{W_h}$ be its corresponding walk. For example, if $N = 4$ and $h_4 \geq h_2 \geq h_1 \geq h_3$ then $\mathbf{W_h} = (\{x_4\}, \{x_2, x_4\}, \{x_1, x_2, x_4\}, X)$. Each FM variable that was encountered during training is assigned a real-valued number during optimization. On the other hand, each missing variable has interval-valued uncertainty [17]. That is, each missing variable has an interval of uncertainty for a ChI instance because of the monotonicity constraints. The missing variable can take any value between the max of its corresponding variables at layer L-1 and the min of the corresponding variables at layer L+1. Algorithm 3 outlines how to calculate the potential variability index.

It should be noted that a low t does not guarantee that the ChI output is correct. It only informs us that data were sufficient to support the learning algorithm. The index t is merely evidence to warn a user when decisions are being made that were not encountered during training.

VI. Experiments

In order to understand how the proposed indices work in a real-world setting, we use the ChI to fuse a set of heterogeneous architecture *deep convolutional neural networks* (DCNNs) for object detection and land classification in remote sensing. Herein, we fuse seven DCNNs — CaffeNet [40], GoogleNet [41], ResNet 50 [42], ResNet 101, DenseNet [43], InceptionResNetV2 [44], and Xception [45] — on the AID remote sensing data set [46]. The AID data set contains 10,000 images of 30 different aerial scene types (see Table VI).

TABLE VI
Summary of the AID Data Set

Types	# images	Types	# images
airport	360	mountain	340
bare land	310	park	350
baseball field	220	parking	390
beach	400	playground	370
bridge	360	pond	420
center	260	port	380
church	240	railway station	260
commercial	350	resort	290
dense residential	410	river	410
desert	300	school	300
farmland	370	sparse residential	300
forest	250	square	330
industrial	390	stadium	290
meadow	280	storage tanks	360
medium residential	290	viaduct	420

TABLE VII
Fusion Improvement on AID Data Set

	CaffeNet	GoogleNet	ResNet 50	ResNet 101
Accuracy	0.9395	0.9528	0.9580	0.9589
	DenseNet	InceptionResNetV2	Xception	**ChI**
Accuracy	0.9480	0.9484	0.9622	**0.9821**

The DCNNs were trained following the procedures outlined in [47], which included state-of-the-art steps like transfer learning, data augmentation, drop out, and regularization. For fusion, the DCNNs were *locked*, which means no further training occurs in the networks. The DCNNs were trained using five-fold cross validation, meaning we have 5 sets and 80% of the data was used for training and 20% for testing. Per DCNN fold, two-fold cross validation is used for fusion. An approximately equal number of samples are randomly selected per class across the fusion folds to ensure that each class is represented in each of the folds. The reader can refer to [48] for a more detailed discussion of how to train the ChI; one per class or a shared ChI across classes.

Herein, it is not our goal to reiterate the already demonstrated accuracy of the ChI for DCNN fusion (see Table VII and [48]). Instead, our goal is to use the proposed XAI indices to shed light into what was learned.

A. Shapley Index-Based Discoveries

Due to the high number of experiments (five neural network cross validation folds, two fusion folds, and 30 object classes), all of the results are not reported. Instead, we report a few interesting discoveries (Table VIII).

As shown in Section III, missing variables can have a large impact on our understanding of a source's worth. Table VIII highlights a few instances from the AID data set where a source's worth fluctuates as much as 10% based on the classical Shapley definition versus our Walk-Centric Shapley formulation. Furthermore, it is worth highlighting that this fluctuation results in a different rank ordering of sources based on their individual worth.

TABLE VIII
SHAPLEY INDEX FOR AID DATA SET: SHAPLEY VALUE (RANK ORDER VALUE)

Class	Shapley index	CaffeNet	GoogleNet	ResNet 50	ResNet 101	DenseNet	InceptionResNetV2	Xception
1	All variables	0.0630 (1)	0.0870 (3)	0.0886 (4)	0.0760 (2)	0.0947 (5)	0.2211 (6)	**0.3696 (7)**
	Data supported variables	0.0499 (1)	0.0715 (3)	0.0753 (4)	0.0642 (2)	0.0880 (5)	0.2300 (6)	**0.4212 (7)**
6	All variables	0.0439 (1)	0.0710 (4)	0.0453 (2)	0.1228 (5)	0.1358 (6)	0.0688 (3)	**0.5124 (7)**
	Data supported variables	0.0331 (2)	0.0623 (4)	0.0279 (1)	0.0979 (5)	0.1240 (6)	0.0616 (3)	**0.5932 (7)**
28	All variables	0.0976 (2)	0.1328 (5)	0.1136 (4)	0.1025 (3)	0.0951 (1)	**0.3199 (7)**	0.1385 (6)
	Data supported variables	0.0634 (1)	0.1142 (5)	0.0949 (3)	0.0952 (4)	0.0804 (2)	**0.4336 (7)**	0.1182 (6)
30	All variables	0.1202 (3)	0.1613 (5)	0.1261 (4)	**0.1972 (7)**	0.1070 (2)	0.1828 (6)	0.1054 (1)
	Data supported variables	0.1406 (4)	0.1934 (6)	0.0697 (1)	0.1486 (5)	0.0975 (2)	**0.2410 (7)**	0.1093 (3)

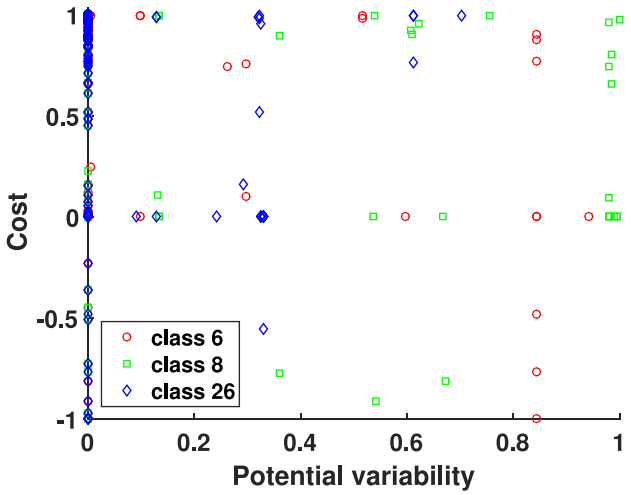


Fig. 3. Potential variability for classes, 6, 8, and 28.

B. Potential Variability Index-Based Discoveries

In this experimental section, there is not a "ground truth" *per se*. As such, our goal was to explore how our potential variability index affects the ability of fusion to compute an accurate output. In Fig. 3, values for the potential variability (horizontal-axis) and cost/penalty (vertical-axis) for samples from three different arbitrary classes are provided, where the cost is the confidence of the classifier and negative values represent missed classifications. Ideally, we would like to see a value of one on the vertical-axis with a potential variability of zero (aka, all variables are supported). However, Fig. 3 informs us that a low potential variability index does not guarantee an accurate cost value. This concept is demonstrated by class 26 (blue); as many points have a low cost despite having zero potential variability. On the other hand, a high potential variability does not mean a low cost. Class 8 (green) contains several samples that have a high potential variability while maintaining a high cost value. It is often said, garbage in, garbage out. Fusion is not guaranteed to solve every problem. If each DCNN says it is not the correct class, fusion will not fix that. For example, this index helped us discover that class 26 had errors because of the classifiers, not because there are data unsupported variables.

C. Data-Centric Index-Based Discoveries

Next, the data-centric indices exposed additional weaknesses. Table IX highlights the percentage of data supported variables, number of walks visited, percentage of walks observed, and the frequency of the top two most visited walks. Ideally, we would hope that the percentage of walks observed would be one and that the frequency of each walk would be equal. However, the reader can see that at most 2% of walks are observed and two walks typically dominate our training data. This is concerning because it tells us that there is a dangerously low amount of variety in our training data to aid learning a quality fusion solution. In one case, the top two walks account for over 80% of the data.

D. Overall XAI Assessment

The proposed indices need to be combined to garner a complete picture of the quality of our fusion solution. The reality is, these indices are not independent of one another. At a high-level, the indices highlight that we have dangerously low variety as it relates to learning a trustworthy fusion solution. Furthermore, the Shapley and interaction indices highlight that there are no definitive DCNNs that can be eliminated, especially when we look across classes. This is promising for fusion, meaning there is no single dominant network and fusion is likely needed and beneficial.

These XAI observations prompted us to investigate the dominant walks. The most frequently encountered sort order is 1, 2, 3, etc. The reason this occurs is because all the networks are producing the same value, e.g., all zeros, ones, etc. Our analysis has led us to believe that our DCNNs are too strong of classifiers — they almost always say the same thing, right or wrong. As a result, we should explore weaker learners and simultaneously learning of nets alongside fusion. This is not alarming, it is a strategy used in areas like ensemble learning.

In summary, only a very small part of our fusion solution was learned and as a result, it is likely that future scenarios can arise and impact the generalization of our model.

TABLE IX
DATA-CENTRIC INDICES FOR DCNN FUSION ON THE AID DATA SET

Class	Percentage of data supported variables	Number of walks visited	Percentage of LCS observed	Frequency of top walk	Frequency of walk-2	Frequency of top two walks
1	0.780	55	0.011	0.385	0.310	0.696
2	0.638	39	0.008	0.411	0.382	0.793
3	0.567	32	0.006	0.373	0.317	0.689
4	0.622	36	0.007	0.475	0.317	0.792
5	0.701	50	0.010	0.376	0.313	0.689
6	0.701	49	0.010	0.298	0.277	0.575
7	0.709	56	0.011	0.335	0.309	0.644
8	0.740	59	0.012	0.341	0.309	0.651
9	0.677	50	0.010	0.465	0.250	0.715
10	0.591	38	0.008	0.415	0.304	0.719
11	0.654	42	0.008	0.338	0.285	0.623
12	0.449	22	**0.004**	0.462	0.332	0.794
13	0.803	70	0.014	0.361	0.282	0.643
14	0.441	25	0.005	0.561	0.244	**0.805**
15	0.709	42	0.008	0.393	0.334	0.728
16	0.567	34	0.007	0.438	0.300	0.738
17	0.512	31	0.006	0.462	0.332	0.794
18	0.772	67	0.013	0.373	0.280	0.653
19	0.669	42	0.008	0.458	0.267	0.726
20	0.732	57	0.011	0.402	0.260	0.662
21	0.701	47	0.009	0.442	0.274	0.716
22	0.756	57	0.011	0.339	0.285	0.624
23	0.819	90	0.018	0.405	0.219	0.623
24	0.795	59	0.012	0.341	0.303	0.644
25	0.890	96	**0.019**	0.301	0.264	0.565
26	0.591	37	0.007	0.428	0.369	0.797
27	0.850	82	0.016	0.363	0.270	0.632
28	0.559	35	0.007	0.407	0.331	0.738
29	0.512	32	0.006	0.343	0.330	0.673
30	0.370	23	0.005	0.464	0.307	0.770

VII. CONCLUSION

In this article, we proposed metrics to address different questions about a learned fusion. Specifically, we proposed indices to assess the quality of a model, quality of our sources and their interactions, and how much we should trust a particular output of the ChI. We demonstrated analytical cases and the benchmark AID remote sensing data set was used as a case study. Each index plays a pivotal role in understanding and diagnosing weaknesses in fusion. Furthermore, our extension of the Shapley and interaction indices enable a more accurate assessment based on what was observed from training data.

While this paper is a step in the right XAI direction, it merely provides evidence via different indices. As we showed in the Section VI, these indices should ideally be combined by an algorithm or expert to make even more useful high-level conclusions. In future work, we will seek additional mathematics/algorithms to infer and build linguistic summarizations. Another research avenue is to formally take the exposed weaknesses and to fold them into the optimization process. For example, we observed a potential flaw due to independently training DCNNs. A next step is to use our indices to help promote the simultaneous training of a set of diverse learners to increase performance.

REFERENCES

[1] L. Duu, G. Mauris, and P. Bolon, "A fast and accurate rule-base generation method for mamdani fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 2, pp. 715–733, Apr. 2018.

[2] O. Cordon, F. Herrera, and P. Villar, "Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 667–674, Aug. 2001.

[3] J. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May 1993.

[4] A. Jain and J. M. Keller, "Textual summarization of events leading to health alerts," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Medicine Biol. Soc.*, Aug. 2015, pp. 7634–7637.

[5] G. Smits, P. Nerzic, O. Pivert, and M.-J. Lesot, "Efficient generation of reliable estimated linguistic summaries," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–8.

[6] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vision*, 2014, pp. 818–833.

[7] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2010, pp. 2528–2535.

[8] Q. Zhang and S. Zhu, "Visual interpretability for deep learning: A survey," *Frontiers Inform. Technol. Electron. Eng.*, vol. 19, pp. 27–39, Jan. 2018, doi: 10.1631/FITEE.1700808.

[9] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[10] F. K. Doilovi, M. Bri, and N. Hlupi, "Explainable artificial intelligence: A survey," in *proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron.*, May 2018, pp. 0210–0215.

[11] H. Tahani and J. Keller, "Information fusion in computer vision using the fuzzy integral," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 3, pp. 733–741, May/Jun. 1990.

[12] M. Grabisch, "The application of fuzzy integrals in multicriteria decision making," *Eur. J. Oper. Res.*, vol. 89, no. 3, pp. 445–456, 1996.

[13] C. Labreuche, "Construction of a Choquet integral and the value functions without any commensurateness assumption in multi-criteria decision making," in *Proc. EUSFLAT Conf.*, 2011, pp. 90–97.

[14] R. E. Smith *et al.*, "Genetic programming based Choquet integral for multi-source fusion," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jul. 2017.

[15] X. Du and A. Zare, "Multiple instance Choquet integral classifier fusion and regression for remote sensing applications," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 5, pp. 1–8, May 2019.

[16] G. J. Scott, K. C. Hagan, R. A. Marcum, J. A. Hurt, D. T. Anderson, and C. H. Davis, "Enhanced fusion of deep neural networks for classification of benchmark high-resolution image data sets," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 9, pp. 1451–1455, Sep. 2018.

[17] M. A. Islam, D. T. Anderson, A. J. Pinar, and T. C. Havens, "Data-driven compression and efficient learning of the Choquet Integral," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 4, pp. 1908–1922, Aug. 2018.

[18] G. Choquet, "Theory of capacities," in *Annales de l'institut Fourier*, vol. 5. Institut Fourier, 1954, pp. 131–295.

[19] M. Sugeno, "Theory of fuzzy integrals and its applications," Ph.D. dissertation, Tokyo Institute Technol, Tokyo, Japan, 1974.

[20] M. Sugeno and T. Murofushi, "Pseudo-additive measures and integrals," *J. Math. Anal. Appl.*, vol. 122, pp. 197–222, 1987.

[21] M. Anderson, D. T. Anderson, and D. J. Wescott, "Estimation of adult skeletal age-at-death using the Sugeno fuzzy integral," *Amer. J. Phys. Anthropol.*, vol. 142, no. 1, pp. 30–41, 2010.

[22] T. C. Havens, A. J. Pinar, D. T. Anderson, and C. Wagner, "SPFI: Shape-preserving Choquet fuzzy integral for non-normal fuzzy set-valued evidence," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jul. 2018, pp. 1–6.

[23] D. T. Anderson, T. C. Havens, C. Wagner, J. M. Keller, M. F. Anderson, and D. J. Wescott, "Extension of the fuzzy integral for general fuzzy set-valued information," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 6, pp. 1625–1639, Dec. 2014.

[24] N. Shilkret, "Maxitive measure and integration," *Indagationes Mathematicae (Proceedings)*, vol. 74, pp. 109–116, 1971.

[25] M. A. Islam, D. Anderson, F. Petry, and P. Elmore, "An efficient evolutionary algorithm to optimize the Choquet integral," *Int. J. Intell. Syst.*, vol. 34, pp. 366–385, Mar. 2019.

[26] M. A. Islam, D. T. Anderson, A. Pinar, T. C. Havens, G. Scott, and J. M. Keller, "Enabling explainable fusion in deep learning with fuzzy integral neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 7, pp. 1291–1300, 2020.

[27] J. M. Keller and J. Osborn, "Training the fuzzy integral," *Int. J. Approx. Reasoning*, vol. 15, no. 1, pp. 1–24, 1996.

[28] A. J. Pinar, J. Rice, L. Hu, D. T. Anderson, and T. C. Havens, "Efficient multiple kernel classification using feature and decision level fusion," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 6, pp. 1403–1416, Dec. 2017.

[29] J. M. Keller and J. Osborn, "A reward/punishment scheme to learn fuzzy densities for the fuzzy integral," in *Proc. Int. Fuzzy Sys. Assoc. World Cong*, 1995, pp. 97–100.

[30] G. Beliakov, "Construction of aggregation functions from data using linear programming," *Fuzzy Sets Syst.*, vol. 160, no. 1, pp. 65–75, 2009.

[31] X. Du, A. Zare, J. M. Keller, and D. T. Anderson, "Multiple instance Choquet integral for classifier fusion," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2016, pp. 1054–1061.

[32] S. Angilella, S. Corrente, and S. Greco, "Stochastic multiobjective acceptability analysis for the Choquet integral preference model and the scale construction problem," *Eur. J. Oper. Res.*, vol. 240, pp. 172–182, Jan. 2015.

[33] J. Marichal and M. Roubens, "Determination of weights of interacting criteria from a reference set," *Eur. J. Oper. Res.*, vol. 124, pp. 641–650.

[34] S. Angilella, S. Greco, and B. Matarazzo, "Non-additive robust ordinal regression: A multiple criteria decision model based on the Choquet integral," *Eur. J. Oper. Res.*, vol. 201, pp. 277–288.

[35] A. J. Pinar, D. T. Anderson, T. C. Havens, A. Zare, and T. Adeyeba, "Measures of the shapley index for learning lower complexity fuzzy integrals," *Granular Comput.*, vol. 2, pp. 303–319, 2017.

[36] D. T. Anderson, S. R. Price, and T. C. Havens, "Regularization-based learning of the Choquet integral," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jul. 2014, pp. 2519–2526.

[37] L. S. Shapley, "A value for N-person games," *Contributions Theory Games*, vol. 2, pp. 307–317, 1953.

[38] T. Murofushi and S. Soneda, "Techniques for reading fuzzy measures (iii): Interaction index," in *Proc. 9th Fuzzy Syst. Symp.*, Sapporo, Japan, 1993, pp. 693–696.

[39] M. Grabisch and M. Roubens, "Application of the Choquet integral in multicriteria decision making," in *Fuzzy Measures and Integrals*. Physica Verlag, 2000, pp. 348–374.

[40] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, New York, NY, USA, 2014, pp. 675–678.

[41] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. Comput. Vision Pattern Recognit.*, 2015, pp. 1–9.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[43] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jul. 2017, pp. 2261–2269.

[44] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf Artif. Intell.*, 2016, pp. 4278–4284.

[45] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jul. 2017, pp. 1800–1807.

[46] G. Xia *et al.*, "AID: A benchmark data set for performance evaluation of aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3965–3981, Jul. 2017.

[47] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum, and C. H. Davis, "Training deep convolutional neural networks for land-cover classification of high-resolution imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 4, pp. 549–553, Apr. 2017.

[48] D. Anderson, G. Scott, M. Islam, B. Murray, and R. Marcum, "Fuzzy Choquet integration of deep convolutional neural networks for remote sensing," in *Computational Intelligence for Pattern Recognition*, W. Pedrycz and S.-M. Chen, Eds. Berlin, Germany: Springer, 2018.

Bryce J. Murray (Student Member, IEEE) received the B.S. degree in Computer Science, Mathematics, and Physics from Mississippi College, Clinton, MS, USA, in 2015 and M.S. degree in Electrical and Computer Engineering from Mississippi State University, Mississippi State, MS, USA, in 2018. He is a Ph.D. Candidate at the University of Missouri, Columbia, MO, USA. His interests include data/information fusion, machine learning, deep learning, computer vision, remote sensing, and eXplainable AI.

Muhammad Aminul Islam (Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2005 and the Ph.D. degree in electrical and computer engineering from Mississippi State University, Starkville, MS, USA in 2018. He is currently an Assistant Professor in the Department of Electrical & Computer Engineering and Computer Science at the University of New Haven. His research interests include deep learning, computer vision, data/information fusion, autonomous driving, and remote sensing.

Anthony J. Pinar (Member, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from Michigan Technological University, Houghton, MI, USA, in 2014 and 2017, respectively. He is currently a Lecturer with Michigan Technological University and his research interests include data fusion, machine learning, and signal processing.

Derek T. Anderson (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering (ECE) from the University of Missouri, Columbia, MO, USA, in 2010. He is an Associate Professor in electrical engineering and computer science (EECS) at the University of Missouri and director of the Mizzou Information and Data Fusion Laboratory (MINDFUL). His research is information fusion in computational intelligence for signal/image processing, computer vision, and geospatial applications. Dr. Anderson has published over a 150 articles, he received the Best Overall Paper Award at the 2012 IEEE International Conference on Fuzzy Systems (FUZZIEEE), and the 2008 FUZZ-IEEE best student paper award. He was the Program Co-Chair of FUZZ-IEEE 2019, an Associate Editor for the IEEE TRANSACTIONS ON FUZZY SYSTEMS, Vice Chair of the IEEE CIS Fuzzy Systems Technical Committee (FSTC), and an Area Editor for the International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.

Grant J. Scott (Senior Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in computer engineering and computer science from the University of Missouri, Columbia, in 2001, 2003, and 2008, respectively. He is a founding Director of the Data Science and Analytics Masters Degree program at the University of Missouri. He is an Assistant Professor with the Electrical Engineering and Computer Science Department, University of Missouri. Dr. Scott is exploring novel integrations of computational hardware and software to facilitate high performance advances in large-scale data science, computer vision, and pattern recognition. His current research efforts encompass areas such as: real-time processing of large-scale sensor networks, parallel/distributed systems, and Internet of Things (IoT) data; deep learning technologies applied to geospatial data sets for land cover classification and object recognition; extensions of enterprise RDBMS with HPC co-processors; crowd-source information mining and multi-modal analytics; high performance & scalable content-based retrieval (geospatial data, imagery, biomedical); imagery and geospatial data analysis, feature extraction, object-based analysis, and exploitation; pattern recognition databases and knowledge-driven high-dimensional indexing; and image geolocation. He has leveraged this experience in the development of innovative remote sensing (satellite and airborne) change detection technologies, resulting in 5 US Patents. He has participated in a variety of professional networking and academic events, as well as worked with a variety of groups to bring data science training to their people (MU Public Policy, USDA, IEEE international conferences).

Timothy C. Havens (Senior Member, IEEE) received B.S. and M.S. degrees in electrical engineering from Michigan Technological University and the Ph.D. degree in Electrical and Computer Engineering from the University of Missouri. Prior to joining Michigan Tech, he was an NSF/CRA Computing Innovation Postdoctoral Fellow at Michigan State University. Prior to his Ph.D. work, he was an Associate Technical Staff at MIT Lincoln Laboratory. He is the William and Gloria Jackson Associate Professor of Computer Systems and Associate Dean for Research in the College of Computing at Michigan Technological University. He is Director of the Institute of Computing and Cybersystems (ICC) and the ICC Center for Data Sciences. Dr. Havens has an active research program in the areas of sensor and data fusion, sensor and signal processing, and machine learning. He has published over 140 technical papers in various journals and conference proceedings. In 2012, Dr. Havens was awarded the Best Paper Award at the IEEE International Conference on Fuzzy Systems and in 2011 was awarded the IEEE Franklin V. Taylor Award for best paper at the IEEE Systems, Man, and Cybernetics conference. Dr. Havens is an Associate Editor for the IEEE TRANSACTIONS ON FUZZY SYSTEMS and was a General Co-Chair of the 2019 IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS.

James M. Keller (Fellow, IEEE) is now the University of Missouri Curators Distinguished Professor Emertitus in the Electrical Engineering and Computer Science Department on the Columbia campus. Jim is an Honorary Proferssor at the University of Nottingham. His research interests center on computational intelligence with a focus on problems in computer vision, pattern recognition, and information fusion including bioinformatics, spatial reasoning, geospatial intelligence, landmine detection and technology for eldercare. Professor Keller has been funded by a variety of government and industry organizations and has coauthored over 500 technical publications. Jim is a Life Fellow of the IEEE, is an IFSA Fellow, and a past President of NAFIPS. He received the 2007 Fuzzy Systems Pioneer Award and the 2010 Meritorious Service Award from the IEEE Computational Intelligence Society. He finished a full six year term as Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS, followed by being the Vice President for Publications of the IEEE CIS from 2005-2008, and then an elected CIS Adcom Member. He is VP Pubs for CIS again, and has served as the IEEE TAB Transactions Chair and as a member of the IEEE Publication Review and Advisory Committee from 2010 to 2017. Jim has had many conference positions and duties over the years.

Chapter 3

Title: How should simulated data be collected for AI/ML and unmanned aerial vehicles?

Venue: SPIE Defense + Commercial Sensing

Date Published: June 13, 2023

How should simulated data be collected for AI/ML and unmanned aerial vehicles?

Jeffrey Kerley, Derek Anderson, Brendan Alvey, Andrew Buck

Jeffrey Kerley, Derek T. Anderson, Brendan Alvey, Andrew Buck, "How should simulated data be collected for AI/ML and unmanned aerial vehicles?," Proc. SPIE 12529, Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications, 125290J (13 June 2023); doi: 10.1117/12.2663717

SPIE.

Event: SPIE Defense + Commercial Sensing, 2023, Orlando, Florida, United States

How Should Simulated Data Be Collected for AI/ML and Unmanned Aerial Vehicles?

Jeffrey Kerley, Derek T. Anderson, Brendan Alvey, and Andrew Buck

Electrical Engineering and Computer Science Department, University of Missouri, Columbia, MO 65211, USA

ABSTRACT

Large and diverse datasets can now be simulated with associated truth to train and evaluate AI/ML algorithms. This convergence of readily accessible simulation (SIM) tools, real-time high performance computing, and large repositories of high quality, free-to-inexpensive photorealistc scanned assets is a potential artificial intelligence (AI) and machine learning (ML) game changer. While this feat is now within our grasp, what SIM data should be generated, how should it be generated, and how can this be achieved in a controlled and scalable fashion? First, we discuss a formal procedural language for specifying scenes (LSCENE) and collecting sampled datasets (LCAP). Second, we discuss specifics regarding our production and storage of data, ground truth, and metadata. Last, two LSCENE/LCAP examples are discussed and three unmanned aerial vehicle (UAV) AI/ML use cases are provided to demonstrate the range and behavior of the proposed ideas. Overall, this article is a step towards closed-loop automated AI/ML design and evaluation.

Keywords: simulation, procedural, formal language, LSCENE, LCAP, AI, ML, artificial intelligence, machine learning, drone, unmanned aerial vehicle

1. INTRODUCTION

Current generation narrow versus general artificial intelligence (AI) is overly reliant on labeled data. But, where does the data and its truth come from? How large and diverse is the data and is it sufficient for achieving the task at hand? What is the overall cost for collecting and labeling the data? While older methodologies like unsupervised learning and newer strategies like self-supervised learning are being explored to address limitations with supervised learning, future AI will likely always be reliant on data with truth to some degree. In a similar vein to how data and high performance computing rejuvenated neural networks and help set deep learning (DL) in motion, simulation (SIM) is a powerful tool that can be used to help mitigate issues like the aforementioned. The current article is a step towards SIM for AI. Specifically, we explore the formal specification and procedural generation of a scene and collection.

Figure 1 illustrates our goal of closed-loop AI training and evaluation in SIM. The idea is an automated framework to learn a formal procedural language (P) to model a given task (T), e.g., object detection, monocular vision, drone autonomy, etc. While P is useful on its own for domain knowledge discovery, it can also be sampled by a generator to produce a dataset (D) to train and evaluate an AI model. However, in the real-world we do not always know what underlying information (I) drives T, let alone what D to collect in support of $\{I, T\}$. To date, we typically use expert knowledge to design a data collection under real-world constraints (time, money, etc.). However, this rarely translates into a dataset that is rich enough to train and/or truly evaluate a trustworthy, reliable, and unbiased ML model that operates across a range of contexts and environments. This trial-and-error process, which is how other challenges like material design work, is repeated until T is solved or funding is depleted. The point is, it is unlikely that a human knows I and will provide a sufficient initial P. This must be discovered iteratively in the real-world, in a surrogate environment like SIM, or most likely, a combined effort.

For sake of illustration, consider the following application. One of our research interests is object detection, identification, and dynamic interrogation of explosive hazards (EH) across environments, emplacements, sensors, and operating conditions (speed, standoff distance, etc.) using a low altitude unmanned aerial vehicle (UAV). For this particular application, task T is detection and localization of an EH. Information I that drives T can range from spatial shape features to multi-spectral, environment, platform, and/or emplacement context. We are

Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications, edited by Christopher L. Howell, Kimberly E. Manser, Raghuveer M. Rao, Proc. of SPIE Vol. 12529, 125290J
 · 0277-786X · doi: 10.1117/12.2663717

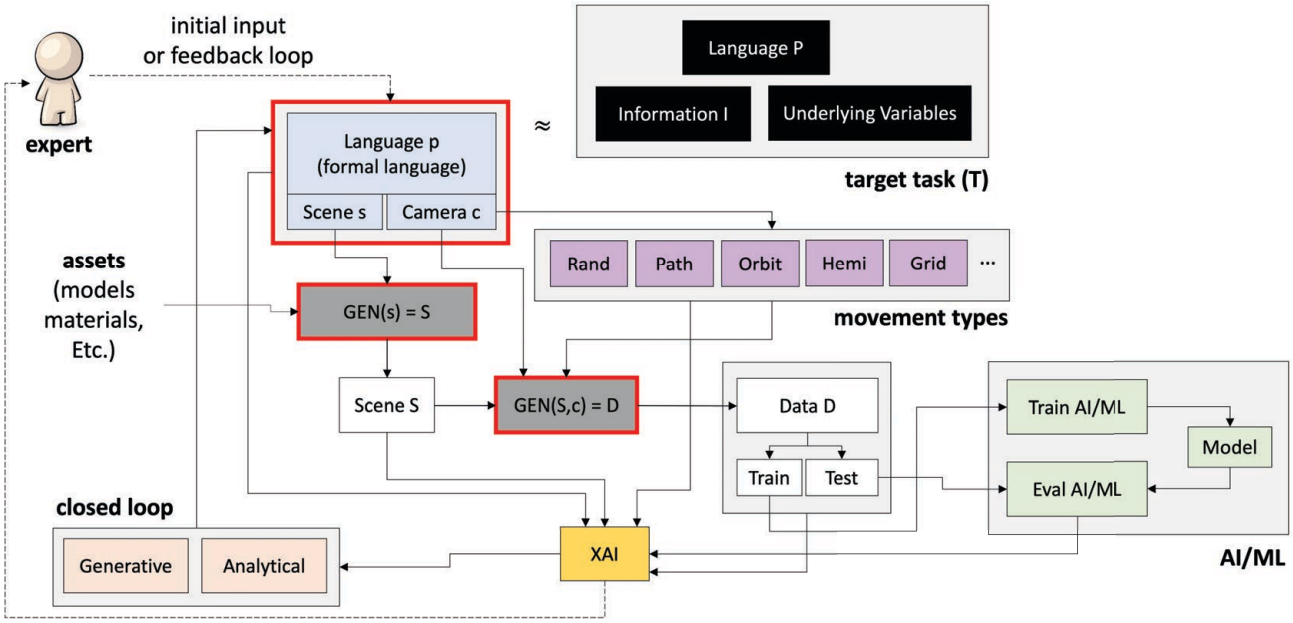


Figure 1: Our goal of a closed loop AI design and evaluation framework. Red indicates current article contributions. A target task is formally modeled via a procedural language. The resultant dataset is used to train an AI model and/or discover data, algorithm, and/or model biases and shortcomings.

interested in learning or discovering a language (formal task description) that is succinct yet sufficiently captures $\{I, T\}$. This challenge is extremely hard and costly to address in the real-world. SIM focused AI/ML has the potential to help us explore and address this task, or at least identify candidate solutions that can be explored further or in combination with the real-world in a more efficient and timely manner.

Herein, we put forth the following contributions. First, we investigate an explainable and scaleable procedural language for scene generation, called LSCENE, and data collection, LCAP. Next, we discuss data, ground truth, metadata, and details about how to produce, store, and format SIM information relative to state-of-the-art AI libraries, tools, and practices. Last, use cases and examples are provided to demonstrate the proposed ideas.

2. RELATED WORK

2.1 SIM Environments

A classical SIM environment, driven largely by robotics, is Gazebo.[1] While Gazebo is not state-of-the-art with respect to photorealism, it has extensively been integrated with the robotic operating system (ROS)[2] and it supports a wide range of physics and sensor SIM (e.g., RGB, RGBD, GPS, IMU, LiDAR, sonar, etc.). With respect to realism, Apple's HYPERSIM, Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding,[3] does not support physics but it produces nearly indistinguishable photorealistic imagery with information about scene geometry, material information, lighting, per-pixel instance segmentation's with respect to diffuse reflectance, diffuse illumination, and non-diffuse residual term that captures view-dependent lighting effects. Another approach is NVIDIA's Isaac Gym,[4] an end-to-end SIM platform, built from the ground up to run large-scale, physically accurate multi-sensor SIM for applications like autonomous vehicles and virtual worlds. The Unreal Engine (UE),[5] is another solution that has recently achieved near photorealistic results. UE has a generous licence that varies based on terms of use, ranging from free to percentage of profits. The UE started as a game engine and it has recently settled into new applications like architecture, film, computer graphics, and beyond. Last, a more recent competitor to UE is Unity.[6]

The above examples are from the robotics, computer graphics, gaming, and hobbyist communities. SIM dates back decades and its scope is beyond the current article. SIM frequently involves approaches like ray or finite

element modeling to accurately model topics like electromagnetics and physics. Examples include COMSOL[7] for multiphysics SIM and MEEP,[8] a free and open-source software package for electromagnetics simulation via the finite-difference time-domain (FDTD). These SIMs are often based on simulating sensors such as the visual spectrum (aka RGB) to multispectral, hyperspectral, radar, LiDAR, acoustics, and beyond. A recent physics-based example from the literature is the virtual autonomous navigation environment (VANE) and ANVEL.[9] The reader can refer to the internet for more large scale domain specific business examples like Scale AI.

While the number of SIM solutions seems endless, these tools are only a part of the big picture. How accurate are they? Most SIM solutions are provided "as is". That is, they have some level of trust associated with them, typically based on their design. However, most SIM solutions take shortcuts or they are discrete approximations to continuous processes. Only a few SIM environments have truly undergone a rigorous process of verification and validation (V&V). The reader can refer our article[10] for a recent historical review of the development of V&V theories for SIM models.

2.2 SIM and AI/ML

While Sub-Section 2.1 outlines different general SIM tools, the more relevant question to the current article is which of these have been used for AI/ML, and how? By far, the most work exists for training AI/ML/DL algorithms, e.g., synthetic data for DL,[11,12] imitation training,[13] selective instance switching,[14] indoor object detection,[15] driver assistance,[16] optical flow estimation,[17] urban semantic segmentation,[18] and adversarial training.[19] Work also exists in how to generate SIM ground truth, e.g., 5M photorealistic images of synthetic indoor trajectories,[20] playing for data,[21] and our recent article on flawed assumptions in SIM ground truth.[22] Another interesting focus is SIM digital twins and proxy worlds for AI/ML, e.g., virtual-KITTI.[23] While training is important, work also exists for evaluating AI/ML algorithms, e.g., evaluation of 3D reconstruction pipelines[24] and synthetic feature tracks for SfM evaluation.[25] Furthermore, while SIM is a wonderful place to experiment with AI/ML, works exist to try to bridge SIM models to the real world, e.g., domain randomization[26,27] and self-supervised monocular depth.[28] In many scenarios, we do not know what to SIM for AI/ML. Work exists to help learn what an AI/ML needs in SIM, e.g., autosimulate[29] and learning to simulate.[30] While full image SIM is just now starting to reach the point of photorealism where it might be able to be a good surrogate for the real world, a number of hybrid SIM and real data solutions exists, e.g., self-supervised image harmonization,[31] domain translation for image harmonization,[32] "cut, paste, and learn" for image synthesis and instance detection,[33] and a recent survey on deep image composition.[34] By no means is this a comprehensive list, these are just a few examples of SIM for AI/ML.

2.3 Procedural SIM

The process of generating content, e.g., textures, 3D models, etc., has been researched for several decades in the video game and film industries. In 1985, Perlin introduced algorithms around noise functions to create procedural materials.[35] Perlin first demonstrated these ideas on a marble vase. This procedural way of thinking grew to include methodologies like fractals and L-systems[36,37] for constructing plants, trees, and vegetation. Later works focused on real-time procedural terrains[38] and city generation.[39] While academically interesting, these ideas have been significantly scaled up in recent times by companies like SideFX in commercially available tools like Houdini. Many of these professional tools are now the standard for creating urban and rural cities in games and film. While much effort has gone into procedural algorithms for automatic content generation (including procedural creatures, animation, etc.), researchers have expanded this thinking and focused a users experience. For example, in Left4Dead, Valve introduced the AI Director,[40] a procedural algorithm that monitors user experience and changes the spawning and properties of entities in a map for dynamic pacing of game play. Procedural text-based games and procedural maps, e.g., dungeons, have existed for an even longer time in video games. The point is, while many procedural ideas have been put forth to date, most are focused on content generation and users (people). Relatively little research to date has gone into procedural algorithms for AI/ML. What type (volume, variety, and attributes) of data does AI/ML need? AI/ML is a relatively new "consumer" in the field of procedural.

2.4 Formal Languages and SIM

Successful video game procedural content generators still rely on the formulation of a grammar to specify the rules of generation. Not all grammars are created equally, however, and in the popular case of *No Man's Sky*, a video game developed in 2016, an L-system grammar was used for generation. This was done in an effort to better model natural systems that need to have high levelsXXX of details. Parallel rule rewriting in these systems help facilitate this process. As one branch of a simulated tree grows, each new branch grown will also be rewritten which leads to fractal like patterns. This process does allow for almost infinite detail, and is built from several discrete rules applied at each rewrite step. Simulating an entire level or entire scene, rather than a single object often requires a grammar with more control. Recently, DeepMind also presented an idea focused on AI called Open-Ended Learning Leads to Generally Capable Agents.[41] An agent played within simulated levels that had generic goals, both of which were procedurally generated using a formal grammar. These rules were capable of creating scenes such as "Put purple sphere near black pyramid" or "See the red player or stand on orange floor." These statements are then paired with four other statement descriptors which judge "Competitive, Balance, Options, and Exploration Difficulty." These metric functions take the difference between the maximum upper bounds of certain properties of scenes, and any new generated scene from the grammar sentence. As a result, a much more controlled environment is created for a reinforcement learning (RL) algorithm. Certain actions and other spatial relationships are present in this grammar, like holding, or being near another object. Huge amounts of meaningful data can now be simulated and generated procedurally, while still being able to maintain the appropriate level design to teach an RL algorithm.

3. PROCEDURAL LANGUAGES

This section describes the current state of our two procedural languages. While grammars and their implementations vary, solutions typically involve at a minimum *symbols* (terminals and non-terminals) and rules. To facilitate domain understanding, we desire a succinct language with understandable symbols and logic. Our language needs to also satisfy the request for mass content generation, while enforcing relevant rules. Furthermore, generated content should allow for strict labeling (markup) on every action taken. As a result of these constraints, we propose a rule-based procedural SIM language that relies on sampling a problem space, and an ordered hierarchy within a 3D scene. The next two sub-sections detail our current solution.

3.1 LSCENE: Scene Language

Table 1 is the combined LSCENE and LCAP symbol set and Table 2 is LSCENEs function set. We begin this sub-section with a working LSCENE example (see Example LSCENE 1). The aim is to help the reader develop intuition about what we have done before fine details are discussed.

Example LSCENE 1
Simple scene with a sky, single random object, and many rocks and bushes

```
LSCENE = ONE(S) ONE(W) ONE(O1) MANY(O2)
S = skybox{color=white,clouds:{ type={cumulus:0.5,nimbostratus:0.5},density=[0.1,0.2]}}
W = terrain{size=[201,201],height:{Perlin:0.2},material:grassy}
O1 = object{class:{A:0.5,B:0.5},rotation:{range:[0,0,0],[360,360,360]},position:{ON:W}}
IO1 = object{class:cube,visibility:false,extent[1000,1000,1]},position:{ON:W,AROUND:O1}}
O2 = object{class:{bush:0.2,rocks:0.8},scale:[1.0,1.5]},overlap:true,position:{ON:IO1}}
```

Example LSCENE 1 is a trivial environment that has a sky with a fixed color, one of two cloud types, and a range of possible cloud densities. Thus, this symbol (sky) has a combination of crisp, probabilities, and interval-valued data. The terrain is a fixed size, its geometry is determined by a Perlin noise function, and its material (texture) is of type grassy. Next, a single object will be placed on our terrain, from user defined class type A or B. The relative rotation and size of these objects have been specified, along with a spatial qualifier about where to place the object. Next, a non-renderable object is specified with a spatial qualifier. A number of other 3D objects, bushes or rocks, are placed in this volume. Overall, this simple LSCENE describes a grassy environment that has rocks, bushes, and a single object.

Table 1: LSCENE and LCAP Symbols

Symbols	**Meaning**
LSCENE	Start symbol for scene language
LCAP	Start symbol for capture language
S	Sky
DL	Direction Light
A	Atmosphere
SL	Skylight (Ambient Light)
W	Terrain
O	Object
CO	Camera orbit
P	Path
CL	Collection
SCS	Scene Change Set

Table 2: LSCENE Function Set

Function Name	**Meaning**
ON	Intersects a bounding box with the top of an object's bounding box
IN	Intersects a bounding box with an object's bounding box
AROUND	Intersects a bounding box with an object's bounding box with N-extent
ONE	Creates 1 object in LSCENE
MANY	Creates N objects in LSCENE
Position	Samples a 3D point from a bounding box and sets an object position
Rotation	Samples from a range of 3D rotation angles
Extent	Sets an object's model to extend to a given bounding box extent
Visible	True/false to display an Object's model
Bound	Sets an object's model extent to another object's size
Material	Samples a UE material and sets on an object's model
Scale	Samples between a min and max value to set an object's model scale
Overlap	True/false to allow an object's model to overlap other object models
Class	Adds a 3D model to an object
Blueprint Class	Inserts a created blueprint asset into an object
Perlin	Uses a terrain object and applies a Perlin noise function
Cloud	Uses a skybox and sets the specified cloud type.
Intensity	Uses a skybox and sets the lighting intensity (lux)
Density	Defines a percentage of a given input to use (clouds, atmospheric settings, etc)

Next, we proceed to a more detailed description of LSCENE. Table 1 outlines several components of a scene, namely, the sky (and all lighting), terrain, and any objects featured in this example scene. All symbols correspond to the bare minimum required to produce pseudo real SIM scenes. No constraints on the quantity of any element are specified, nor constraints on the type of each symbol outside of this minimal set (such as meta specifications like clutter, buildings, etc.). The goal of our initial language is to act as a basic interface for generating complex 3D scenes, not a specification that uses hundreds of properties about real objects. Additional rendering and scene details needed to achieve realism can still be used in tandem with our language. Each symbol can be arranged in an arbitrary hierarchical order. Many real world tasks require different constraints to hold true, such as geospatial imagery where atmospheric settings may override other lighting settings, or target classification which could need all targets to be fully visible. The JSON file is parsed sequentially, allowing back references to previously defined objects, such as lighting or terrain, which may be referenced by other objects. This simple format accurately covers what is involved in scene construction, and it allows for a minimum of rules to define a scene (easiest for user/algorithm readability and writing).

Our language relies on UE for 3D scene creation and manipulation. While our ideas can be extended to other SIM environments, UE was selected due to its diverse set of capabilities, photorealism, extended set of plugins (e.g., Infinite Studio for multi-spectral data), programming support, online community, licensing, and online free high quality 3D scanned assets like Quixel. Our LSCENE JSON file is translated into UE C++ functions and variables, and UE executes the instructions to the rendering pipeline. The world state is sequentially changed with each function. The overall speed of this system is constrained primarily by the initial JSON parsing. Afterwards, all code is in C++ function calls that maintain the typical speed of close to hardware programming languages. Our language aims to extend these UE functionalities, while preserving existing engine optimization and constraints. It also allows for flexible extension.

Functions in LSCENE only operate on the world state provided to them (the list of all current objects), and it simply uses the other inputs to derive the current context. For example, "Rotation(Object1, Rotation Properties, World State)" uses the object name to apply the rotation changes to the correct object in the world state. All functions in Table 2 make use of any built-in game engine functionality, such as creating, deleting, and manipulating object properties. The LSCENE functions are responsible for changing scene properties, thus fulfilling the goal of LSCENE to act as an interface for many scene constraints. All objects referenced by a function are evaluated in the order they are presented in the JSON, allowing the user to define priority. As an example, consider the object detection problem in computer vision, where target overlap and occlusion needs to be defined. Sometimes a variable amount of overlap, or closeness to other objects is desired. The size of a target can be made to be larger than normal, which will hide other objects. After the collision test is complete with the "Position" function, the real target size is applied, resulting in a gap between all other objects and the given target. The reverse function order can be applied to force a variable percentage of overlapping.

Although LSCENE provides plenty of functions to facilitate scene building, certain problems are not practically solvable with just the use of LSCENE. Terrain with complex geometry will result in less precise placement of objects. Content databases such as Quixel, Unreal Engine Marketplace, TurboSquid, etc., will often have incompatible materials or artifacts which are not accounted for when they were exported. Any asset with "errors" could result in LSENE producing incorrect scenes from the description given to it. Yet, even outside of these few examples where a user would need to first scan the content before blindly using, LSCENE has limitations. Creating scenes with any change over time will need to introduce more robust functions to account for the complexity. Thus, temporal elements and macro oriented structures within the language need to be implemented, which we define next via LCAP in Section 3.2.

3.2 LCAP: Capture Language

As in Section 3.1, we begin with an example. **Example LCAP** 1 produces a dataset with respect to a camera moving in a hemisphere (hemi) pattern around a target object of interest. First, CO defines the camera with a set of settings, field of view and output resolution. This camera exports target O1 with a per-pixel label of 5. Two post process effects (PPMaterial's) are added to the export data layers list (default data layers in Figure 12). The camera follows a hemi movement. A range of angles and positions are generated to capture multiple sides of an object. In this case, object O1 is centered about the origin, which is the default hemi orientation.

Example LCAP 1

Camera movement in a hemisphere (hemi) pattern around a target object to make a dataset

```
LCAP = ONE(CO) ONE(P) ONE(SCS) ONE(CL)
CO = camera{csettings{fov:30,res:[2048,2048]},targets:[O1:5],PPMaterial:[EdgeBlur,CartoonStlye]}
P1 = path{create hemi:{cam radius:[1000,2000,5],obj angle:[0,360,4],cam angle:[45,45,1]}}
CL1 = collection{C1:{move:[P1:3]},O1:{when:{after:P1:1},material:MetalGold},{when:{after;P1:2},
      rotation:{range:[0,0,0],[100,100,0]}}}
```

Finally, the collection is defined, which defines the behavior of our scene over time. Symbol "move" generates a list of transforms on our camera. The notation "P1:3" generates 3 copies of the path defined with labels P1(1), P1(2), and P1(3). Symbol "when" tells when in time to apply a scene change. This function takes in a reference to an instanced path, e.g., P1(1), and triggers after the first iteration of P1 is complete, applying a different material to O1. This function can be repeated any number of times, and also triggers after P1 iteration 2 is complete to apply a random rotation to O1. Any and all LSCENE functions are callable here.

In general, LSCENE specifies a static scene. Many applications, however, require more parameter changes, e.g., scale variations, material changes, or slight positional perturbations over time. The capabilities of LSCENE therefore need to be extended. LCAP was created to solve this dilemma. Furthermore, collecting information from within a 3D scene has specifications outside of the physical and material properties of a scene. Tying the SIM scenes to their real counterparts highlights another important concept LSCENE will not account for: platform operating conditions. How is LSCENE being sampled? What are the settings on the camera doing the scene capture? What platform positions and viewing angles are needed? Accounting for these added complications results in LCAP being a higher-level language that leverages the existing functionality of LSCENE to cover a wide range of data collection requirements.

To reiterate, LSCENE lays the groundwork for manipulating data in a scene. While LCAP is responsible for changing the scene with time, setting the platform operating conditions, and managing which data is ultimately exported. Section 5 represents some of the minimum use case requirements of LCAP, e.g., scanning a single object, grid and random search, etc. More so, AI relies on large amounts of labeled data which LCAP should facilitate through several of the techniques discussed in Section 4.2, like domain randomization. These ideas (data augmentation, exportation, and variation), are some of the motivations for LCAP. Every function in Table 3 leverages object transforms and data types defined in LSCENE. Just as LSCENE is using UE functions to manipulate scene objects, LCAP is using LSCENE for the same purpose. This uses the capability already developed in LSCENE.

Another LCAP complexity is how it uses LSCENE data types to apply changes to other objects. Every time an object is referenced in a collection, LCAP calls an LSCENE function with a specified input. Even camera paths are created via this method of passing LSCENE functions into LCAP functions. First, a generic point (an object) with a 3D box representing noise in space is created. The same LSCENE function calls used for any other object with these details are used. Essentially, this acts as a blueprint for LCAP to use and generate more points. Then, LCAP uses an "increment point" function to generate N points in a grid like structure across another specified 3D space, in a similar fashion to other functions that reference objects in LSCENE. The end result is a uniform grid over a terrain (or object) with variable perturbations in the x, y and z dimensions. This is desirable, as operating conditions in many problems have noise in the real data, so it is pivotal that LSCENE/LCAP can introduce noise as well.

The concept of sets in LCAP should be discussed. These sets are composed of LSCENE objects with some number of relevant properties. In the context of "move", a path (which is also a set), is taken in and the position and rotation properties are used to generate the respective "position" and "rotation" LSCENE functions on the camera, or object. The last LCAP function that needs significant explanation is "when." Video editing tools rely on key-frames to apply some transformation at a specific time. LCAP can use frame numbers to apply changes, but this is often not desired. Rather, by naming collections and creating many instances of them, a user is able to create minimum sized blocks of time in which changes take place. References to these sets currently rely on temporal terms like "after" and "during." After will produce changes once a collection finishes, right

Table 3: LCAP Function Set

Function Name	Meaning
Target	Set the semantic segmentation ID of a given object
Camera Settings	Sets the camera FOV, image output, focal distance, and focal length.
Capture Settings	Configures UE render location output and final scene collection folders
Create Point	Defines a 3D rotation and 3D position sampled from a 3D position/rotation space
Create Hemi	Uses camera radius, object angle, and camera angle to create 3D points
Increment Points	Creates 3D points in a 3D space, given some separating extent (x,y,z)
Create Path	Defines a space and a number of points to generate within the space
Increment Point	Uses an x,y,z offset amount, to create the 3D positions of points in a path
Scene Change	Generates a set of scene transforms on an object (material, scale, etc)
Create Set	Concatenated scene changes and paths into one labeled collection
When	"When" in time to apply some scene change. Uses a set, path, or frame number.
During	During every frame tick on a set, some scene change will be applied.
After	When a set, frame number, or path is finished, apply some scene change.
PP Material	Apply a post process material to the render output

before the next collect starts. During applies a scene change every moment in time in a collection. These are not all encompassing terms, rather they are starting points for many applications, in the same way the spatial constraints proposed in LSCENE are starting points for many spatial relationships.

A more abstract language is proposed in LCAP. Once a 3D scene is expanded in the time dimension, so must the language. LCAP defines a robust collection reference system that reduces unneeded specifications when timing scene changes. Set combinations, complex way point generation functions, and data export settings all reside in LCAP. As more functions are added to LSCENE, the capabilities naturally grow in LCAP. This is the core connection of LCAP and LSCENE. Even if LCAP adds more functions, this will not change the fundamentals in LSCENE. Therefore, we have created a formal system that relies on a set of symbols, see Table 1, functions (operators), see Tables 2 and 3, and the ordering of these symbols and operators to produce meaningful 3D scenes with properly labelled data.

4. WORKING EXAMPLES

To fully appreciate LSCENE/LCAP, one must see it in action. The next sub-sections highlight two simple scenes.

4.1 Example 1: Simple Biome with a Single Object

Figure 2 shows nine random images from a data collection that could be used to train and/or evaluate (in conjunction with produced truth) an object detection and localization algorithm. Specifically, Figure 2 shows photorealistic SIM imagery for a simple scene with *a few* target objects (blue water cans) in an open grassland biome. This LSCENE specified *some* foliage objects (bushes) on a grassy terrain (different textures/materials) with sparse grass. The corresponding LCAP specified an interval of off-Nadir drone poses, operating altitudes, and times of day. A large part of this simple LSCENE example is foliage. Figure 3 is an example of the foliage section from our LSCENE JSON configuration file.

4.2 Example 2: Randomized Clutter

Section 4.1 focused on one of the most trivial scene and data collection scenarios possible, a single object type, terrain, and vegetation. In example 2, we show how a minor change to LSCENE and LCAP results in a great deal of complexity that could be used to train or stress test an AI algorithm. In Figure 4, we changed the terrain texture/material and bushes to obtain a rocky sandy biome. Basic regex statements allow for precise asset discovery from a local content database or an API like Quixel. The decision to change the biome was just so the reader could see that LSCENE maintains efficacy across different environments. Next, we specified a *massive* amount of random clutter in LSCENE. This is just a change in the range of possible objects and pointing LSCENE to a clutter folder to sample from. The reader should note that this scene is not designed with realism

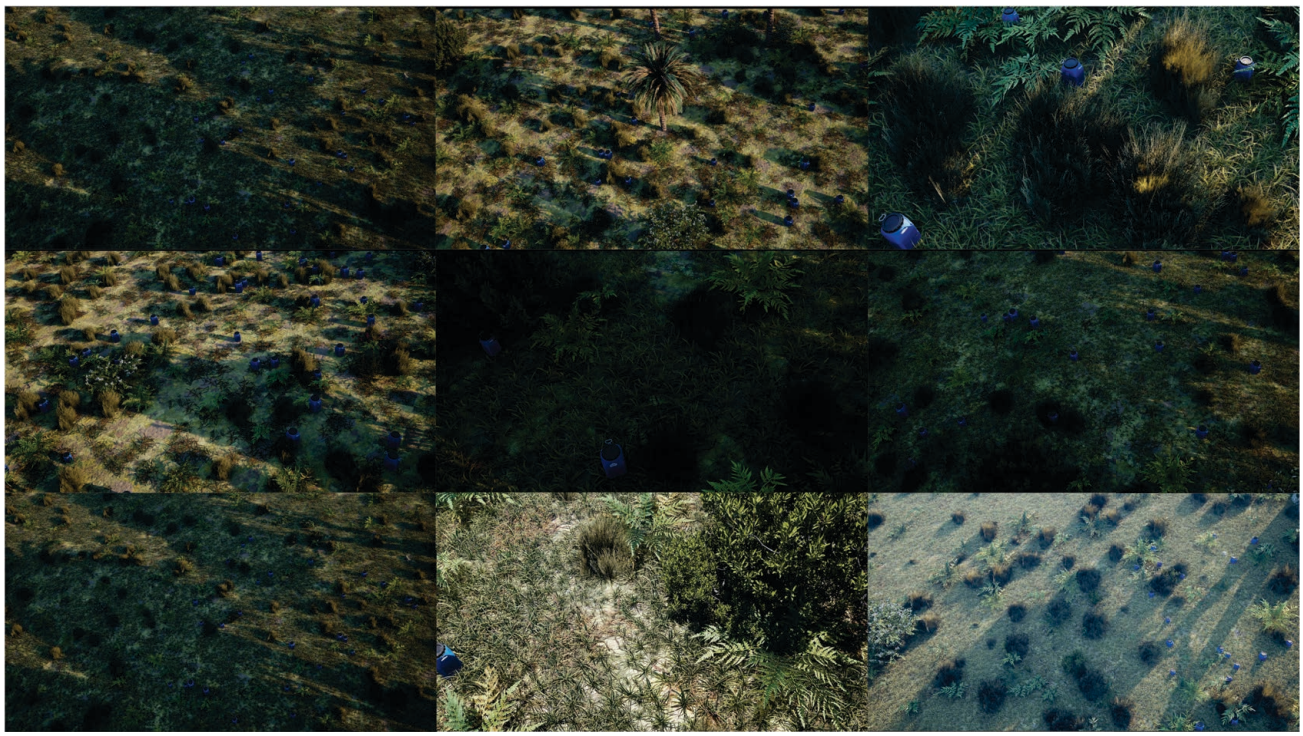


Figure 2: LSCENE Example 1 for a grassland biome and target object. See Section 4.1 for more details.

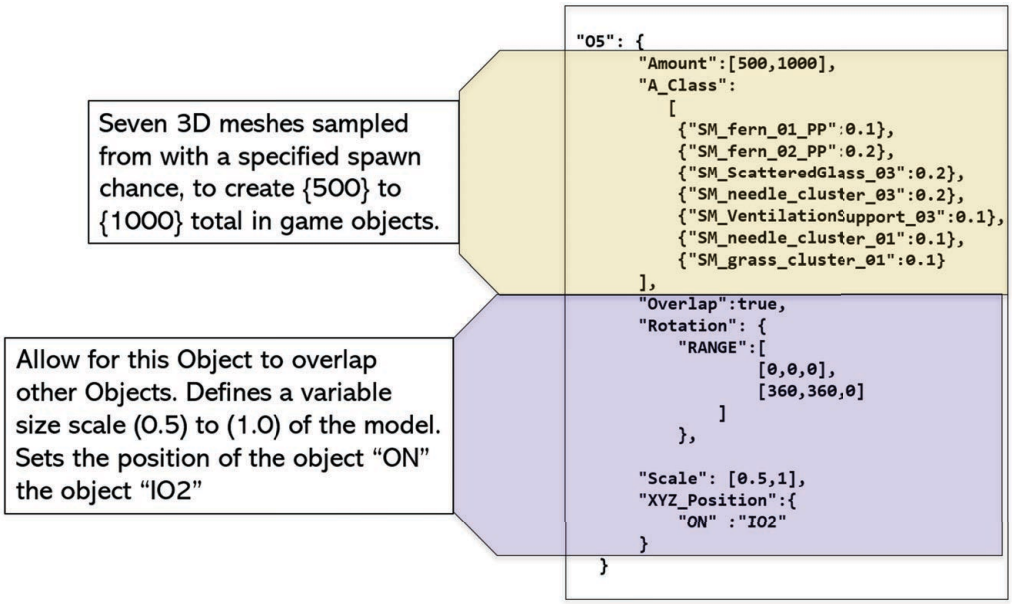


Figure 3: LSCENE JSON syntax for foliage spawner in Section 4.1.

in mind. The aim is to sample a range of clutter, nature and man made, with varying sizes, textures, colors, and other attributes. Current generation AI/ML algorithms require a great deal of data, and variety at that, to learn and ultimately generalize. A challenge is, if we generate only a single scene, we can collect a dataset. However, we would be getting a large number of looks on a relatively few number objects. This is a similar situation, and potential problem, of most real world data collections. Instead, LCAP acts like a modifier to LSCENE and in each frame, objects textures/materials and other user specified attributes, e.g., size, rotation, etc., are

Figure 4: LSCENE/LCAP Example 2 for scene with random manmade clutter. See Section 4.2 for more details.

Figure 5: LSCENE/LCAP Example 2 with randomized natural clutter. See Section 4.2 for more details.

varied. The aim is to realize a reduced size dataset with high inner scene variation via randomized sampling. AI algorithms do not always scale well with data and the *cost* (computation, storage, financial, e tc.) required to train and evaluate models on Big Data can be a serious, if not limiting, problem. LCAP was designed with these constraints in mind. These ideas are partially motivated by recent findings i n r elated t opics l ike domain randomization and sim-to-real.[26,42–45]

While Figure 4.1 is an example with suspended realism, Figure 5 is a realistic and natural looking scene. Namely, in Figure 5 we do not insert tons of arbitrary objects (tires, boxes, barrels, etc.). Instead, assets that are compatible with that biome are specified and s ampled. T he r esult i s a m ore r ealistic n ature (vs m an made) cluttered scene, but proper content labeling is required to properly grab biome specific c lutter. Q uixel (a 3D content delivery service) supports a tag based labeling scheme, with object identifiers such as desert, plants, arid, rocky, etc. This does allow for procedural asset selection as it pertains to biome compatibility.

In closing, some applications, e.g., where context does not perhaps matter, stand to benefit from extreme man made clutter insertion and randomization (Figure 4.1). However, for applications where context is important, a LSCENE like Figure 5 can be used to specify complex natural scenes and more fine detailed LSCENE language features like spatial modifiers and allowable occlusion percentage can be used to control where objects are placed and how those objects are viewed by our camera/platform. Last, another set of applications could benefit from complex environment realism and randomization, e.g., Figure 5, coupled with a limited amount of randomized man made clutter insertion. AI algorithms are not just about learning a target class. The machine must not only learn features and logic to recognize desired objects, but also features and logic that can discriminate those objects from other confusers. It remains an open research question if one of these strategies is superior and applicable to all applications. Instead, it is more likely the case that different c hallenges r equire different approaches. This is how we designed LSCENE/LCAP. It is not a single tool for a single approach. Instead, it is a single approach that can do many different things and we ultimately leave it up to the AI to help us decide what strategy is the most effective.

5. EXAMPLE AI/ML USE CASES

The last few sections focused on simple and analytically tractable LSCENE/LCAP examples. In this section, more complex scenarios from our computer vision and drone autonomy research are provided. These use cases (UC) highlight applications of the proposed tools and processes to date.

5.1 UC1: Object-Centric

Use Case 1 (UC1) is focused on generating SIM data in support of a target object. Two examples from our group include generating then inserting SIM objects into real imagery (UC1a) and full image SIM for evaluating a trained AI/ML model (UC1b). In UC1a, we previously introduced a method called altitude modulated augmentation (AMA),[46] which intelligently inserts a hemisphere (hemi) or orbit collected SIM data into real drone imagery with respect to metadata to directly train or augment a DL model in domains with low-to-no data. Figure 6 highlights the scripted movement of a camera and sun around a stationary object with varying attributes (color and texture). This is a trivial LSCENE/LCAP program. LSCENE has a single terrain, object, sky, and sun. LCAP has a direct function built in (see Table 3), Create Hemi, versus requiring the user to calculate and specify the required set of waypoints and relative poses for this movement type. Figure 7 is a full SIM output image (person on a road) and a white benign background template, which AMA uses to insert our object (person) into real imagery. In scenario UC1b, LSCENE/LCAP can be used to make a full SIM dataset in different c ontexts (environments, o cclusion l evels, e mplacement c ontexts, e tc.) f or e xplainable A I (XAI). In,[47] we performed a camera-object relative scan to mimic a low altitude drone moving around a point of interest (potential target/object). This dataset was then subjected to a DL object detector built using real world data. From here, we produce a set of graphical and linguistic XAI summarizations that highlight our data, algorithm, model successes, and shortcomings.

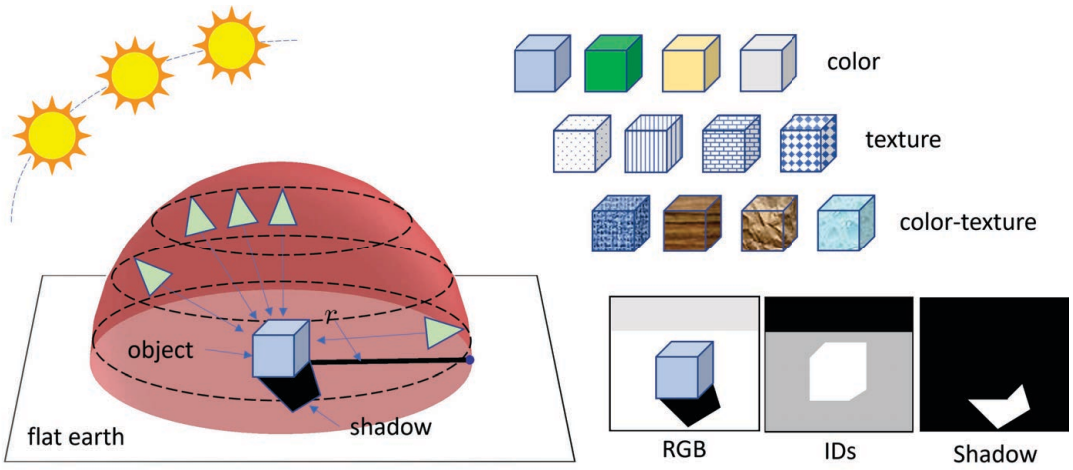


Figure 6: UC1: Hemisphere scan with corresponding data layers.

5.2 UC2: Randomization

In UC2 (Figure 8), LCAP can be used to generate a randomly sampled dataset for a desired operating context. For example, we used this philosophy to build a dataset to train and evaluate data-driven monocular vision algorithms across environments and relative viewing contexts.[48] While a grid or pre-determined waypoints can be useful, it is a pattern and therefore subject to potential bias. Grids and waypoints are also not efficient strategies for sampling large combinatorial object, environment, and platform variable spaces. In the literature, it is well documented (but not proven) that variety is equally, if perhaps not more important, than volume for training AI. When the variables that drive a task are not well understood, or when no specific experiment has

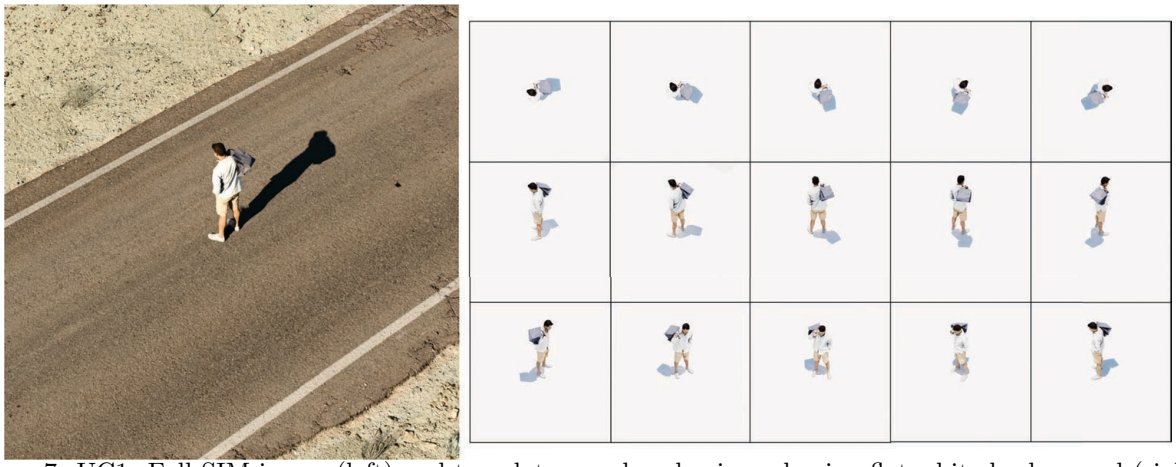


Figure 7: UC1: Full SIM image (left) and templates produced using a benign flat white background (right).

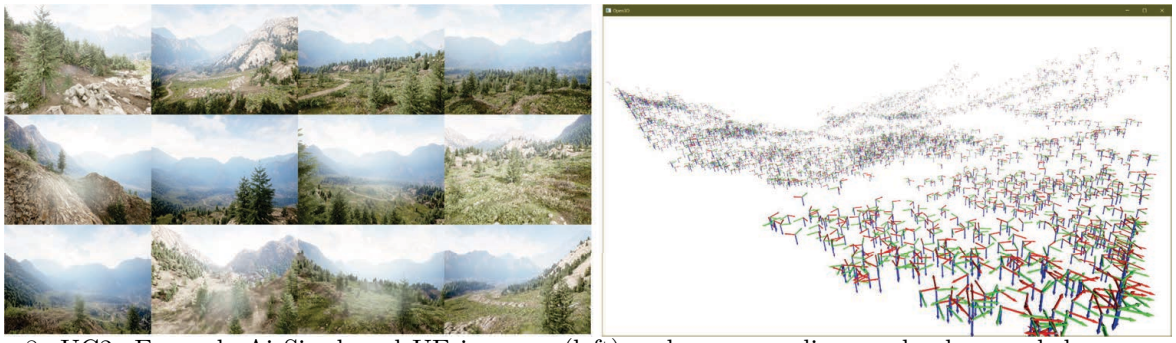


Figure 8: UC2: Example AirSim-based UE imagery (left) and corresponding randomly sampled camera poses (right) around a user specified drone operating context (desired range of altitudes and poses).

been outlined, random data collection and analysis, over repeated trials can be an effective strategy for training and helping understand a given AI. These are just a few of the many reasons why we built a random move capability into LCAP.

5.3 UC3: Motion-Centric

In Sub-Section 5.2, we argued for inclusion of randomization in LCAP. In this sub-section, we argue the opposite, inclusion of grid and custom waypoint motions. Figure 9 shows example controlled LCAP movements in support of structure from motion (SfM) algorithms. Recently, we introduced a SIM framework and workflows to help understand and characterize hand crafted and data-driven ML SfM algorithms.[49] We used the various data layers, truth, and metadata from SIM for evaluation at each image and across images to build a gold standard and ultimately focused metrics that compare and help us select algorithm parameters and ideal flight contexts (altitude, camera pose, etc.). We also used this LCAP capability to capture uncertainty in monocular depth estimation by constructing fuzzy voxel maps.[50] Figure 10 shows how this process was used to improve the quality (accuracy of free, occupied, and unknown space) of an aggregated (multi-look) voxel space for SfM. Last, we also used this LCAP capability to build datasets to train and evaluate a data-driven monocular vision algorithm for passive ranging in the context of autonomous ground vehicles11. In summary, there are many AI, computer vision, and drone autonomy scenarios that required controlled camera movements.

6. DATA, GROUND TRUTH, METADATA, AND FILE FORMATS

The previous sections focused on tools and examples. This section details specifics regarding data needed by our algorithms during training and/or evaluation. Our focus is breadth and coverage vs a single AI workflow, e.g., what's the optimal way to generate and train a detection and localization algorithm like You Only Look Once (YOLO).[51–53] Hereafter, instead of calling everything *data*, the following terminology is adopted.

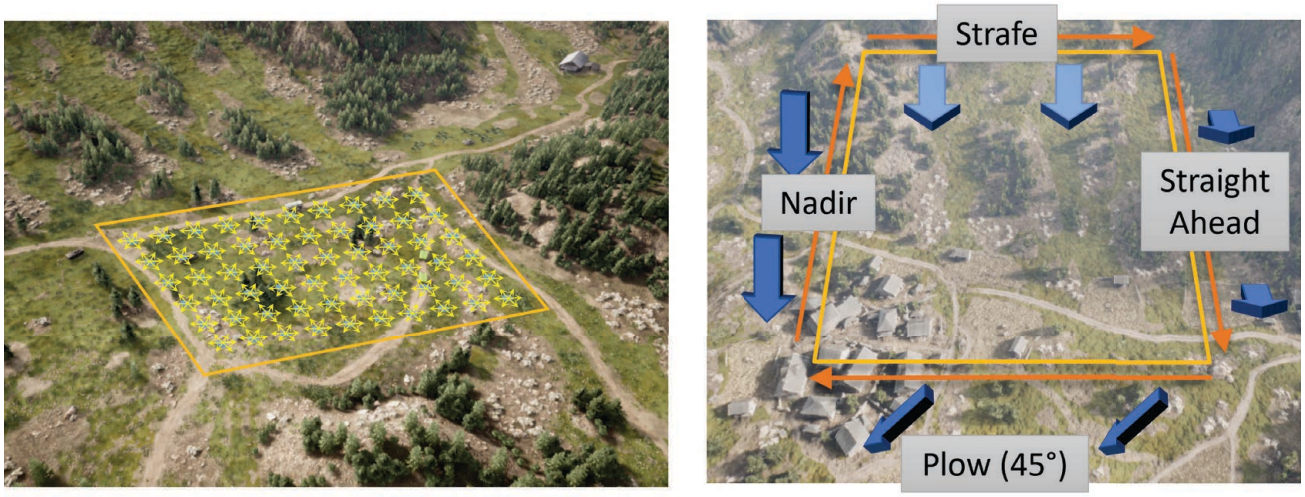


Figure 9: UC3: (left) Grid collection and (right) custom waypoints with different viewing conditions.

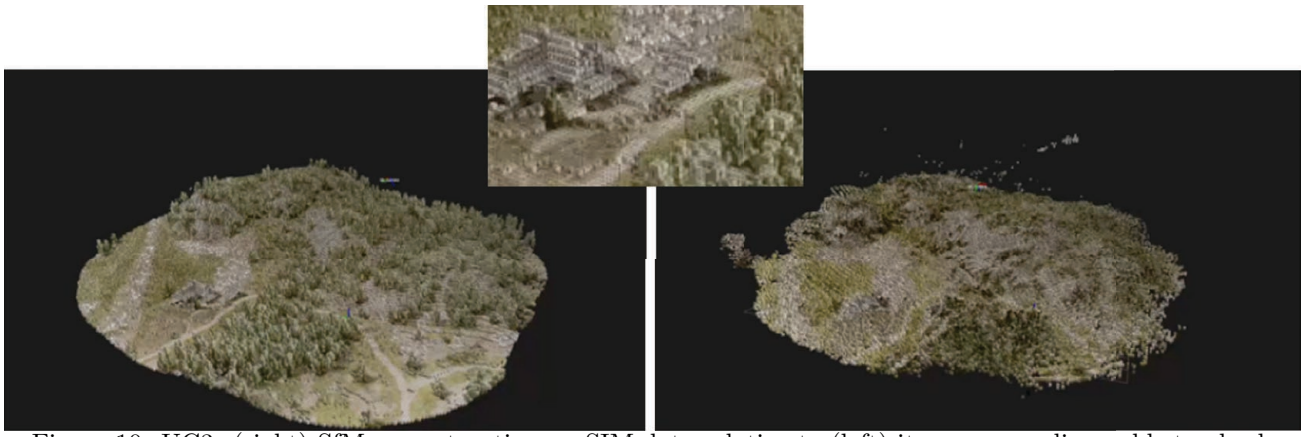


Figure 10: UC3. (right) SfM reconstruction on SIM data relative to (left) its corresponding gold standard.

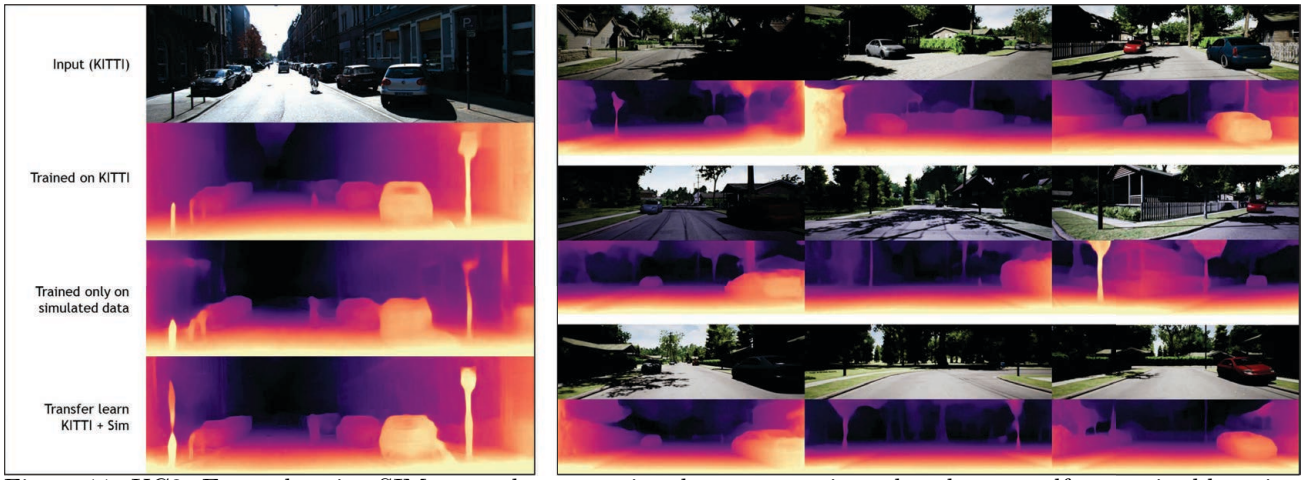


Figure 11: UC3. Example using SIM to produce a motion dataset to train and evaluate a self-supervised learning based monocular vision DL algorithm. (right) Example SIM training data with associated depth truth. (left) Example of a real, full SIM, and hybrid models evaluated on KITTI vehicle dataset.

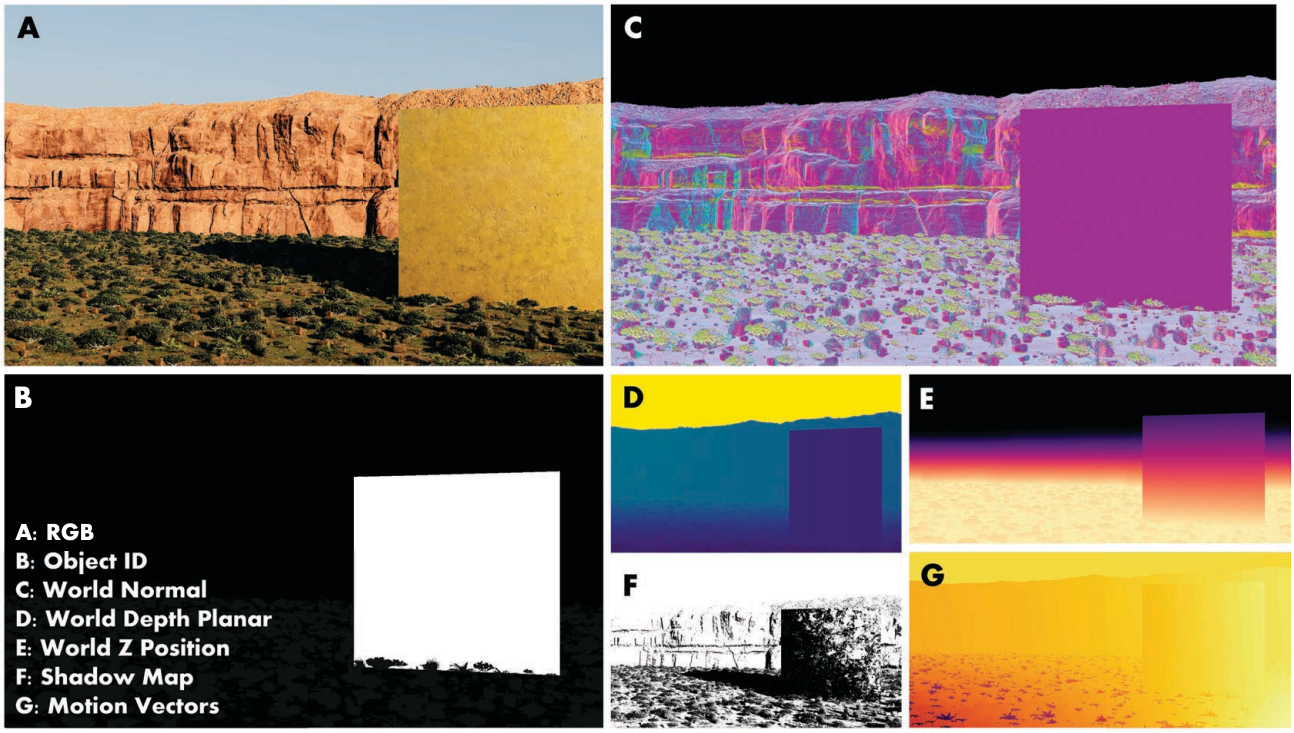


Figure 12: Example data and ground truth layers. See article for additional details.

- **data**: this is typically raw or normalized numeric values from a passive or active sensor that the AI/ML is processing and using for decision making. Examples include imagery from a visual spectrum (aka RGB) camera, raw 14 bit normalized (e.g., mean absolute difference (MAD)) infrared imagery, Velodyne LiDAR puck position and/or magnitude values, etc.
- **ground truth**: information about the objects, environment, events, context, and other factors that define and drive our task. Unlike data, this information is not explicitly available to an AI at test/run time. Examples include object identifiers (e.g., class labels), bounding volumes (e.g., maxis aligned bounding boxes (AABBs)), emplacement photos (additional imagery that helps a user better visually understand the scene and/or objects), relative or global reference frames and coordinate systems (UTM, WGS84, MGRS, etc.), depth (relative or global reference frame), 3D surface/point cloud/voxel space, surface normal's, object positions and poses, temperature, time of day, etc. In the real-world, most ground truth is assumed fixed/static in time. However, simulators have the potential of storing such information at the expense of disk space and complexity at each moment in time (frame) if/when desired.
- **metadata**: numeric values measured at usually different sampling rates by the platform and sensors. Examples include position, pose (roll, pitch, yaw), temperature, light sensor that measures irradiance, camera settings like white balance, gain, etc. As metadata is measured, it has associated error.

As the reader can see, these terms do not have clear boundaries. Herein, data is the primary input (e.g., image) to our algorithms, ground truth is not available at run time, and metadata is measured data that helps an AI/ML better understand and/or process its data. The following sub sections detail our data generation, layers, file formats, and organization. Figure 12 are examples of what data we generate and store.

6.1 Data Layers

This section focuses on information that an AI has and can use for online decision making. Specifically, we restrict the focus of our current article to RGB imagery. Figure 13 is an example of the photorealism possible

with UE5. If the reader wishes to simulate multispectral data in UE, then the Infinite Studio plugin can be used[54*]. First, there is no single accepted way of generating RGB SIM data. Example approaches involve traditional and efficient rasterization, ray tracing, FDTD, etc. Furthermore, each of these are approximations and what parameters should we use? This is a challenge, all this freedom, in part because we still do not really understand what "type" of data AI/ML needs. Does the RGB imagery need to be 100% accurate? What level of realism is acceptable? Furthermore, are there any "fingerprints", things present in SIM that are not in real data, that a machine might pick up on, create unwanted biases, impact our SIM-vs-real world discoveries, or ultimately, prohibit our ability to transfer or use SIM data/models to the real world?

Figure 13: Example RGB urban and rural photorealistic images from UE5.

The following is our current RGB workflow. AirSim[56] was not used because while it is useful for simulation of an aerial platform, it results in less than photorealistic RGB imagery. Instead, we use UEs Movie Render Queue (MRQ), which is used for offline non-gaming processes like film, for rendering with deferred passes to make RGB and other layers. We developed a C++ Plugin and talk to the UE MRQ library. RGB imagery is stored in the EXR file format. EXR is an open-source high-dynamic-range image file format created by Industrial Light and Magic. Specifically, we use OpenEXR. Beyond storing higher bit depths (beyond 8) and addressing compression (e.g., uncompressed), OpenEXR allows for arbitrary and custom channels like specular, diffuse, alpha, normals, and various other types like we discussed in the ground truth sub-section. OpenEXR also allows for storage of sensor or rendering metadata, which is often needed to understand the generation and/or address calibration. Our approach involves letting the producer (e.g., the MRQ) generate its EXR data however it deems fit. At a minimum, we require red, green, and blue layers and fields specifying the width, height, and number of channels. In addition to EXR RGB imagery, a JSON is produced with custom description of how the data was produced. We discovered that many of our documented details were not known ahead of time. This is one part an evolving set of UE tools with frequent updates, and another part the fundamental nature of open research[†]. For example, a user might specify in our JSON that the UE5.2 engine was used with Lumen, along with any other relevant miscellaneous markups they see fit (e.g., lux settings, volumetric fog, screen space ambient occlusion settings, etc.). In the majority of our current AI/ML workflows, we use the following settings. We disable per image antialiasing (AA), disable temporal multisample effects, enable spatial multisampling, and Game Overrides[‡].

6.2 Metadata

Herein, metadata is information observed at runtime by the platform and/or camera. Ground truth records real (true) values. We store metadata in a JSON and they are measured observations. If an engine and simulation is perfect, then metadata and ground truth are equal in our approach. However, libraries like AirSim exist to model

[*]Simulating multispectral data does come at an extra added expense of extended material and physics specification and simulation, which can add significantly more processing and offline content modeling time and expense. In comparison, free resources like Quixel[55] exist for high quality real world scanned RGB materials and 3D models.

[†]Albert Einstein's famous quote, "If we knew what it was we were doing, it would not be called research."

[‡]This include Cinematic Quality Settings, texture streaming, level of detail (LOD) zero, disable hierarchical LOD, high quality shadows, overide distance scale, and extend view distance scale

drones and real-world flight. Furthermore, some SIM investigations intentionally introduce controlled error at different levels to assess its impact on AI. In any respect, metadata is simply addressed herein as a single JSON file with true or perturbed measurements.

6.3 Organization

To accommodate a range of end users and AI applications, we adopted the following strategy. Each data and truth layer is stored in as an EXR file in a separate folder with a simple name, e.g., "rgb", "worlddepth", etc. After attempting a number of complicated strategies, we discovered that less is more. More sophisticated ideas unfortunately required frequent changes or they conflicted with end users and their niche applications. Our flat organization worked best and the accompanying JSON files allow for customization. Overall, we strive for generality and minimal user constraints. Any consumer, e.g., an object detection and localization algorithm, can easily parse this flat structure, look for keywords in the folders like "rgb", and parse the JSON to determine attributes and if they want to use it or not for training and/or evaluation.

6.4 Ground Truth Layers

This sub-section is perhaps one of the most compelling reason to couple AI and SIM. Its incredibly hard, if not impossible, to get accurate truth in the real world. The act of having and exposing truth to AI is a game changer. Truth can be used in a number of ways from simply scaling up AI (making more data) to improved supervised learning (vs resorting to unsupervised, self-supervised, or other paradigms) and deep contextual evaluation. In anticipation of needing data sets for various use cases, we always output the following (see Figure 12) layers: object IDs, depth, shadows, surface normals, motion vectors, and world locations.

Object IDs: Object IDs are a per-pixel class or instance coded layer. Typically, this is a monochromatic integer valued image where the closest object ID is recorded[§]. Modern engines like UE typically tackle object IDs in an efficient fashion via a stencil buffer, which allows for $2^8 = 256$ unique IDs[¶]. The reader can manually extend the number of supported IDs by using a strategy like a custom integer or floating point render buffer with an ad hoc encoding. While 256 IDs may be sufficient for environments with a small number of object classes, dense and/or complex modern scenes like Epic's photo realistic Matrix City has hundreds of classes and thousands to millions of object instances. We do not store instance IDs by default as the resultant floating point-based file sizes add up quickly and most workflows do not take advantage of instance IDs. The reader should also note that a large percentage of pixels in an image are "mixed", i.e., the volume in space covered by a pixel includes multiple objects. Although this is true, its of relatively low value to a game engine. As such, engines only store a single or mixed (e.g., average in the case of anti-aliasing or spatial or temporal up sampling) value versus a list of all objects[‖]. In summary, we currently store integer-valued object class IDs with a single (not mixed) per-pixel value. We have not found nor logically been able to identify an AI/ML application that welcomes averaged object IDs. Such an operation results in IDs that belong to another class[**].

Depth: We record depth in world space units. Many engines are capable of producing different types of depth. For example, UE supports planar (all points that are plane-parallel to the camera have same depth) and perspective (depth from camera using a projection ray that hits that pixel). In UE, depth is recorded in cm or meters,

[§]The UE MRQ supports exporting object IDs to the EXR file format.[57] The reader can pick full, material, actor, actor with hierarchy, folder or layer. For example, selecting actor tracks the location of each pixel for each actor in the image. While useful, this can result in extremely large file sizes. For example, an image with resolution 2,048 squared results in approximately a one terabyte EXR file.

[¶]We used deferred rendering and included depth into UEs post-processing stencil buffer material to address occlusion. Otherwise, IDs of occluded pixels are recorded in the ID map (see Figure 12). This is sometimes a debated topic. For example, consider the case of a car occluded by a tree. Should an object ID map include each pixel for the car, occluded or not? This is very application dependent. We stick with the typical convention of recording just closest non occluded object IDs. If the reader wants all pixels, occluded and not, then the MRQ and its object ID exporting in EXR format can be used versus a stencil buffer.

[‖]See our 2023 SPIE paper about what is a pixel, does simulation really have truth, and how can biases be mitigated in training and evaluation via recording and using multiple samples.[22]

[**]For example, let object 1 be ID 10 and object 2 be ID 2 and assume a mixed pixel with half object 1 and half object 2. The averaged object ID would be 6, which is not object 1 or 2.

depending on how the user configured their project. Figure 12 shows an example depth map, which has been color coded between min and max value for display purposes. Details of interest include the following. We store depth as a double for precision. A single depth value is recorded per pixel, versus a combined (e.g., averaged) value. If mixed pixel depths are averaged then "3d point cloud trails" result. This is unacceptable for the AI/ML applications we have explored. This is why we use a single depth from one of the objects vs an incorrectly assigned number somewhere in free space between the objects. For example, consider a close tree and mountains at a far off distance. Depth pixels on the edge of the tree would have averaged points at the half way point of the mountain and tree. The reader can see our 2023 SPIE article for how to improve this process by recording and using a pixel bundle for unbiased training and evaluation.[22] However, such a solution is a tradeoff of accuracy versus storage and compute.

Shadow: Shadows are often discussed in terms of umbra, penumbra and antumbra. However, this information is not of general use to game engines and it can be expensive, is possible at all, to extract. In our experience, users are typically interested in relatively simple shadow information, e.g., a binary indicator if an object is in shadow or not (independent of a particular light source). As such, we have adopted a computationally efficient and extremely simple strategy where the difference, or a ratio, is generated between the base render pass and the final rendered image. Figure 12 shows an example. The result is a single floating point shadow image, where 0 is full shadow and 1 is no shadow. It is important to note that this method has limitations. For example, dark (e.g., black) surfaces will obviously not be addressed appropriately. Furthermore, the intensity of lighting is not well accounted for, e.g., this value changes with respect to lux. This is also an inclusive calculation, e.g., it considers rendering approximations like ambient occlusion mapping and parallax occlusion mapping. While simple, this has been useful in our research with respect to simulating objects and inserting them into real drone imagery.[46] We scan an object on a white background. Pixels on the ground, identified using object IDs, that are darkened, i.e., are not full white, are locations where our object is casting a shadow (on flat earth). If detailed shadow information is important to the reader, a strategy beyond what we have outlined here is necessary.

Normals: Many applications are concerned with estimating and using 3D information like points, voxels, and/or surfaces. Offline, normals have use in tasks like evaluating and understanding 3D AI/ML algorithms, e.g., SfM, MVS, etc. In real time, a number of recent ML/DL models have been proposed to estimate normals from a single image for tasks like object pose estimation and/or detection. Regardless, herein we store unit length surface normals in the engines world reference space[††]. Figure 12 shows an example three channel world space normal image, where red encodes x, green is y, and blue is z. As discussed in the previous sections, we store a single normal per pixel, not an averaged value. While an averaged normal along an object seem appropriate, averaged normals along the edges of objects are not. Conversely, a bundle can be computed and stored.[22]

Motion: In many applications, e.g., object detection, tracking, and depth estimation, optical flow and motion estimation is important, with respect to training and real-time decision making. Herein, we make use of the UE motion vector deferred rendering pass.[57] Specifically, motion vectors are stored per pixel in $[0, 1]$ for x and y, where 0.5, 0.5 is no motion. UE stores motion vectors normalized to the entire screen.

JSON: The past few paragraphs focused on layers as images. Our group produces a single JSON per data collection. This JSON has an entry for each image and we store information like 3D OBBs, 2D screen space AABBs, time, position, and orientation per object of interest. This information is crucial for algorithms like object detection and localization. Specifically, we store this data in a simple JSON file because there are various specific file formats used in practice to train and score algorithms. For example, YOLO uses Darknet TXT and Computer Vision Annotation Tool (CVAT) has its own file format for labeling and viewing results. The point is, we store a single simple JSON and write translators to whatever utility it needs to interface with.

7. CONCLUSION AND FUTURE WORK

Modern AI is unfortunately heavily dependent on large quantities of high quality labeled data. In this article, we discussed our design and implementation of two procedural languages, LSCENE and LCAP, for synthetic data generation in the context of AI and low altitude aerial drone applications. Analytically tractable examples

[††]The default units in UE are cm, positive y is right, positive x is forward, and positive z is up.

were given and more complicated use cases were discussed. We also detailed our data, ground truth, metadata, formats, and storage processes.

In future work, implementations of more robust LSCENE and LCAP functions will be explored. Currently, the spatial constraints in LSCENE have a precision only up to a 3D bounding box. Extending the core functionality of each function presented in LSCENE/LCAP, will hopefully result in more transparent and intricate scenes that better reflect the description of the language. Once the overall competency of LSCENE/LCAP is finalized, an AI/ML algorithm would be able to generate scenes automatically. For every scene auto generated, a neural network, genetic algorithms, or other learning method could help produce scenes that better reflect a desired goal. These goals can vary in nature, but a basic example would be for target detection, where data is produced to minimize a classifiers error rate. A formalized logic system in tandem with some system that builds truth statements from a SIM scene, could also find use in building and describing complex scenes. UE has a header tool which automatically creates a reflection system of all of the functions and types in their engine. A formal system can then query this reflection tool about possible functions, query for the parameters and parameter types, and then call these found functions with good inputs that lead to the same minimization problem as before. Our LSCENE/LCAP framework already reduces the search space a great degree, but to advance the language more, fundamental questions about the engine are asked through this reflection system to introduce new terms into the more abstract LSCENE/LCAP language.

Last, the current article is primarily documentation and an open discussion about how to formally describe, sample, and produce SIM data in support of AI/ML. In future work, we will take the next step and explore automated closed loop language learning for a specific real world task like EHD. While two objectives are the efficient production of SIM data at scale and training high quality AI/ML models, another real-world objective is opening up hood and analyzing the learned language (variables and rules) for domain knowledge discovery. Before this feat can be accomplished, however, we must first research and develop the specification and production steps outlined in this article.

APPENDIX A.

In this appendix, we summarize two algorithms used by LSCENE and LCAP. Any function in Table 2 has the potential to be probabilistic or deterministic. Inherent to each function, however, is the nature of sampling from many possible solutions. Essentially, every function is called with some lower and higher bound for what the function output can be. In this appendix, we can examine Algorithm 1 to understand how each function "space" is confined or expanded. Any function which can take in a range of inputs, will sample the inputs in the same manner. First, 0 to N inputs are provided for a function. Next, f(inputs) generates the potential solutions, such as in Algorithm 1, creating the minimum space to satisfy spatial arguments. Once a solution set is found, a uniform random sampling is applied to the set to select a single output. For N objects, the random sampling would occur N times on the generated solution set.

Algorithm 1 LSCENE Position Function

AO = A list of all world objects given a the symbol of that object, terrain, skybox, etc.
AC = A list of all spatial constraints to apply to the objects potential spawn locations.
while AO **do** ▷ Iterate over all objects defined under the given symbol
  $CB \leftarrow$ max sized 3D bounding box ▷ Start potential spawn location large
  while AC **do**
    $CB \leftarrow AC_N(CB)$ ▷ Shrink the 3D bounding box to fit the constraint
  end while
  AO_N.Position←Random 3D Point in CB ▷ After all spatial constraints applied, generate random point
  if AO_N is Overlapping **then**
    Check AO_N constraints vs Overlapping Object
    Delete, hide, move, or allow overlapping objects ▷ Based on constraints of overlapping and current obj.
  end if
end while

Last, a simple parsing algorithm, see Algorithm 2, is defined and used to convert the JSON into C++ data types and function calls.

Algorithm 2 Parsing JSON into UE Data and C++ Functions

FL = A Map with string keys, and C++ U5 Function pointers as values (A reflection system to give C++ Functions a string reference name).
FC = A list of referenced Function pointers by the JSON
VC = A list of referenced variables by the JSON
LS = A List of the symbols in the language
JSON = Current LSCEN JSON object
CS = Current symbol context
Call Function: $f(JSON_{0,0}, null)$

```
while Node_{branch,0} ≠ null do                          ▷ While more tree branches
    depth ← 0
    while Node_{branch,depth} ≠ null do                  ▷ While more tree nodes at branch N
        if CS ∈ LS then                                  ▷ Add Symbol Context to every function
            VC.Insert(CS)
        end if
        if Node_{branch,depth}.Key ∈ LS then
            f(Node_{branch,depth+1}.Value, Node_{branch,depth}.Key)   ▷ Call f Again with New Symbol Context
            branch ← branch + 1                          ▷ Get other symbols on the same depth
        else if Node_{branch,depth}.Key ∈ FL then        ▷ Add current key to Function list if it is Function List
            Func ← FL.At(Node_{branch,depth}.Key)
            FC.Insert(Func)
            depth ← depth + 1
        else
            VC.Insert(Node_{branch,depth}.Key)           ▷ else, found a variable name or value
            depth ← depth + 1
        end if
    end while
    branch ← branch + 1
end while
```

REFERENCES

[1] "Gazebo." http://gazebosim.org/. (Accessed: 25 March 2023).
[2] "Robot Operating System (ROS)." https://ros.org. (Accessed: 25 March 2023).
[3] Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M., "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in [*ICCV*], (2021).
[4] Liang, J., Makoviychuk, V., Handa, A., Chentanez, N., Macklin, M., and Fox, D., "Gpu-accelerated robotic simulation for distributed reinforcement learning," (2018).
[5] "Unreal Engine." https://www.unrealengine.com/. (Accessed: 1 March 2023).
[6] "Unity." https://unity.com/. (Accessed: 25 March 2023).
[7] "COMSOL." https://www.comsol.com/. (Accessed: 1 March 2023).
[8] Oskooi, A. F., Roundy, D., Ibanescu, M., Bermel, P., Joannopoulos, J., and Johnson, S. G., "Meep: A flexible free-software package for electromagnetic simulations by the fdtd method," *Computer Physics Communications* **181**(3), 687–702 (2010).
[9] 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).

[10] Durst, P., Bethel, C., and Anderson, D., "A historical review of the development of verification and validation theories for simulation models," *International Journal of Modeling, Simulation, and Scientific Computing* **08** (01 2017).
[11] Nikolenko, S. I., "Synthetic data for deep learning," (2019).
[12] Lai, K.-T., Lin, C.-C., Kang, C.-Y., Liao, M.-E., and Chen, M.-S., "Vivid: Virtual environment for visual deep learning," in [*Proceedings of the 26th ACM International Conference on Multimedia*], *MM '18*, 1356–1359, Association for Computing Machinery, New York, NY, USA (2018).
[13] Kishore, A., Choe, T. E., Kwon, J., Park, M., Hao, P., and Mittel, A., "Synthetic data generation using imitation training," in [*Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*], 3078–3086 (October 2021).
[14] Wang, H., Wang, Q., Yang, F., Zhang, W., and Zuo, W., "Data augmentation for object detection via progressive and selective instance-switching," (2019).
[15] Georgakis, G., Mousavian, A., Berg, A. C., and Kosecka, J., "Synthesizing training data for object detection in indoor scenes," (2017).
[16] Haltakov, V., Unger, C., and Ilic, S., "Framework for generation of synthetic ground truth data for driver assistance applications," in [*Pattern Recognition*], Weickert, J., Hein, M., and Schiele, B., eds., 323–332, Springer Berlin Heidelberg, Berlin, Heidelberg (2013).
[17] Mayer, N., Ilg, E., Fischer, P., Hazirbas, C., Cremers, D., Dosovitskiy, A., and Brox, T., "What makes good synthetic training data for learning disparity and optical flow estimation?," *International Journal of Computer Vision* **126**, 942–960 (apr 2018).
[18] Saleh, F. S., Aliakbarian, M. S., Salzmann, M., Petersson, L., and Alvarez, J. M., "Effective use of synthetic data for urban scene semantic segmentation," (2018).
[19] Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R., "Learning from simulated and unsupervised images through adversarial training," (2017).
[20] McCormac, J., Handa, A., Leutenegger, S., and Davison, A. J., "Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth," (2017).
[21] Richter, S. R., Vineet, V., Roth, S., and Koltun, V., "Playing for data: Ground truth from computer games," (2016).
[22] Buck, A., Derek Anderson, J. F., Kerley, J., and Palaniappan, K., "Ignorance is bliss: flawed assumptions in simulated ground truth," in [*SPIE*], (2023).
[23] Gaidon, A., Wang, Q., Cabon, Y., and Vig, E., "Virtual worlds as proxy for multi-object tracking analysis," (2016).
[24] Marelli, D., Bianco, S., and Ciocca, G., "Ivl-synthsfm-v2: A synthetic dataset with exact ground truth for the evaluation of 3d reconstruction pipelines," *Data in brief* **29**, 105041 (April 2020).
[25] Degol, J., Lee, J. Y., Kataria, R., Yuan, D., Bretl, T., and Hoiem, D., "Feats: Synthetic feature tracks for structure from motion evaluation," in [*2018 International Conference on 3D Vision (3DV)*], 352–361 (2018).
[26] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., "Domain randomization for transferring deep neural networks from simulation to the real world," (2017).
[27] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S., "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," (2018).
[28] Raghavan, N., Chakravarty, P., and Shrivastava, S., "Sim2real for self-supervised monocular depth and segmentation," (2020).
[29] Behl, H. S., Baydin, A. G., Gal, R., Torr, P. H. S., and Vineet, V., "Autosimulate: (quickly) learning synthetic data generation," (2020).
[30] Ruiz, N., Schulter, S., and Chandraker, M., "Learning to simulate," (2019).
[31] Jiang, Y., Zhang, H., Zhang, J., Wang, Y., Lin, Z., Sunkavalli, K., Chen, S., Amirghodsi, S., Kong, S., and Wang, Z., "Ssh: A self-supervised framework for image harmonization," (2021).
[32] Cong, W., Niu, L., Zhang, J., Liang, J., and Zhang, L., "Bargainnet: Background-guided domain translation for image harmonization," (2021).

[33] Dwibedi, D., Misra, I., and Hebert, M., "Cut, paste and learn: Surprisingly easy synthesis for instance detection," (2017).
[34] Niu, L., Cong, W., Liu, L., Hong, Y., Zhang, B., Liang, J., and Zhang, L., "Making images real again: A comprehensive survey on deep image composition," (2022).
[35] 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).
[36] Church, E. M. and Semwal, S., "Simulating trees using fractals and l-systems," (2006).
[37] Prusinkiewicz, P., Hanan, J., Hammel, M., and Mech, R., "L-systems: from the theory to visual models of plants," (2001).
[38] Olsen, J., "Realtime procedural terrain generation," (2004).
[39] Kelly, G. and McCabe, H., "A survey of procedural techniques for city generation," **14** (01 2006).
[40] Booth, M., "The ai systems of left 4 dead," in [*Artificial Intelligence and Interactive Digital Entertainment Conference*], Stanford University (2009).
[41] Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., McAleese, N., Bradley-Schmieg, N., Wong, N., Porcel, N., Raileanu, R., Hughes-Fitt, S., Dalibard, V., and Czarnecki, W. M., "Open-ended learning leads to generally capable agents," (2021).
[42] Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D., "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," 8973–8979 (05 2019).
[43] Valtchev, S. and Wu, J., "Domain randomization for neural network classification," (09 2020).
[44] Chen, X., Hu, J., Jin, C., Li, L., and Wang, L., "Understanding domain randomization for sim-to-real transfer," (2022).
[45] Truong, J., Rudolph, M., Yokoyama, N. H., Chernova, S., Batra, D., and Rai, A., "Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation," in [*6th Annual Conference on Robot Learning*], (2022).
[46] 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 [*SPIE*], (2021).
[47] Alvey, B. J., Anderson, D. T., Yang, C., Buck, A., Keller, J. M., Yasuda, K. E., and Ryan, H. A., "Characterization of Deep Learning-Based Aerial Explosive Hazard Detection using Simulated Data," in [*2021 IEEE Symposium Series on Computational Intelligence (SSCI)*], 1–8 (Dec. 2021).
[48] Buck, A., Deardorff, M., Murray, B., Anderson, D., Keller, J., Popescu, M., Ho, D., and Scott, G., "Estimating depth from a single infrared image," in [*MSS*], (2023).
[49] Akers, J., Buck, A., Camaioni, R., Anderson, D., Luke, R., Keller, J., Deardorff, M., and Alvey, B., "Simulated gold standard for quantitative evaluation of monocular vision algorithms," in [*SPIE*], (2023).
[50] Buck, A., Anderson, D., Camaioni, R., Akers, J., Luke, R., and Keller, J., "Capturing uncertainty in monocular depth estimation: Towards fuzzy voxel maps," in [*FUZZ-IEEE*], (2023).
[51] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You only look once: Unified, real-time object detection," (2015).
[52] Redmon, J. and Farhadi, A., "Yolo9000: Better, faster, stronger," (2016).
[53] Redmon, J. and Farhadi, A., "Yolov3: An incremental improvement," (2018).
[54] "Infinite Studio." https://infinitestudio.software/. (Accessed: 25 March 2023).
[55] "Quixel." https://quixel.com/. (Accessed: 25 March 2023).
[56] "AirSim." https://github.com/microsoft/AirSim. (Accessed: 25 March 2023).
[57] "UE MRQ Render Passes." https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/Sequencer/HighQualityMediaExport/RenderSettings/RenderPasses/. (Accessed: 25 March 2023).

Chapter 4

Title: Simulated gold-standard for quantitative evaluation of monocular vision algorithms

Venue: SPIE Defense + Commercial Sensing

Date Published: June 15, 2023

PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Simulated gold-standard for quantitative evaluation of monocular vision algorithms

Jack Akers, Andrew Buck, Raub Camaioni, Derek Anderson, Robert Luke, et al.

Jack Akers, Andrew Buck, Raub Camaioni, Derek T. Anderson, Robert H. Luke III, James M. Keller, Matthew Deardorff, Brendan Alvey, "Simulated gold-standard for quantitative evaluation of monocular vision algorithms," Proc. SPIE 12525, Geospatial Informatics XIII
, 125250A (15 June 2023); doi: 10.1117/12.2657567

SPIE.

Event: SPIE Defense + Commercial Sensing, 2023, Orlando, Florida, United States

Simulated Gold-Standard for Quantitative Evaluation of Monocular Vision Algorithms

Jack Akers[a], Andrew Buck[a], Raub Camaioni[b], Derek T. Anderson[a], Robert H. Luke III[b], James M. Keller[a], Matthew Deardorff[b], and Brendan Alvey[a]

[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

In the physical universe, truth for computer vision (CV) is impractical if not impossible to obtain. As a result, the CV community has resorted to qualitative practices and sub-optimal quantitative measures. This is problematic because it limits our ability to train, evaluate, and ultimately understand algorithms such as single image depth estimation (SIDE) and structure from motion (SfM). How good are these algorithms, individually and relatively, and where do they break? Herein, we discuss that while truth evades both the real and simulated (SIM) universes, a SIM CV gold-standard can be achieved. We outline an extensible SIM framework and data collection workflow using Unreal Engine with the Robot Operating System (ROS) for three dimensional mapping on low altitude aerial vehicles. Furthermore, voxel-based mapping measures from algorithm output to a SIM gold-standard are discussed. The proposed metrics are demonstrated by analyzing performance across changes in platform context. Ultimately, the current article is a step towards an improved process for comparing algorithms, evaluating their strengths and weaknesses, and automating algorithm design.

Keywords: point cloud, voxel, structure from motion, monocular depth estimation, single image depth estimation, unreal engine, simulation, ground-truth, gold-standard, XAI, evaluation

1. INTRODUCTION

The field of 3D reconstruction (3DR) is decades old with countless groundbreaking innovations from academic and commercial parties. 3DR enables many applications like self driving cars, unmanned aerial vehicles (UAV), film, augmented reality, earth observation, security and defense, agriculture, robotics, computer vision (CV), archaeology, and beyond. However, the physical universe is a 3DR bottleneck. Accurate and dense truth at scale is impractical if not impossible to obtain. How do we perform objective, quantitative analysis for tasks like fine-tuning, training, and/or evaluating 3DR algorithms? Where does an algorithm work? Where does it fail? Is one algorithm better than another, by how much and in what context? A lack of real world truth significantly hinders our ability to address these fundamental and essential questions.

Herein, we use simulation (SIM) to create datasets in a controlled environment with dense and highly accurate ground-truth. Different metrics are discussed for quantitative evaluation of a 3DR algorithm relative to truth in a static scene. Figure 1 illustrates how current article contributions fit into our longer-term research goals. These operations enable a human-in/on/over-the-loop (HITL/HOTL) to iteratively explore and fine-tune 3DR algorithms in different platform (e.g., drone) contexts. These operations are also needed for fully automated data-driven tasks like closed-loop optimization.

The remainder of the article is structured as follows. In Section 2, related work is summarized. Next, in Section 3 we discuss our SIM framework and workflow, Section 4 outlines truth, and Section 5 presents different voxel-based evaluation metrics. Last, Section 6 has two examples for a real-time 3DR algorithm that show the proposed ideas in action. The reader can refer to Table 1 for notation and acronyms.

Send correspondence to Jack Akers
E-mail: jdapm8@missouri.edu

Geospatial Informatics XIII, edited by Kannappan Palaniappan, Gunasekaran Seetharaman, Joshua D. Harguess, Proc. of SPIE Vol. 12525, 125250A · © 2023 SPIE
0277-786X · doi: 10.1117/12.2657567

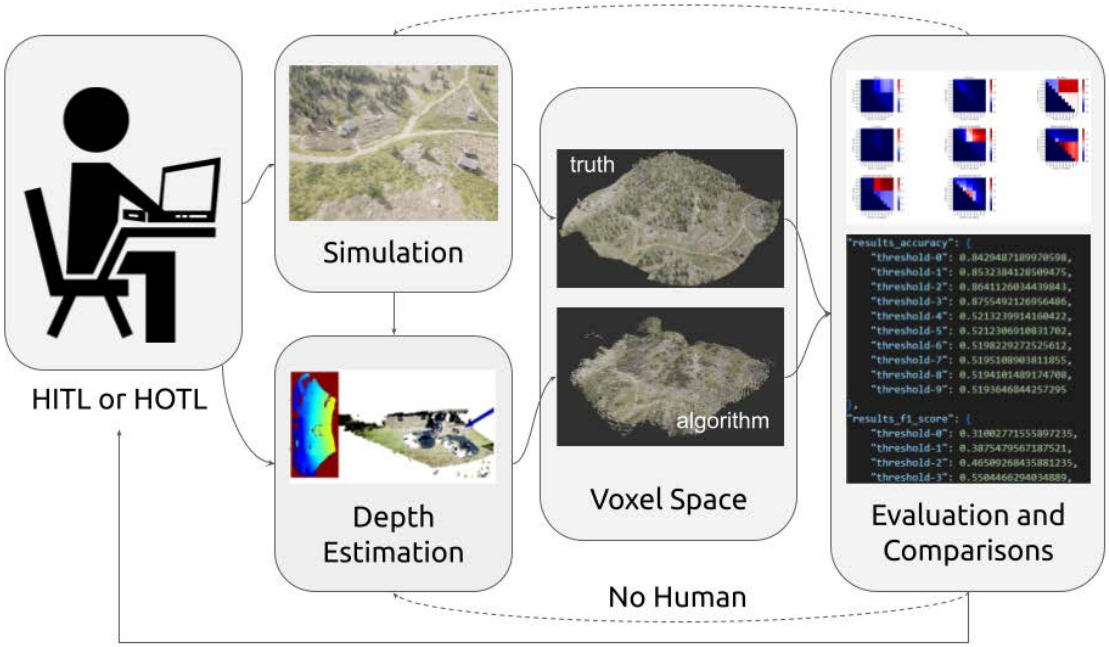


Figure 1: Iterative process for depth estimation algorithm and unmanned aerial vehicle flight context design and evaluation. See Section 1 for more details.

Table 1: Notations and Acronyms

Acronym	Description	Notation
3DR	3D Reconstruction	
SfM	Structure from Motion	
MVS	Multi View Stereo	
SIM	Simulation	
PC	Point Cloud	
Truth	True state of nature	
Gold-standard	Not the truth, best that can be achieved	
SIDE	Single Image Depth Estimation	
	Image at time t, pixel (x, y)	$I_t(x, y) \in \{0, 255\}$
	Depth at time t, pixel (x, y)	$D_t(x, y) \in \Re^+$
VS	Voxel Set	$V(i, j, k) \in [0, 1]$
	Free voxel threshold	$\lambda_F \in [0, 1]$
	Occupied voxel threshold	$\lambda_O \in [0, 1]$
$\tilde{V}_O, \tilde{V}_F, \tilde{V}_U$	Estimated Occupied, Free, and Unknown VS	$\tilde{V}_O(i, j, k), \tilde{V}_F(i, j, k), \tilde{V}_U(i, j, k) \in \{0, 1\}$
V_O, V_F, V_U	Gold-Standard Occupied, Free, and Unknown VS	$V_O(i, j, k), V_F(i, j, k), V_U(i, j, k) \in \{0, 1\}$

2. RELATED WORK

2.1 Monocular Depth Estimation

Many small UAVs are equipped with a monocular image sensor and a GPS/IMU/magnetometer module that provide a color image and camera extrinsics. This is sufficient for techniques to estimate 3D depth. While the current article focuses on a 3D mapping algorithm called EpiDepth,[1] related state-of-the-art methods include multi-view stereo (MVS) and structure from motion (SfM),[2] simultaneous localization and mapping (SLAM),[3–5] etc.), deep learning (DL) for single image depth estimation (SIDE)[6–8]), and DL-based optical flow.[9]

The EpiDepth algorithm[1] takes a stream of world-aligned image frames as input and produces (for acceptable image pairs) a dense, per-pixel depth estimation. Figure 2 shows example EpiDepth image pairs on SIM data collected using AirSim and Unreal Engine (UE). EpiDepth can run in real-time on embedded hardware with

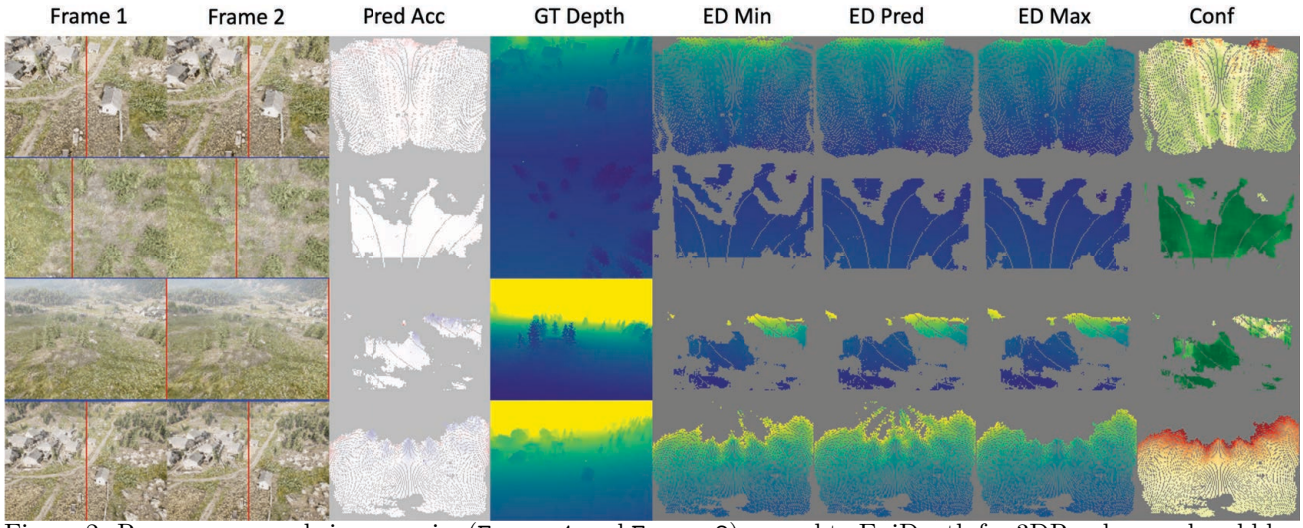


Figure 2: Rows are example image pairs (`Frame 1` and `Frame 2`) passed to EpiDepth for 3DR, where red and blue lines indicate the location of the epipole. `Pred Acc` is a seismic color coded error map, i.e., ground-truth depth minus EpiDepth predicted depth. `GT Depth` is a viridis color coded truth image, and `GT Min`, `GT Pred`, and `GT Max` are EpiDepth's minimum, average, and maximum possible depth assignments per-pixel. Last, `Conf` is a per-pixel confidence that we use to commit depths to voxel space; see our prior article on capturing uncertainty in monocular depth estimation for further details.[10]

parameters to adjust the scale and quality. Frames are selected sequentially based solely on their relative camera extrinsics. In order to be considered, a pair of frames must have a temporal and spatial displacement within a defined range, and also be oriented in the same general direction. From a valid frame pair, the color images are warped and aligned so that optimized block matching techniques can be applied to produce disparity and depth images. The camera-space pixels are then projected into a 3D point cloud (PC) and mapped into world coordinates from the known camera extrinsics. In this study, we do not perform any refinement of the camera position (i.e. using SLAM methods), and rely on the accuracy of the UAV position and orientation sensors to place the points into a common world-space coordinate frame. For the sake of this article, Epidepth generates a depth map, $D_t(x, y) \in \Re^+$, for a selected image pair. While any units could be used, e.g., cm, feet, meters, in whatever space, e.g., UTM, depth maps in this article are in meters in UEs global coordinate system.

2.2 World Space Representation

Next, a world *model* can be addressed in many ways. For example, the majority of 3D mapping algorithms yield unstructured PCs in a local camera coordinate system. Methods like SfM and SLAM combine observations across looks (images or image pairs) to produce a relative, or when external position information is present, global 3D PC. Frequently, such data is quantized into a more efficient data structure like an octree. In areas like robotics, many simplify the world further into an occupancy grid map (OGM). In an OGM, each discrete unit typically has a number (e.g., probability) that is ultimately used to determine its state, e.g., unknown, free, or occupied (UFO). In recent years, more advanced probabilistic structures such as OctoMap[11] and UFOMap[12] have been proposed for ground robotics and drones. In Ref. 13, O'Meadhra et al. introduced a compressed, generative, and efficient variable resolution structure based on probabilistic OGM via Gaussian mixture models for autonomous cave surveying[14] and reactive collision avoidance.[15] Last, for sake of completness, another common practice is to convert PCs into a mesh by the use of methods like the ball-pivoting algorithm (BPA) or Poisson reconstruction.

In this article, an Epidepth depth map at time step t is provided to UFOMap to produce a probabilistic voxel set $V(i, j, k) \in [0, 1]$. Given user specified parameters λ_F and λ_O, $V(i, j, k)$ can be partitioned into binary sets $\tilde{V}_O, \tilde{V}_F, \tilde{V}_U$, corresponding to occupied, free, and unknown. Figure 3 shows a UFO map.

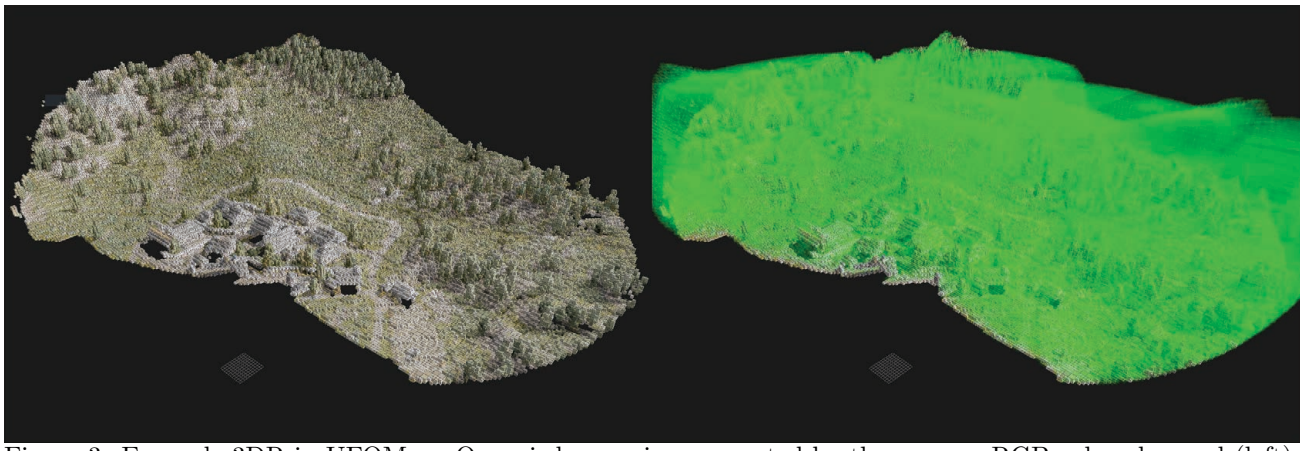


Figure 3: Example 3DR in UFOMap. Occupied space is represented by the average RGB color observed (left), while free space may also be (optionally) visualized as a green translucent fog (right). Note that while not displayed, the remaing voxels are considered to be explicitly unknown space.

2.3 Simulation Environment

A classical SIM environment, driven largely by robotics, is Gazebo.[16] While Gazebo is not state-of-the-art with respect to photorealism, it has advantages such as extensive integration with the robotic operating system (ROS)[17] and it supports a wide range of physics and sensors (e.g., RGB, RGBD, GPS, IMU, LiDAR, sonar, etc.). With respect to improved realism, Apple's HYPERSIM, Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding,[18] does not support physics but it can produce nearly indistinguishable (to a human) photorealistic imagery with information about scene geometry, material information, lighting, per-pixel instance segmentations with respect to diffuse reflectance, diffuse illumination, and non-diffuse residual term that captures view-dependent lighting effects. Another current approach is NVIDIA's Isaac Gym,[19] an end-to-end simulation platform, built from the ground up to run large-scale, physically-accurate multi-sensor simulation for applications like autonomous vehicles and virtual worlds. Epic Games' Unreal Engine (UE) is another solution which has an extremely generous licence that varies based on terms of use;[20] ranging from free to percentage of profits. The UE (see Figure 4) started as a game engine, but it has recently settled into new applications like architecture, film, computer graphics, and beyond. A somewhat new trend is extensions to these core tools, like the AirSim[21] plugin for UE4; which allows for external python support and simulation of different low altitude drones. Last, a more recent competitor to UE is Unity.[22]

Figure 4: Example urban and rural images showing UE5 photorealism.

The above examples are from robotics, computer graphics, gaming, and the hobbyist communities. SIM dates back decades and its scope is beyond the current article. SIM frequently involves approaches from finite element modeling, wave simulation, ray casting, and beyond to accurately model topics like electromagnetics

and physics. Examples include COMSOL for multiphysics SIM and MEEP, a free and open-source software package for electromagnetics simulation via the finite-difference time-domain (FDTD). These SIMs are often based on simulating sensors such as the visual spectrum (RGB) to multispectral, hyperspectral, radar, LiDAR, acoustics, and beyond. A recent physics-based example from the literature is the virtual autonomous navigation environment (VANE) and ANVEL platforms.[23] The reader can also refer to the internet for more expensive and custom business solutions, such as Scale AI. While the number of SIM possibilities is endless, these tools are only a part of the big SIM picture. Most of these SIM solutions are provided "as is". That is, they have some level of trust associated with them, typically based on their design, but most take shortcuts or are discrete approximations to continuous processes and only a few have undergone a more rigorous process of verification and validation (V&V). The reader can refer our article[24] on a recent historical review of the development of verification and validation theories for simulation models.

The ideas expressed in this article can be executed on a variety of *platforms*. To date, we have a single implementation that runs on Windows, Ubuntu, and a Jetson for real-time UAV processing and autonomy. Additionally, while we focus on UE, another SIM environment such as Unity could be used with minimal effort. The reader can refer to our article on "How Should Simulated Data Be Collected for AI/ML and Unmanned Aerial Vehicles" for full details about how we produce RGB, depth, object IDs, and other data layers.[25]

3. SIM ENVIRONMENT AND WORKFLOW

This section documents our process for generating and evaluating a 3DR relative to a SIM gold-standard. First, datasets are collected through AirSim and UE. These collections are carefully controlled with all movements pre-scripted and deterministic to ensure repeatability and accuracy. The collect process generates several assets, including precise camera extrinsics, RGB image data, and a gold-standard per-pixel depth. Figure 5 illustrates our pipeline relative to a single flight (data collection).

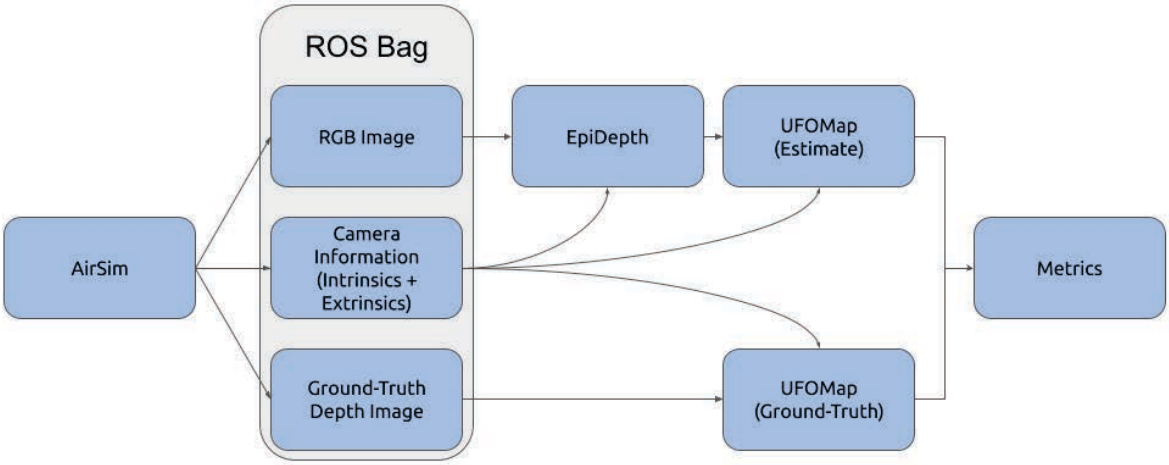


Figure 5: High-level illustration of the flow of data during the course of a single 3D reconstruction and evaluation. First, AirSim is used to fly a drone around and obtain data in UE, resulting in bagged imagery with truth. Next, Epidepth is run offline on the ROS bags to produce an aggregate UFOMap. This 3DR is then subjected to evaluation relative to a ground-truth UFOMap.

Our pipeline uses ROS as a backbone for inter-process communication and modular compatibility. More specifically, we are using ROS Noetic Ninjemys, released on May 23rd, 2020 for Ubuntu 20.04 (Focal). While limited support for the Windows operating system is available, we found that using an Ubuntu 20.04-based Docker container for computing metrics was necessary for ensuring compatibility with the popular "catkin_tools" CLI required to build the UFOMap node.

The result of our first step, data and truth generation, is a ROS .bag file. Our codebase is flexible and it has been run in real-time on an embedded device (Jetson) using real drone imagery and metadata. However, in our

SIM workflow, and when it comes to our real-time platform, there are experimental advantages to producing ROS bags versus online processing. First, due to a multitude of reasons (code, random numbers, computer processing and ROS queues, etc.), subtle differences could arise from run-to-run and lead to slightly different data. If a goal is to explore 3DR algorithm parameters, this adds an uncontrolled confounding factor to our experiment. Instead, if data is collected and stored once to a ROS .bag, then offline we can ensure that algorithms downstream receive exactly the same data. While subtle, this is an important detail.

Next, we replay the contents of a stored .bag file containing the collected SIM data. Note that, since all of the input data is already in a standard ROS format at this point, the entire pipeline is fully compatible with any SIM platform, not just our current choice of AirSim, so long as it is capable of producing the required information. This process is relatively straightforward using the "rosbag" command-line tool included with ROS. However, since downstream nodes may use ROS' TF2 transform system to look up camera extrinsics at an arbitrary point at time, a critical part of this phase is that the Header types contained within several of the messages need to be modified so that their timestamps reflect the time of replay rather than the time. We accomplish this using our own "ros_replay" tool written in Python that interacts directly with ROS' included "rosbag" library. While we found it best to entirely avoid using the TF2 library wherever possible for this application, the "ros_replay" tool is still incredibly useful for manual control and logging of message timing and metadata.

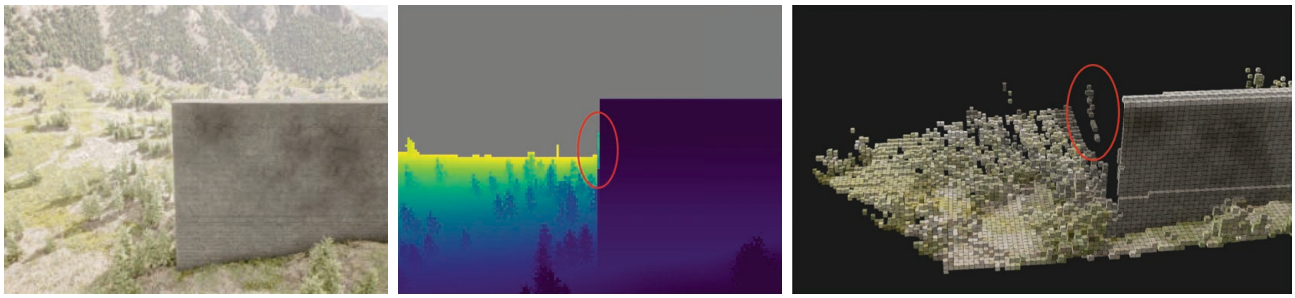
The next step is to run the 3DR algorithm. In this paper, we explore EpiDepth. Our implementation contains native support for ROS and performs all of its I/O through it. The output of this process is a 2D estimated depth map and RGB PC, calculated with respect to the recorded platform extrinsics. Since EpiDepth is processing a fixed ROS bag, its parameters can be varied to better understand our algorithm.

The final processing phase is our custom fork of UFOMap. It uses the output PCs along with camera pose information to form an aggregate probabilistic model of the world in voxel space. We run two instances of UFOMap concurrently: one for aggregating the estimated output, and another for aggregating the gold-standard output. Our custom fork introduces several new ROS services, including the ability to export the map as a CSV file that can be easily processed externally in Python. Thus, we can evaluate the final state of the 3D representation directly using voxel-space metrics.

4. GOLD-STANDARD VS GROUND-TRUTH

In the physical universe, the true state of nature for many phenomena is fundamentally not known, or at least not known to a certain degree with respect to limitations in observation. Thus, we instead use the term *gold-standard* to denote the "benchmark that is the best available under reasonable conditions".[26] This is the case for CV. What is an image? What is a pixel? A pixel (picture element) is a discrete unit of space and time denoting a sensor's response to an incoming collection of information that traveled to the lens, was recorded by the film back, and eventually converted into an electrical signal. A pixel is a "bundle" of data, not a single sample. Thus, questions like "what object is at" or "what is the depth of" a pixel are fundamentally ill-posed questions. A sensor, e.g., RGB camera, only records a single value per pixel. SIM only records a single value as well. It is inefficient for SIM to capture and record all incident information, and most researchers will never make use of such data regardless. However, in its absence, there is no real truth. Instead, we are subject to living with a gold-standard. While this sounds sad, it is simply reality. The good news is, this gold-standard is orders of magnitude more accurate and dense than data collected in the real world.

The aim of the current section is to simply raise awareness of this conundrum, as it cuts at the heart of how to define evaluation metrics relative to SIM. Specifically, not all SIM gold-standard data can be taken with full legitimacy. The reader can refer to our article on "Ignorance is bliss: flawed assumptions in simulated ground-truth"[27] for further details on why, when, and where this occurs, and ultimately methods for remediation. Figure 6 is a common example that demonstrates depth artifacts that arise from mixed pixels on the edge of objects. In our prior article,[27] we discuss the bias and error associated with computing a single truth aggregate per pixel. This frequently occurs in the context of multi-sample (spatial and/or temporal) anti-aliasing when generating an image and its associated truth. Furthermore, we discuss problems with the common misconception that it is possible to simply fix this problem by forcing a single truth, e.g., assign a single object ID or depth in the case of UEs movie render queue. This action is not correct and it results in uni-modal truth that effects AI and/or



(a) (b) (c)

Figure 6: Example illustrating a scenario that can arise in the generation of SIM truth. Consider the wall in 6a. If a single depth is stored per pixel, then object edges are mixed (part wall, part terrain), what should SIM store? In 6b and 6c, AirSim computed an anti-aliased value of the foreground (wall) and background (terrain). The result is floating depth observations that are wrong, as they are neither wall nor terrain. Instances like this are why we refer to the SIM data as a "gold-standard" rather than ground-truth.

our evaluation metrics. Instead, in[27] we argued for a distribution-based approximation to the truth. Similar flaws can occur in SIM contexts like billboard-based foliage, translucent materials such as glass or water, and other areas where SIM takes shortcuts for the sake of efficiency. In summary, SIM limitations exist and impact our ability to obtain 100% accurate truth. The question is how do we minimize such effects, and how should we design metrics with such knowledge in mind?

5. EVALUATION MEASURES

In this section, we explore a few measures to compare 3DR algorithm outputs to a gold-standard acquired via SIM[*]. Herein, we focus on multi-frame voxel set measures[†]. This was decided because our applications only require voxel level resolution, not a PC or mesh. However, there are consequences associated with this decision. First, a voxel space has lost all connection to what images/pixels a given voxel's data is derived from. This level of information would be impractical to store, and its not clear exactly how one would store it (e.g., in UFOMap) or even if it's worth storing. As a result, UFO space can only be compared at an overall aggregate level, not at an individual correspondence resolved observation level. Second, we have followed UFOMap's procedure for converting depth maps and 3D point clouds into a voxel space. Specifically, individual observations are converted to voxel space ray queries, versus view frustums, and probabilistic updates to voxel space are made. By design, the algorithm voxel space is what it is. However, what is the gold-standard voxel space? Herein, we take the gold-standard depth, per frame, and use it to build a UFOMap. The rationale is as follows. If the 3DR algorithm was perfect, these are the exact depths. If we were to propagate those values through UFOMap, the result would be a corresponding gold-standard. Thus, we are comparing what we did get to what we should have obtained. This is different from a procedure that takes the geometry of a SIM scene and directly converts it into UFO voxel sets. The reason why we did not pursue that route is because those sets would contain elements that were not observed during simulated flight and thus also not observable by our algorithm. This complicates scoring. Thus, our outlined procedure is the fairest approach we could imagine. In the remainder of this section, our measures can largely be divided into two categories, set-based (Algorithms 1 and 2) and surface-2-surface based (Algorithm 3).

[*]The reader can refer to the CV literature for more common real-world quantitative measures. Individual frame measures for complex scenes are often based on a sparse, secondary, and often questionable (error-bound) sensors like LiDAR. Typical multi-frame measures include feature track statistics, e.g., how many features, track strengths, track lengths, etc. There even exists preliminary SIM work for synthetic feature track analysis.[28] The literature also possesses many PC-to-PC, PC-to-surface, surface-to-PC, and related measures.

[†]In future work, we plan to address per-frame measures. An advantage of per-frame is pixel correspondence. That is, each estimated pixel depth/location can be compared to its exact corresponding depth/location.

Algorithm 1 : p =Prec(V, $\tilde{V}$) $\in [0,1]$, a =Acc(V, $\tilde{V}$) $\in [0,1]$, r =Recall(V, $\tilde{V}$) $\in [0,1]$, F1 =F1(V, $\tilde{V}$) $\in [0,1]$

INPUT: $V = \{V_O, V_F, V_U\}$, $\tilde{V}$, $\lambda_F \in [0,1]$, $\lambda_O \in [0,1]$

1. Set $\tilde{V}_F(i,j,k) = 1$, $\forall(i,j,k)$ s.t. $\tilde{V}(i,j,k) < \lambda_F$, 0 otherwise ▷ Make binary sets
2. Set $\tilde{V}_O(i,j,k) = 1$, $\forall(i,j,k)$ s.t. $\tilde{V}(i,j,k) > \lambda_O$, 0 otherwise ▷ Make binary sets
3. Calculate confusion matrix M with respect to $\{\tilde{V}_O, \tilde{V}_F\}$ and $\{V_O, V_F\}$ ▷ Conf(Truth, Algorithm)
4. $p = \frac{M(1,1)}{M(1,1)+M(2,1)}$ ▷ Precision
5. $a = \frac{M(1,1)+M(2,2)}{M(1,1)+M(1,2)+M(2,1)+M(2,2)}$ ▷ Accuracy
6. $r = \frac{M(1,1)}{M(1,1)+M(1,2)}$ ▷ Recall
7. $\text{F1} = 2\frac{p*r}{p+r}$ ▷ F1-score

RETURN: $p, a, r, \text{F1}$

In Algorithm 1, the gold-standard and 3DR algorithm UFOMaps are first partitioned into their corresponding free, occupied, and unknown binary sets. From there, a number of standard crisp metrics can be computed, e.g., precision, accuracy, recall, and F1-score. Thus, these measures are based on the logic, "how similar is a 3DR algorithm estimate to the gold-standard on a per voxel basis?" A problem is, this metric is harsh and unforgiving. There is a level of acceptable error, based on the size of a voxel. However, if a 3D mapped point is even a fraction of a percentage outside of the answer voxel, it is counted against the algorithm. It is well-known that error in our depth estimation algorithm increases with range and with respect to factors like baseline. The point being, this class of metric does not exhibit graceful degradation with respect to this nature of inaccuracy.

Algorithm 1 can be carried out a number of ways. We use a confusion matrix based on combining occupied and free space. However, Algorithm 1 can clearly be computed on any combination of crisp sets, e.g., free-to-free, occupied-to-occupied, free and occupied to free and occupied, and etc. It is up to the user to decide what their question is. Occupied-to-occupied informs us about how well two methods agree on mapping the visible regions on observed surfaces. On the other hand, free-to-free informs the reader about how much free space the algorithm should have seen versus what it did. This can be useful in contexts like drone/robot exploration. Note that metrics on unknown space are meaningless, as the default state of the map is an infinite grid of unknown space, and applying a bounding box for calculations in the same manner as free/occupied space would be arbitrary. Thus, only free and occupied space are evaluated.

Algorithm 2 : Raw counts and statistics of VS A in B

INPUT: $A = \{A_O, A_F, A_U\}$, $B = \{B_O, B_F, B_U\}$, distance threshold ϵ

1. l_{AB} is the number of locations in A such that $A_O(i,j,k) == 1$
2. $\hat{l}_{AB}$ is the number of locations in A that have a distance less than ϵ to B
3. $c_{AB} = \left(\frac{\hat{l}_{AB}}{l_{AB}} \in [0,1]\right)$ ▷ Percentage of points in A matched to B with respect to ϵ

RETURN: $\hat{l}_{AB}$, c_{AB} ▷ Raw counts and percentage of matched points

Algorithm 2, is a simpler and more tolerant variant of Algorithm 1. Let the 3DR algorithm voxel set be A and the gold-standard be B. As already stated, the gold-standard (B) may contain questionable observations (it is not actual truth). Furthermore, it is common practice to filter a PC and/or voxel set. We have not considered any of these factors in Algorithm 1. Algorithm 2 begins by counting the total number of occupied points in A[‡]. Next, the distance of each occupied voxel in A is calculated to B. Voxels with distance greater than user defined value ϵ are removed. Thus, only points that have a reasonable match to B are considered. Next, a ratio of points that match to B over all possible mapped points is computed. This $[0,1]$ ratio informs the user how well A maps to B, relative to tolerance ϵ. A high value indicates that the algorithm is very accurate in its mapping. However, the challenge with Algorithm 2 is, consider an algorithm that only assigns a few points. Let that algorithm be extremely good in its mapping. This ratio can be high, but overall only a small portion of the scene is mapped. One way to identify this scenario is to run Algorithm 2 with the truth as A and algorithm as B. Thus, the reader

[‡]The reader can clearly swap out occupied points for free points, but semantics need be considered.

needs to probably consider measures in both direction to understand accuracy and completeness of mapping. It should be noted, this algorithm is slightly improved, in tolerance, over Algorithm 1. However, it is still geared to return a relation in $[0, 1]$, versus helping us understand something like, "what is the average error between our algorithm and truth?"

Algorithm 3 : Surface-2-Surface MSE of VS A to B, $m_{AB} = \text{Surf2Surf}(A, B) \in [0, \infty]$

INPUT: $A = \{A_O, A_F, A_U\}$, $B = \{B_O, B_F, B_U\}$, distance threshold ϵ
1. Initialize $G(i, j, k) = 1, \forall i, j, k$ such that $B_O(i, j, k) == 0$, $G(i, j, k) = 0$ otherwise.
2. $D = \text{DistTransform}(G)$. ▷ Distance to occupied cells (set of distances)
3. Let I be the set of all indices in A such that $A_O(i, j, k) == 1$. ▷ All occupied cells in other set
4. Let Z be the set of all D distances for the set of indices I. ▷ Corresponding distances
5. Remove all values in Z greater than ϵ. ▷ Remove extreme distances and error points in gold-standard
5. Set m_{AB} to the average of Z. ▷ Final MSE value

RETURN: m_{AB}

The last measure considered herein is Algorithm 3. This algorithm is based on surface-2-surface comparison between occupied sets. Since we are considering 3DR CV applications, e.g., a drone mapping an environment, it is assumed that occupied locations are visible surfaces on objects, free space is empty visible volumes, and unknown are locations not seen and solid non-visible volumes. Algorithm 3 begins by computing the distance transform from set A to B (two sets of mapped surfaces). As outlined in the last paragraph, A and B can be our algorithms voxel space and truth, or vice versa. Next, extreme differences (user threshold ϵ) are removed and the average error is computed on the remainder. Unlike Algorithms 1 and 2, this measure considers extreme non-matches, it more gracefully degrades, and it returns an average error vs an overlap amount. However, this error is clearly subject to the underlying voxel resolution. It cannot resolve tolerances less than that resolution and the result is clearly still impacted by the discrete grid that the voxel space resides on.

Before we conclude this section, we discuss the elephant in the room. Which measure is best? This is complicated. Each measure captures a different aspect of our result. Furthermore, say the application is exploring a large area and exploration is the goal. The user might be concerned with the amount of free space observed, its rate of expansion, and accuracy of observed locations might be relatively unimportant. On the other hand, another application might not care about clearing a lot of space. Instead, the user might desire to know how much occupied space was mapped and to what error level. The point is, different applications and users will likely require using different combination logic with respect to these measures. Last, its worth mentioning that these are not mutually exclusive measures. For example, a user might run Algorithm 3 first to get a feel for what the average voxel mapping error is in some unit, e.g., meters. Next, the reader might run Algorithm 2 from algorithm-to-truth and truth-to-algorithm to better understand what percentage of the truth was observed, at what error rate, and how many locations in the algorithms voxel space did not reasonably map to truth. The point is, our initial paper explores different measures and how to connect these calculations to high level user desires. In future work, we will go the next step and explore attributing a task and connecting what measures and combination logic are required to rank order experiments.

6. DATASETS AND EXAMPLES

The ideas outlined in this paper can be used for a number of feats. For example, our workflows can be used to understand and pick an operating context, e.g., relative camera pose, flight altitude, motion, etc. Our process can also be used to iteratively explore and fine-tune a 3DR algorithm. Our process can even be used to compare 3DR algorithms to each other instead of an algorithm to the truth. There is simply too much ground to cover in this initial article. As a result, we focus on a single use case from our research. We assume EpiDepth is operating ideally, thus parameters have already been selected, and we try to use the outlined measures to understand and identify an ideal platform operating context in a specified environment.

Figure 7 shows our UE environment. We used the Mountain Village Environment[29] from the UE Marketplace.[30] This environment was selected for a number of reasons, specifically: 3D object, material, and visual

Figure 7: Example large 2x2 km map with man made structures (houses, roads, etc.) and nature (hills, grass, trees, etc.) in Mountain Village Environment[29] from the UE Marketplace.[30]

quality; cost (79.99 USD); use of man made structures (buildings); nature (trees, rocks, etc.); environment size (2x2 km); and bounded horizon (enclosed by close mountains). Overall, this scene is nice because it allows us to collect data over a region with buildings with common edge and unique features. On the other hand, another portion of this scene can be used that has just rocks, grass, trees, and other nature with perhaps harder to match and map features. The point being, this scene is a nice balanced 3DM sandbox.

In Table 2, we discuss our eight datasets. This data can be described according to the following underlying variables: region, altitude, and look angle. Two regions were explored: *region one*, which has man made structures (roads, buildings, etc.), and *region two*, which is just nature (rocks, grass, trees). Two altitudes were explored, "low" and "high". For this article, this level of granularity is sufficient. We are only interested comparing these two contexts. In our article on training and evaluating passive ranging in SIM,[31] we explored distance as a continuous variable and semantic IDs were used to subdivide performance relative to specific object categories. Thus, we could ask questions like, "what is performance for vehicles as a function of range?" The last variable we consider herein is look angle, which is either nadir (straight down) or slant angle (45°).

Table 2: Dataset Summary

DS1	region 1 (man made), height high, nadir look angle
DS2	region 1 (man made), height high, 45 degree look angle
DS3	region 1 (man made), height low, nadir look angle
DS4	region 1 (man made), height low, 45 degree look angle
DS5	region 2 (nature), height high, nadir look angle
DS6	region 2 (nature), height high, 45 degree look angle
DS7	region 2 (nature), height low, nadir look angle
DS8	region 2 (nature), height low, 45 degree look angle

All together, this results in $2 \times 2 \times 2 = 8$ datasets. The simplest task we could perform is to analyze performance with respect to a single dataset. However, when combined, these datasets help us address more complex questions like the following. A specific question would be Q1="how high should we fly if we are looking nadir in a region with man made structures?" Mid level question is Q2="do we do better in a structured man made region versus nature" or Q3="do we do better flying high or low?" Of course, the highest, thus most generic question we could ask is, Q4="what situation do we do best in?" Clearly, each of these questions require us to compare across different datasets. Q1 requires us to compare performance on DS1 to DS3. Q2 requires comparing {DS1, DS2, DS3, DS4} to {DS5, DS6, DS7, DS8}, and Q4 is analysis over all datasets.

Figure 8 shows the proposed measures applied to DS1. The x-axis in these figures shows variation in the occupied threshold and y-axis free threshold variation. Also, the matrix diagonal and upper diagonal are the only values reported. The lower diagonal values do not correspond to a legitimate scenario, as the free threshold needs to be less than the occupied threshold. These plots are interesting to look at on a number of levels. First, consider `Recall`. The reader can see that there is a performance drop off when free is greater than 0.43 and when occupied is less than 0.51. This makes sense, as 0.5 models unknown. `Recall`'s highest value is when occupied is highest, 0.9, and free is lowest, 0.2. This make sense from the stance of `Recall`, as it represents probably the simplest points to map and therefore the algorithm does best in that context. However, if you look at the `F1 Score`, a combination of `Recall` and `Precision`, you see a different story. When multiple objectives are combined, the best overall performance is a set of thresholds slightly around 0.5. The point is, the user can use these plots to decide, at a quantitative vs qualitative level, what thresholds to select. Furthermore, the user can see what performance, and associated sensitivity with respect to thresholds, can be achieved. Additionally, the `F1 Score` for DS1 is low. Why? It is hard to gather more from these plots. It is also equally hard to look at a single `F1 Score` and know how good that value is. This is the point where a user needs to look into the data at a more qualitative level to help understand if that is perhaps acceptable. Definitely, these values are much easier to use on a comparative basis, e.g., is DS1 better than DS2?

Figure 8 also shows surface-to-surface and intersection ratio results. The reader can see that these are different criteria and there is no obvious winning combination of thresholds across measures. Based on the `F1 Score`, `MSE`, and `Intersection Over` results, a threshold of 0.51 for occupied and 0.43 is likely a good tradeoff, and it leads to to an `Intersection Over Estimate` overlap score of 0.6, `Intersection Over GT` of 0.3, `MSE GT to EpiDepth` of 0.8, and `MSE EpiDepth to GT` of 0.2. This says, "60% of EpiDepth's points match the gold-standard truth, 30% of the truth locations were discovered by EpiDepth, and EpiDepth's error (relative to a 1m cube voxel size) is approximately 0.2m."

Figure 9 is a qualitative comparison of EpiDepth on DS1 relative to the gold-standard. This figure gives the reader a feel for what the EpiDepth reconstruction looks like relative to the numeric measure data just presented. Quickly, the reader can see that Epidepth is having problems with identifying trees. This could not be determined alone just using the provided metrics. Further analysis needs to go into understanding why, e.g., the trees are small structures, EpiDepth's search window was too large, simulation had repetitive textures that made matching features hard, etc. The point is, the metrics give one level of understanding, but its not currently a substitute for qualitative analysis.

Next, Figure 10 shows our measures for DS2. The reason for showing a second set of measures is to explore cross experiment comparison. If we stick with the same set of UFOMap thresholds, then Figure 10 shows a slightly lower `F1 Score` and slightly elevated `MSE Epidepth to GT`. Things got worse for off-Nadir. Why?

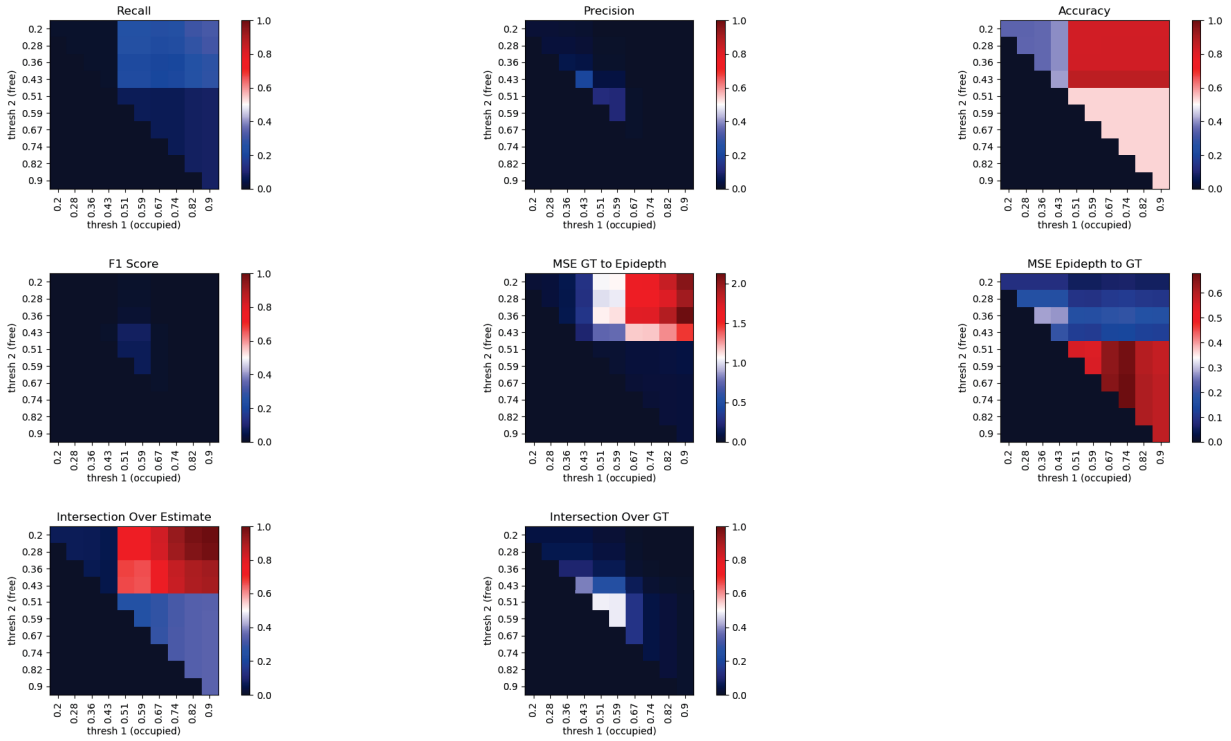


Figure 8: Example measures for DS1. In each image, the x-axis represents the occupied threshold and the y-axis represents the free threshold.

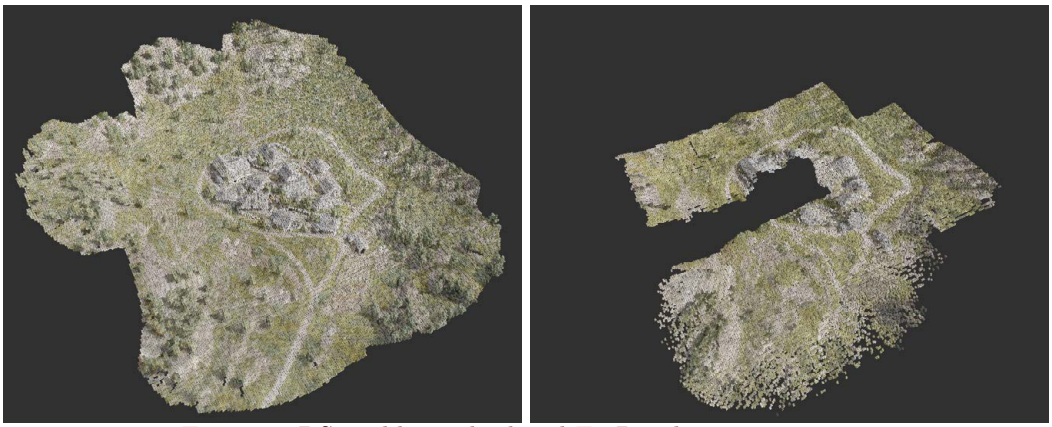


Figure 9: DS1 gold-standard and EpiDepth reconstruction.

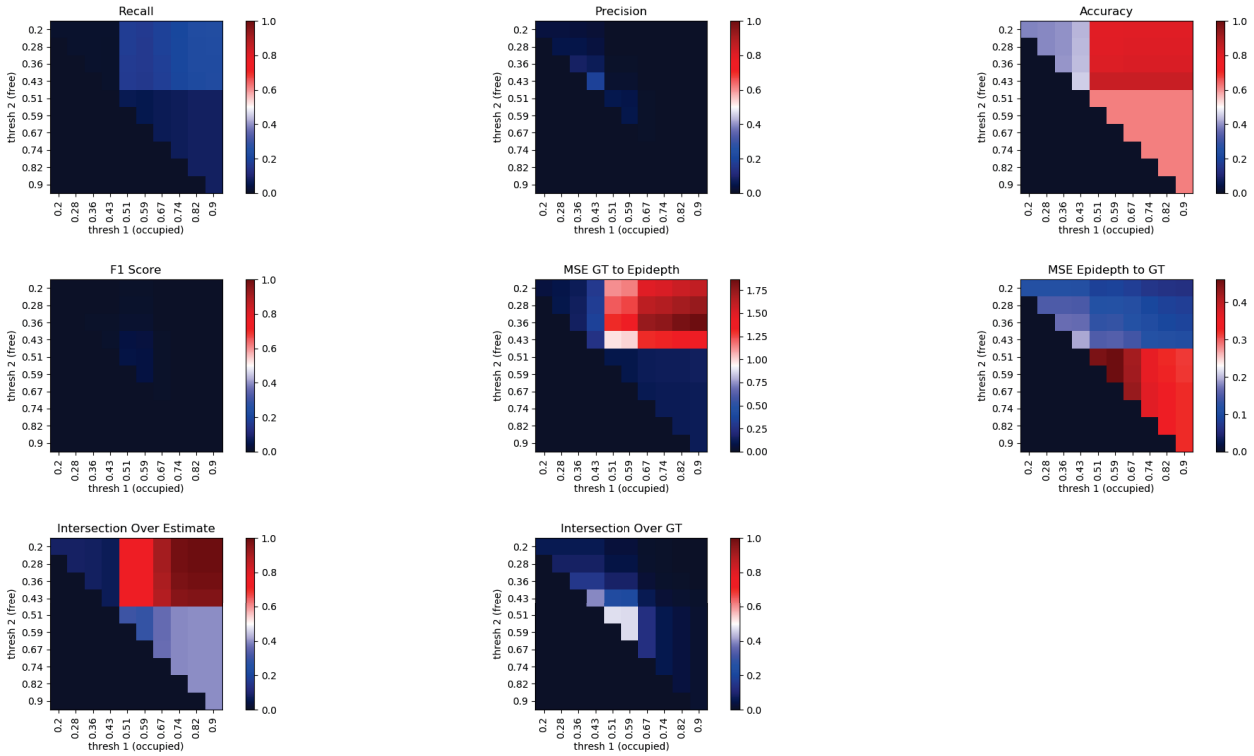


Figure 10: Example measures for DS2.

The point of showing two results is to inform the reader that care should be taken with respect to drawing conclusions across experiments. For example, consider Figure DS1 (Nadir) versus DS2 (slant look angle). These were collected over the same region on the map. This is where the details come in. First, there are two gold-standards, one for each experiment. Second, the slant angle naturally sees more of our map. This makes it challenging to directly compare the measure results. Is it OK if we have a slightly worse F1 Score if we are mapping more space? Should these results be normalized or an intersecting set of co-observed locations be identified and compared? The point is, comparing results across experiments might not be an apples-to-apples experiment.

The point is, we intentionally picked different platform operating conditions because it's a large factor in many areas of research. For example, what is the ideal flight conditions for mapping some area of interest? We are able to, today, do analysis and draw conclusions with respect to a single experiment. However, cross experiment with respect to variation in platform changes presents a harder and open research question. What do we need to do in order to provide a rank-ordered preference across flights? While we focused on platform variation, a simpler and more straight-forward experiment that is less fuzzy is to explore parameters in our algorithm. For example, we could collect one flight, with fixed pose and motion, and change EpiDepth parameters (search window size, frame pair selection, etc.). The platform is not changing, just the 3DR algorithm. This type of multi-experiment exploration is more well-suited for the ideas directly expressed herein.

7. CONCLUSIONS AND FUTURE WORK

In this article, we presented a framework and workflow for collecting SIM data to evaluate the impact of 3DR algorithm parameters and platform motion. Different set and surface-to-surface voxel-based measures were presented. These metrics were then used to individually analyze different datasets.

This article is a preliminary investigation. More research is needed in all of the areas identified above. For example, the proposed measures are focused on aggregated multi-look voxel spaces. We would like to also

explore per-frame measures with improved pixel-level correspondence and aggregation of this information across looks. Our article is also focused on a static scene. It remains a question about how to best address dynamic environments. Furthermore, SIM truth results in a gold-standard. The very nature of mixed truth pixels should be incorporated into our measures, and it's not currently understood to what degree errors in the gold-standard impact and skew our measures. We would also like to explore evaluation of associated metadata, e.g., RGB data or surface normal per voxel. At the moment, we have only considered the task of can the algorithm determine the location of surfaces in the world.

In addition, we only demonstrated preliminary results for one nature scene across eight contexts. A further analysis on multiple scenes with varying conditions, e.g., time of day, weather, etc. should be considered. We also highlighted that a more fair starting point, with respect to comparing measures across experiments, would be algorithm parameter search. We are currently working on a separate article just focused on how to use the human-in-the-loop to achieve this, i.e., 3DR algorithm fine-tuning. Another interesting avenue of exploration is how take this multitude of different measures and to attribute applications. What is the right set of measures and method of combining their logic to make a rank ordering of experiments?

There are also a number of minor, but important, details we would like to address. Herein, AirSim was used to control our agent and get information from UE. AirSim is technically not supported for UE5, but it has been extended by the community. We are working on our own direct C++ plugin that does control and improved photorealism collection via UE's Movie Render Queue. It could be interesting to study what impact these rendering features have on a 3DR algorithm. Furthermore, we would like to improve the generation of data layers, like semantic IDs, as bundles (due to limitations like mixed pixel truth). The measures used herein did not consider class identifier context. Inclusion of this information in evaluation can lead to a much deeper understanding of what "features" drive and limit our CV algorithms.

REFERENCES

[1] Camaioni, R., Luke, R. H., Buck, A., and Anderson, D. T., "EpiDepth: A real-time monocular dense-depth estimation pipeline using generic image rectification," in [*Geospatial Informatics XII*], **12099**, 101–114, SPIE (May 2022).

[2] Schönberger, J. L. and Frahm, J.-M., "Structure-from-motion revisited," in [*2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 4104–4113 (2016).

[3] Kiss-Illés, D., Barrado, C., and Salamí, E., "GPS-SLAM: An augmentation of the ORB-SLAM algorithm," *Sensors* **19**(22) (2019).

[4] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D., "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics* **31**, 1147–1163 (Oct. 2015).

[5] Mur-Artal, R. and Tardos, J. D., "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics* **33**, 1255–1262 (Oct. 2017).

[6] Gordon, A., Li, H., Jonschkowski, R., and Angelova, A., "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras," in [*Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*], (Oct. 2019).

[7] Li, H., Gordon, A., Zhao, H., Casser, V., and Angelova, A., "Unsupervised monocular depth learning in dynamic scenes," (2020).

[8] Goldman, M., Hassner, T., and Avidan, S., "Learn stereo, infer mono: Siamese networks for self-supervised, monocular, depth estimation," in [*Computer Vision and Pattern Recognition Workshops (CVPRW)*], (2019).

[9] Hui, T.-W., Tang, X., and Loy, C. C., "LiteFlowNet: A lightweight convolutional neural network for optical flow estimation," in [*Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 8981–8989 (June 2018).

[10] Buck, A., Anderson, D., Camaioni, R., Akers, J., Luke, R., and Keller, J., "Capturing uncertainty in monocular depth estimation: Towards fuzzy voxel maps," in [*FUZZ-IEEE*], (2023).

[11] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W., "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots* **34**, 189–206 (Apr. 2013).

[12] Duberg, D. and Jensfelt, P., "UFOMap: An Efficient Probabilistic 3D Mapping Framework That Embraces the Unknown," *arXiv:2003.04749 [cs]* (Mar. 2020).

[13] O'Meadhra, C., Tabib, W., and Michael, N., "Variable resolution occupancy mapping using gaussian mixture models," *IEEE Robotics and Automation Letters* **4**(2), 2015–2022 (2019).
[14] Tabib, W., Goel, K., Yao, J., Boirum, C., and Michael, N., "Autonomous cave surveying with an aerial robot," *IEEE Transactions on Robotics* **38**(2), 1016–1032 (2022).
[15] Dhawale, A., Yang, X., and Michael, N., "Reactive collision avoidance using real-time local gaussian mixture model maps," in [*2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*], 3545–3550 (2018).
[16] "Gazebo." http://gazebosim.org/. (Accessed: 1 March 2021).
[17] "Robot Operating System (ROS)." https://ros.org. (Accessed: 8 March 2022).
[18] Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M., "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in [*ICCV*], (2021).
[19] Liang, J., Makoviychuk, V., Handa, A., Chentanez, N., Macklin, M., and Fox, D., "Gpu-accelerated robotic simulation for distributed reinforcement learning," (2018).
[20] "Unreal Engine." https://www.unrealengine.com/. (Accessed: 8 March 2022).
[21] "AirSim." https://github.com/microsoft/AirSim. (Accessed: 1 March 2021).
[22] "Unity." https://unity.com/. (Accessed: 1 March 2021).
[23] 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).
[24] Durst, P., Bethel, C., and Anderson, D., "A historical review of the development of verification and validation theories for simulation models," *International Journal of Modeling, Simulation, and Scientific Computing* **08** (01 2017).
[25] Kerley, J., Anderson, D., Alvey, B., and Buck, A., "How should simulated data be collected for ai/ml and unmanned aerial vehicles?," in [*SPIE*], (2023).
[26] Versi, E., ""gold standard" is an appropriate term.," *BMJ* **305**(6846), 187–187 (1992).
[27] Buck, A., Anderson, D., and Fraser, J., "Ignorance is bliss: flawed assumptions in simulated ground truth," in [*FUZZ-IEEE*], (2023).
[28] Degol, J., Lee, J. Y., Kataria, R., Yuan, D., Bretl, T., and Hoiem, D., "Feats: Synthetic feature tracks for structure from motion evaluation," in [*2018 International Conference on 3D Vision (3DV)*], 352–361 (2018).
[29] "Mountain Village Environment." https://www.unrealengine.com/marketplace/en-US/product/mountain-village-environment. (Accessed: 8 March 2022).
[30] "Unreal Marketplace." https://www.unrealengine.com/marketplace/en-US/store. (Accessed: 1 March 2021).
[31] Buck, A., Deardorff, M., Murray, B., Anderson, D., Keller, J., Popescu, M., Ho, D., and Scott, G., "Estimating depth from a single infrared image," in [*MSS*], (2023).

APPENDIX A. FULL DATASET MEASURE RESULTS

The remaining measures for datasets 3 through 8 are reported in Figures 11 through 16.

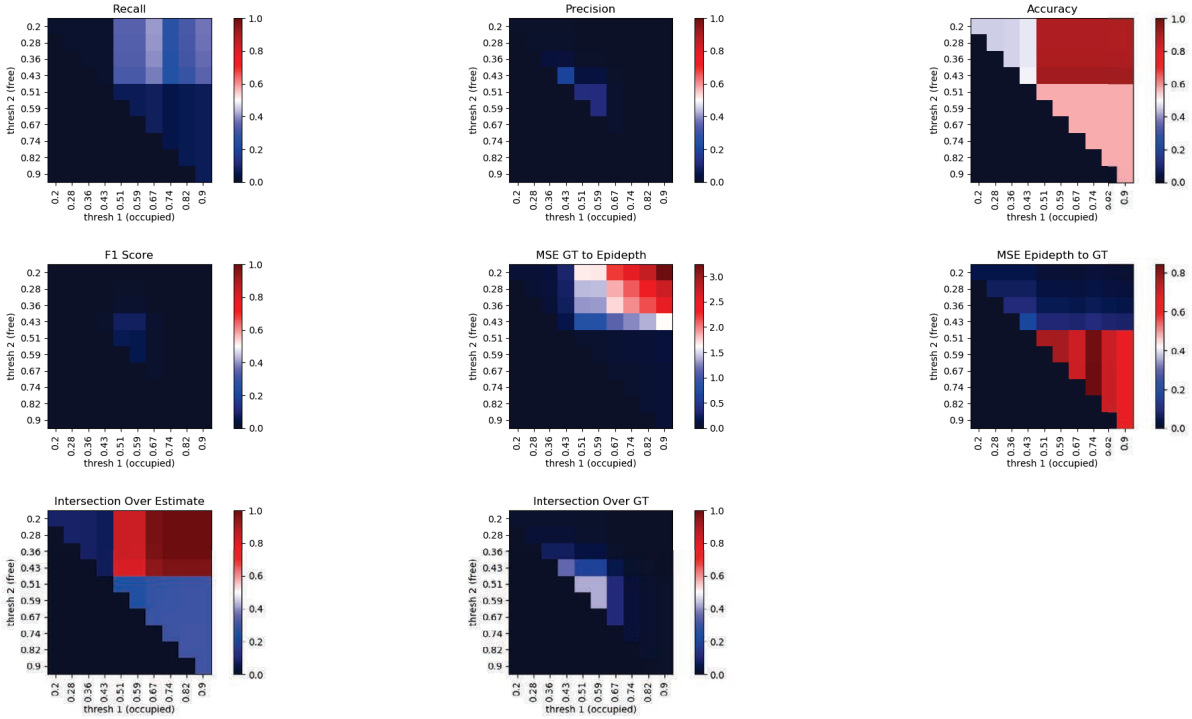


Figure 11: Measures for DS3.

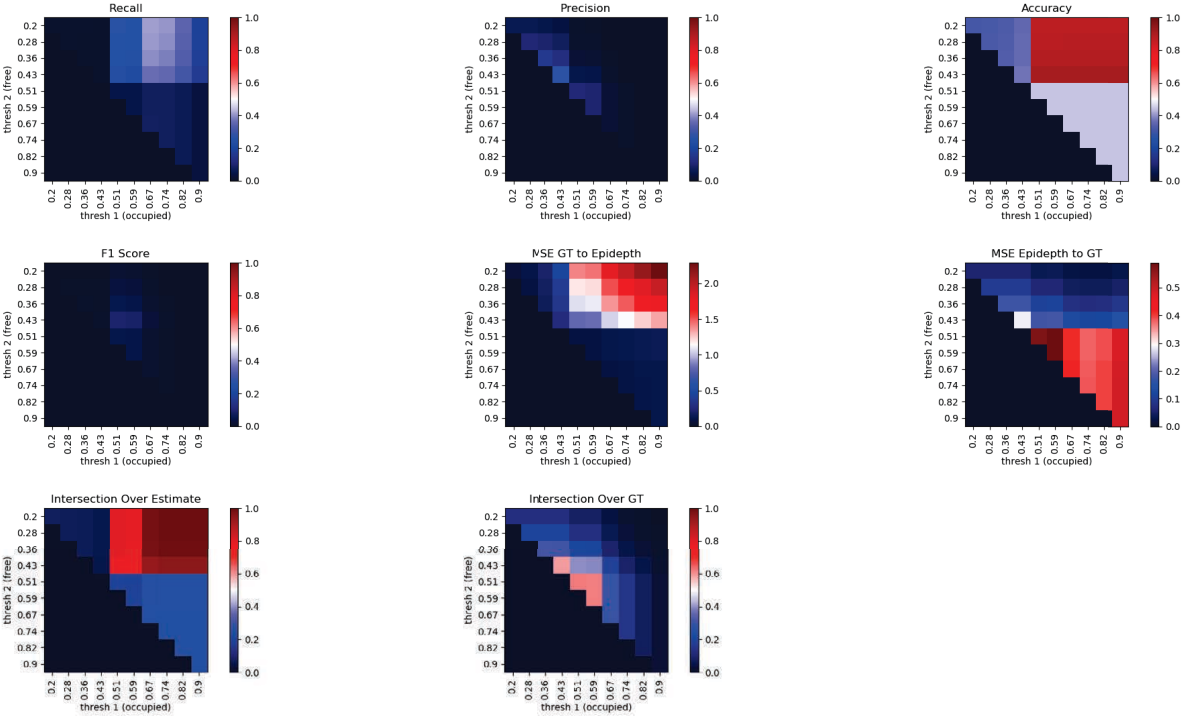


Figure 12: Measures for DS4.

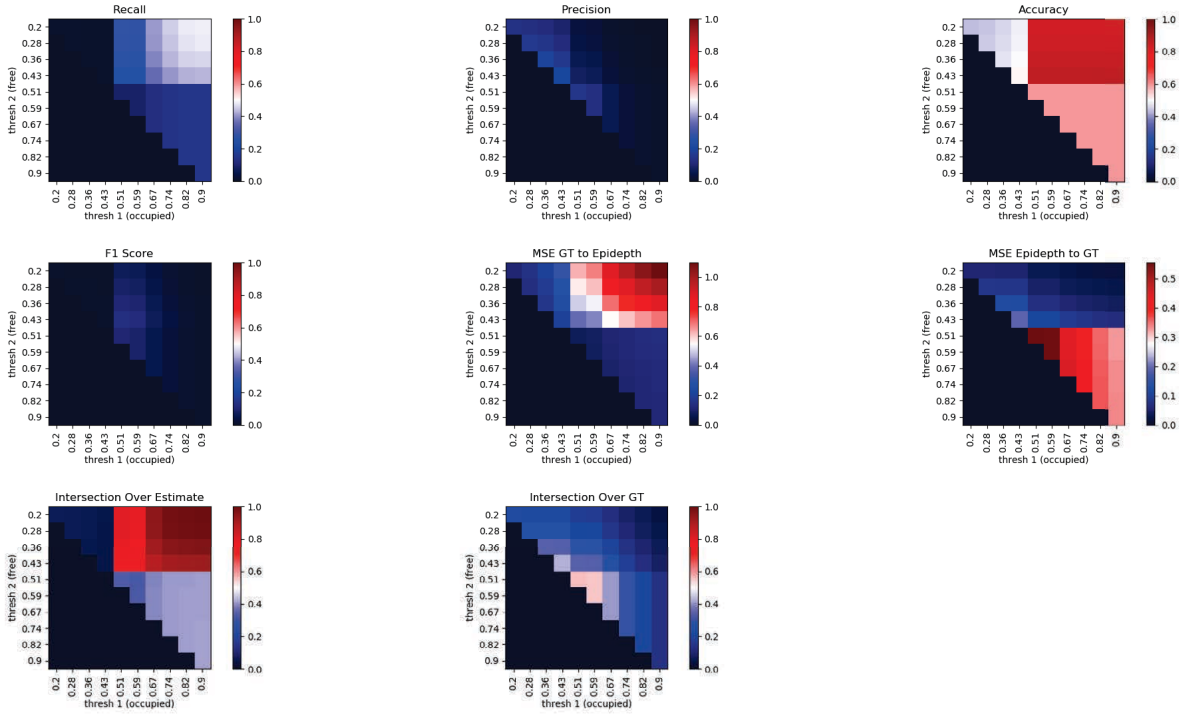


Figure 13: Measures for DS5.

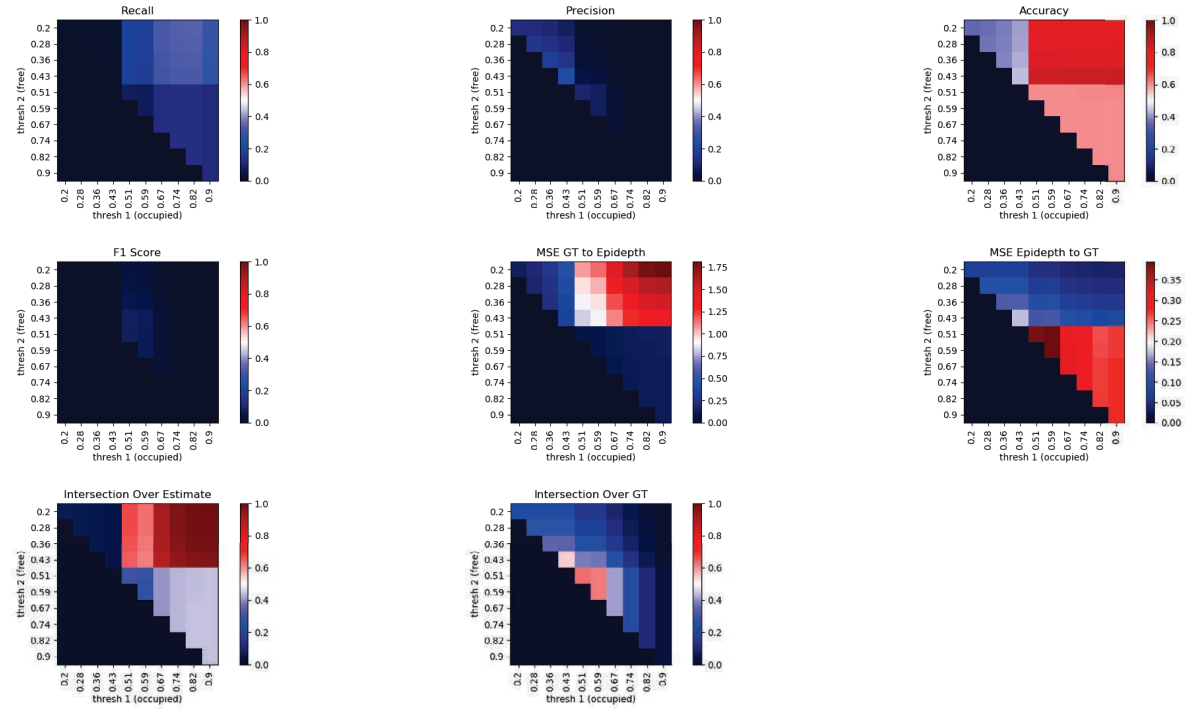


Figure 14: Measures for DS6.

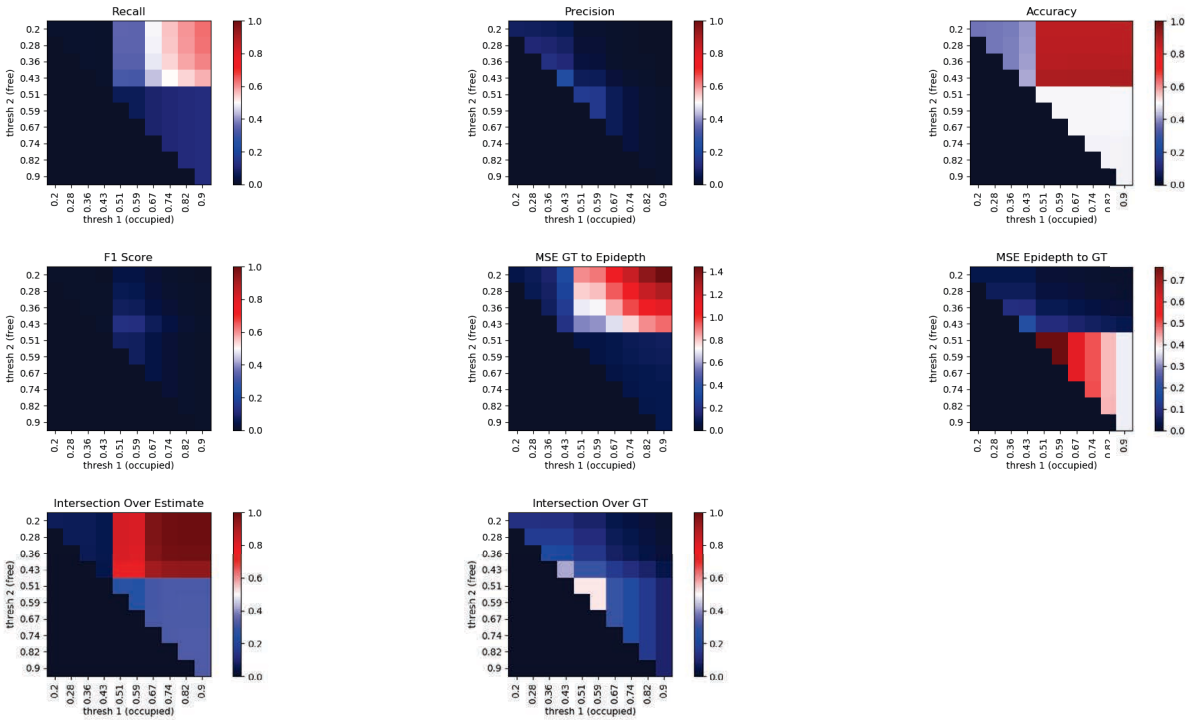


Figure 15: Measures for DS7.

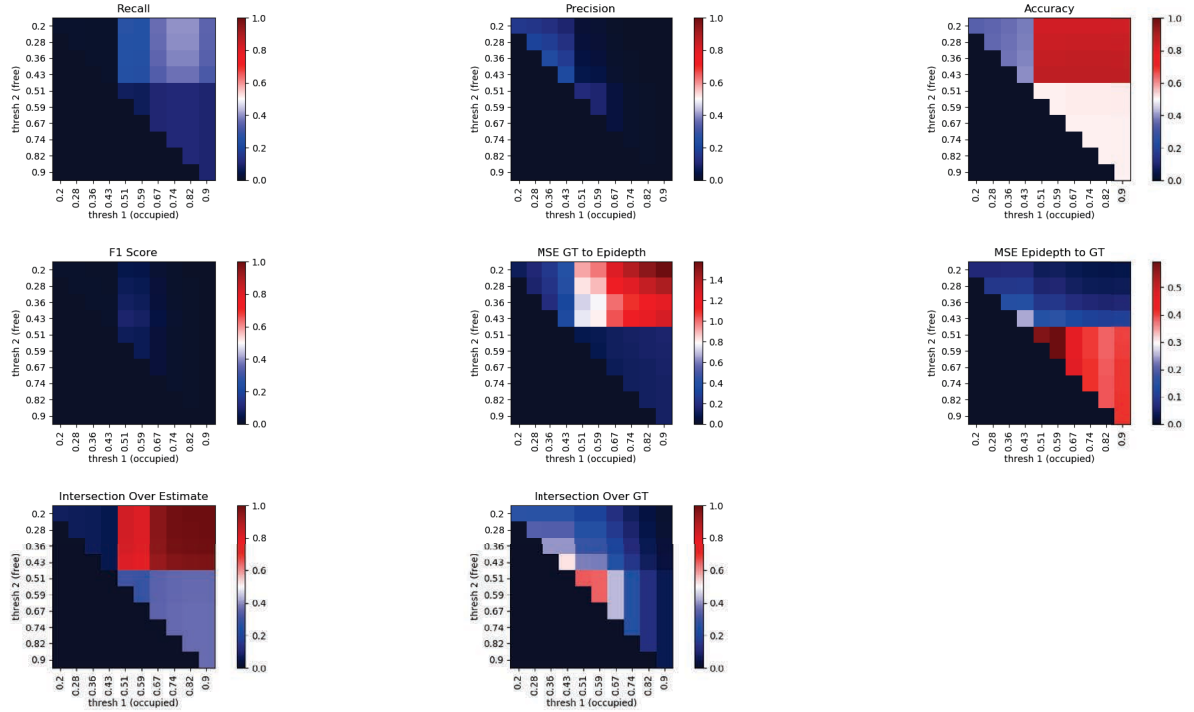


Figure 16: Measures for DS8.

Chapter 5

Title: Procedurally generated simulated datasets for aerial explosive hazard detection

Venue: SPIE Defense + Commercial Sensing

Date Published: May 30, 2022

PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Procedurally generated simulated datasets for aerial explosive hazard detection

Jeffrey Kerley, Aaron Fuller, Madeline Kovaleski, Peter Popescu, Brendan Alvey, et al.

Jeffrey Kerley, Aaron Fuller, Madeline Kovaleski, Peter Popescu, Brendan Alvey, Derek T. Anderson, Andrew Buck, James M. Keller, Grant Scott, Clare Yang, Ken E. Yasuda, Hollie A. Ryan, "Procedurally generated simulated datasets for aerial explosive hazard detection," Proc. SPIE 12116, Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XXIII, 1211611 (30 May 2022); doi: 10.1117/12.2618798

SPIE.

Event: SPIE Defense + Commercial Sensing, 2022, Orlando, Florida, United States

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

Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XXIII, edited by Jason A. Guicheteau, Chris R. Howle, Proc. of SPIE Vol. 12116, 1211611 · © 2022 SPIE · 0277-786X · doi: 10.1117/12.2618798

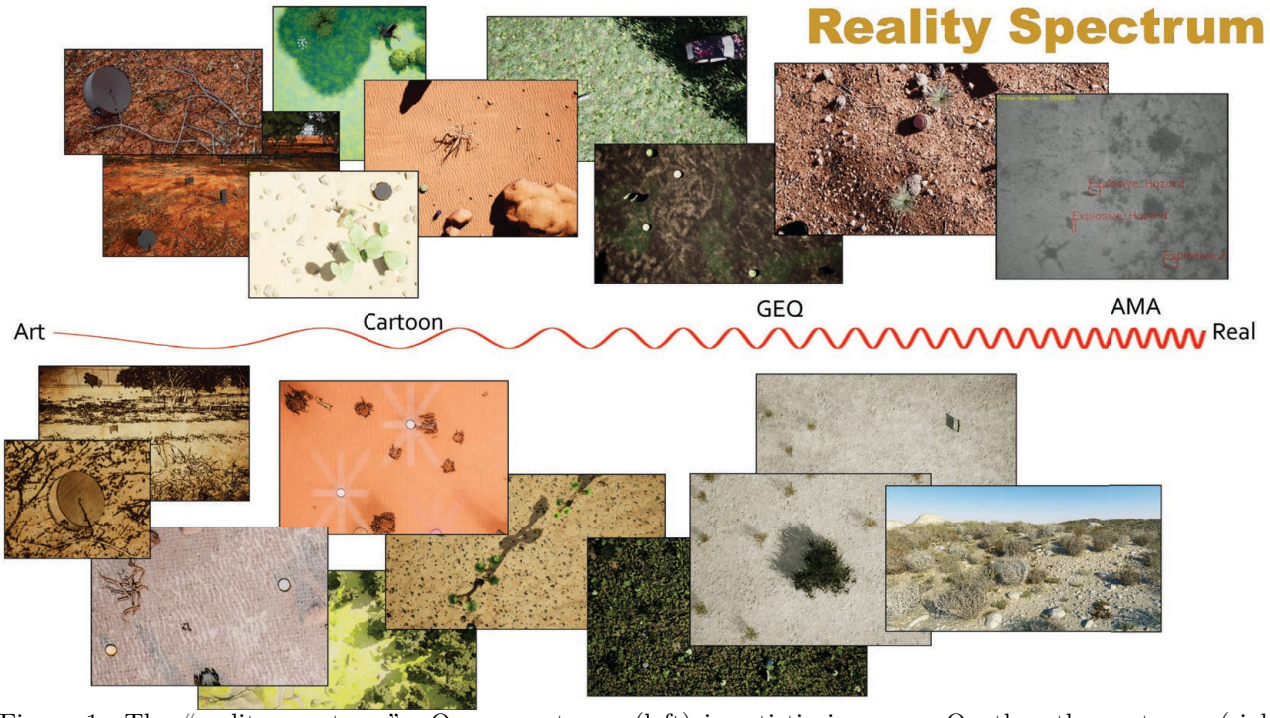


Figure 1: The "reality spectrum". On one extreme (left) is artistic imagery. On the other extreme (right) is real data. Example sample locations in the RS include cartoon and modern game engine quality data (not photo-realistic). Imagery shown are examples we produced using Epics Unreal Engine.

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.

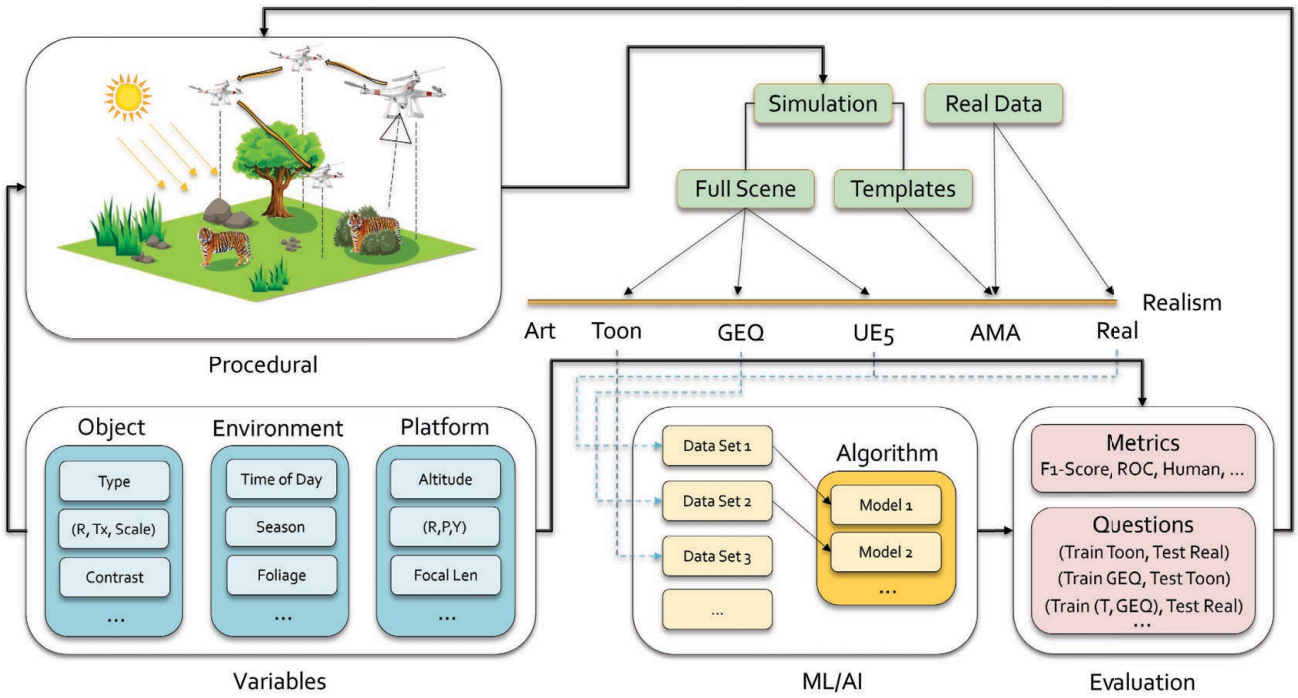


Figure 2: Illustration of this paper: (i) underlying domain variables that drive the task at hand; (ii) generation of data at different points in the "reality spectrum"; (iii) training ML/AI models relative to subsets of the data from the RS; (iv) evaluation; and (v) feedback to procedural scene generation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

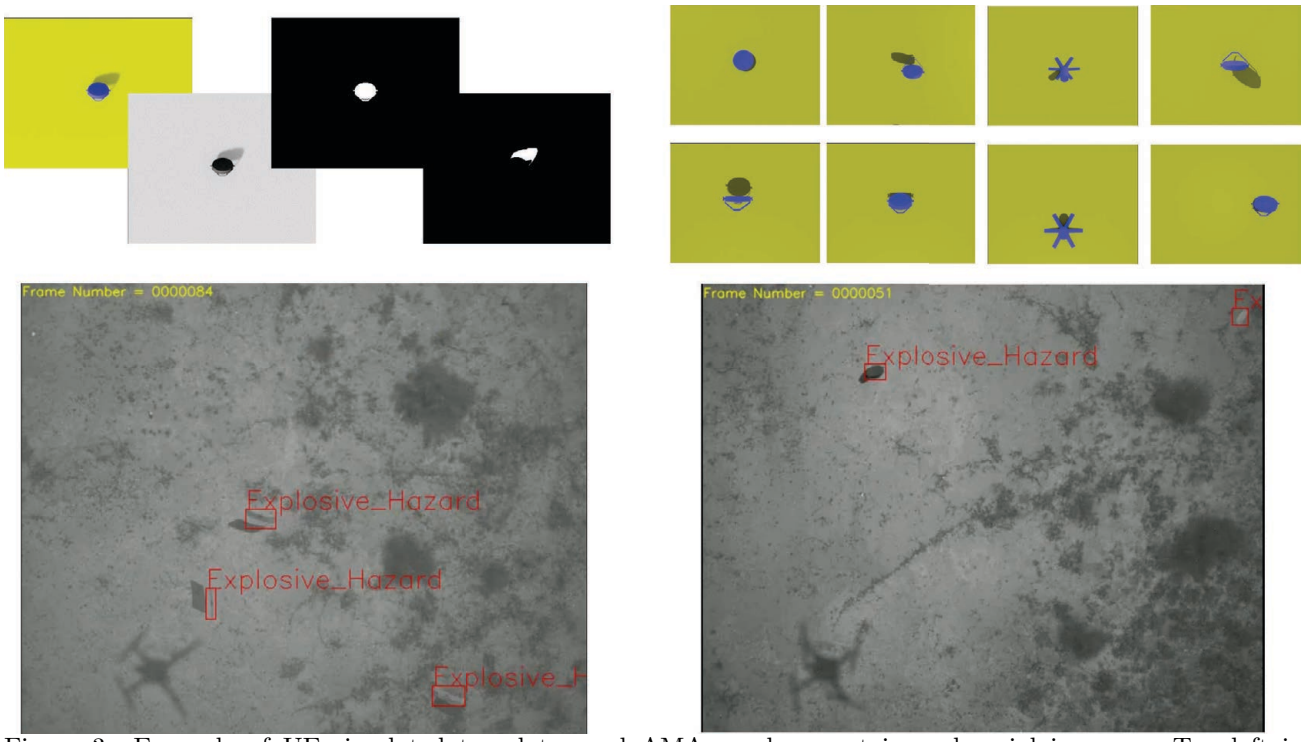


Figure 3: Example of UE simulated templates and AMA emplacement in real aerial imagery. Top left is a simulated false color RGB image (R=grayscale image, G=object pixels, B=shadow pixels) and individual channels. Top right is simulated objects at different camera poses and sun positions. Bottom left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

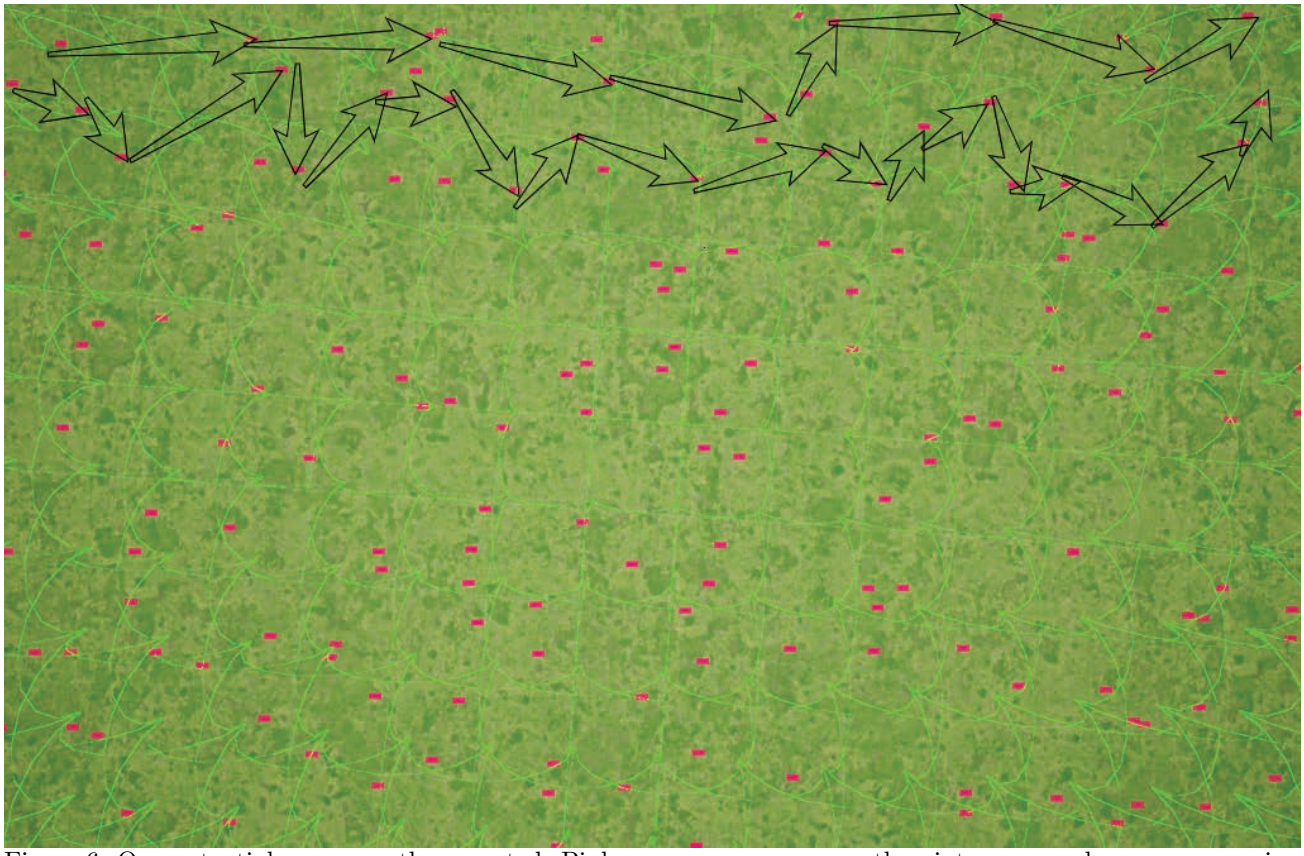


Figure 6: One potential camera path generated. Pink squares are camera path points, green spheres are spawning zones, and arrows are a camera path. As can be seen, this accounts for drone motion, in a randomly controlled manner versus perfect linear flight patterns between waypoints.

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.

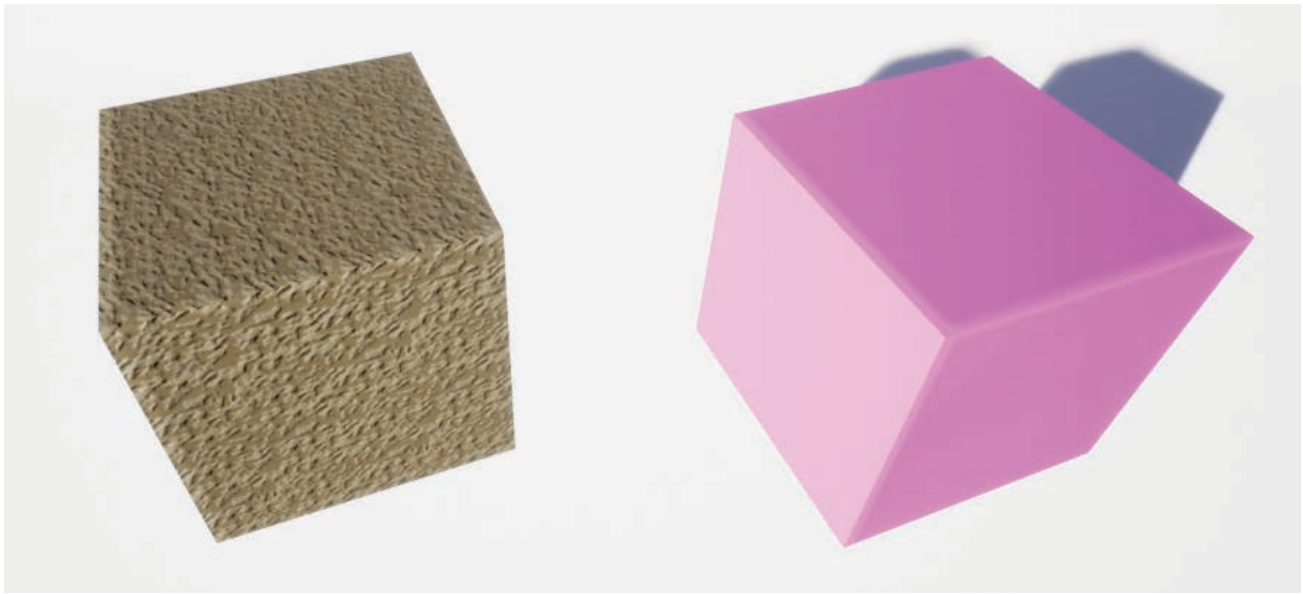


Figure 7: Differences between (left) GEQ and (right) TOON materials on an example UE object.

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.

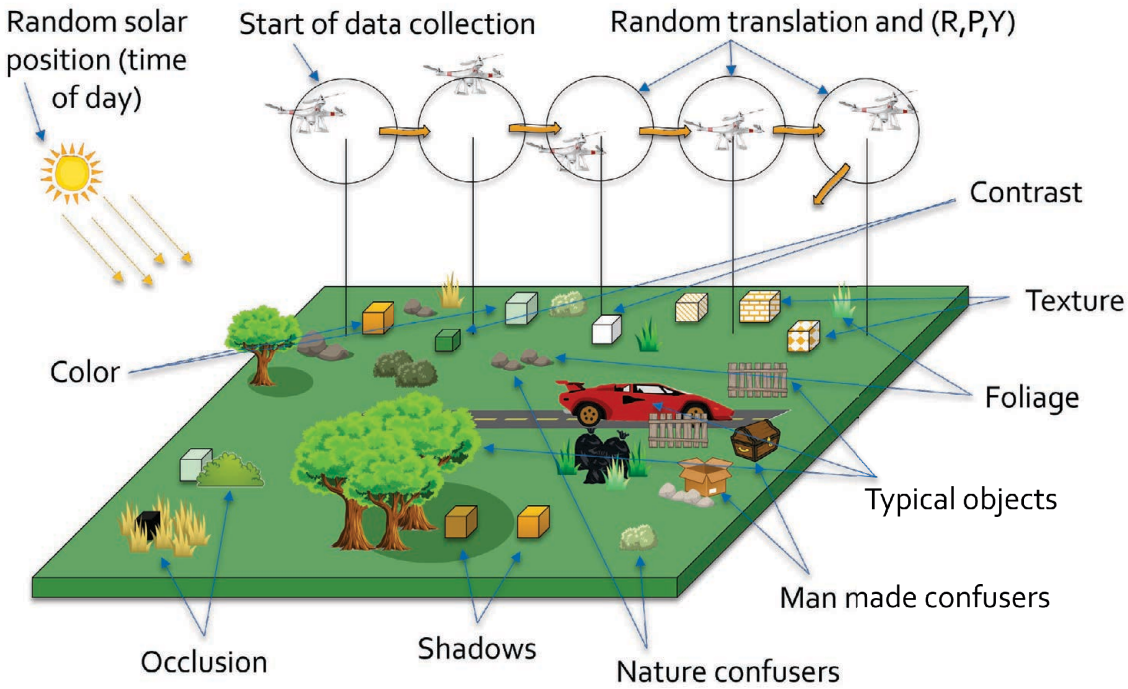


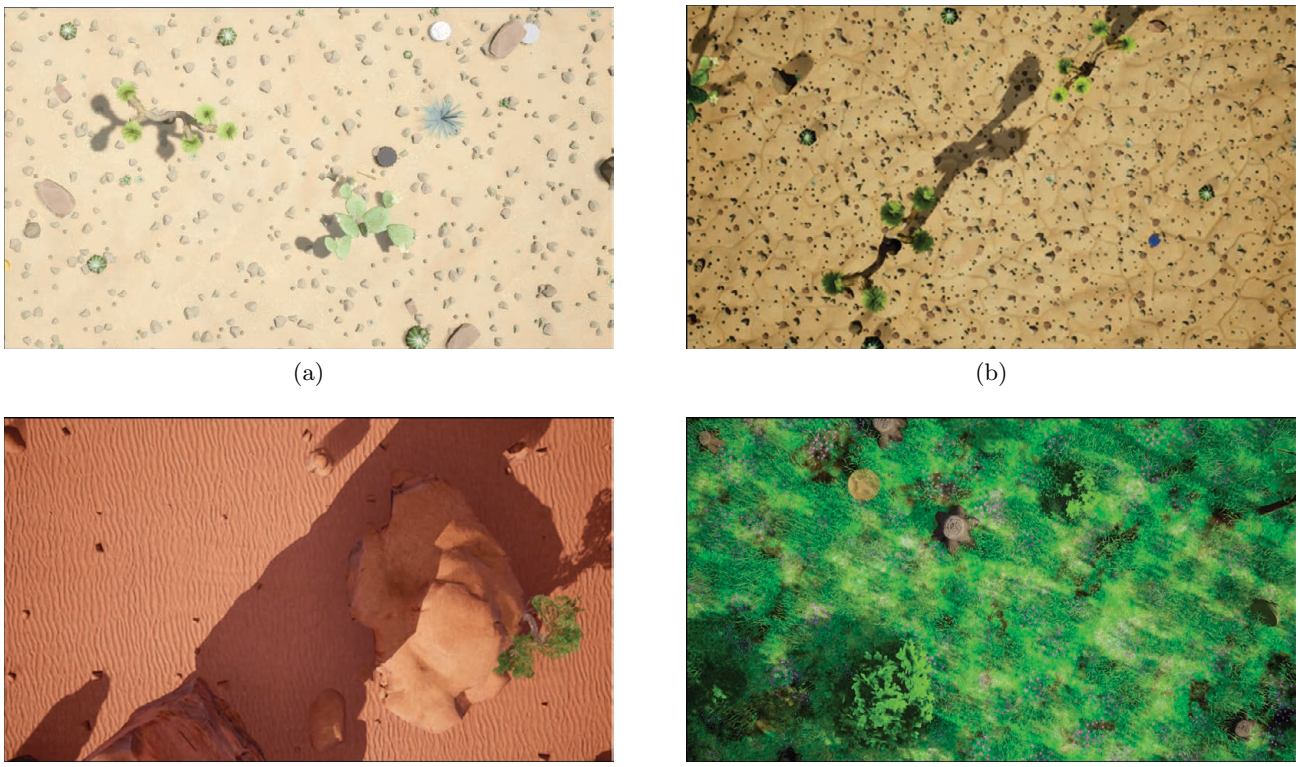
Figure 8: Illustration of our procedural map/scene generation and data collection in the UE. See Section 3.2 for full details. The main idea is to create a dataset that has a dense set of targets, close confusers (nature and man made), and common environment objects (nature and man made). As the primary 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

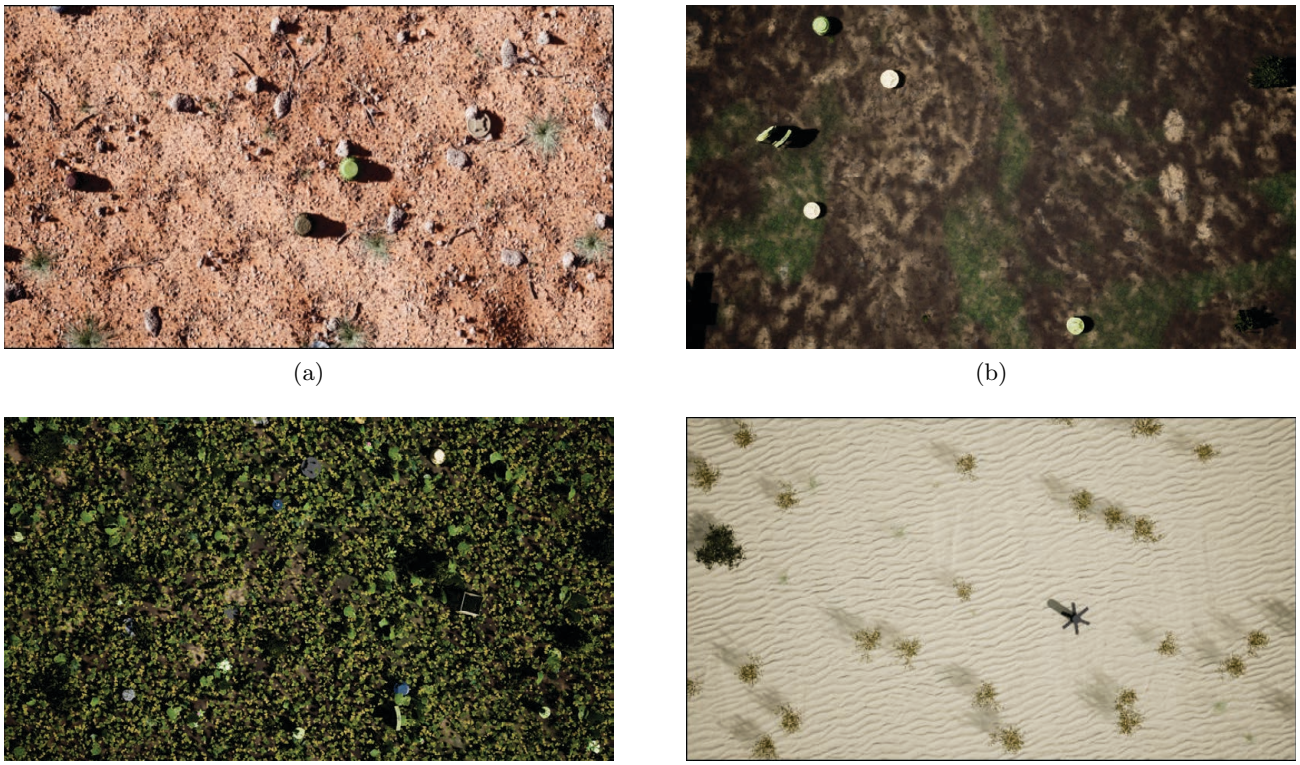
Herein, we explore the U-Net[38] neural network architecture as an EHD prescreener. While U-Nets have been extensively used for image semantic segmentation, they have also become the backbone for problems like passive ranging, deep diffractive neural networks and holography, and beyond. We selected a 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)

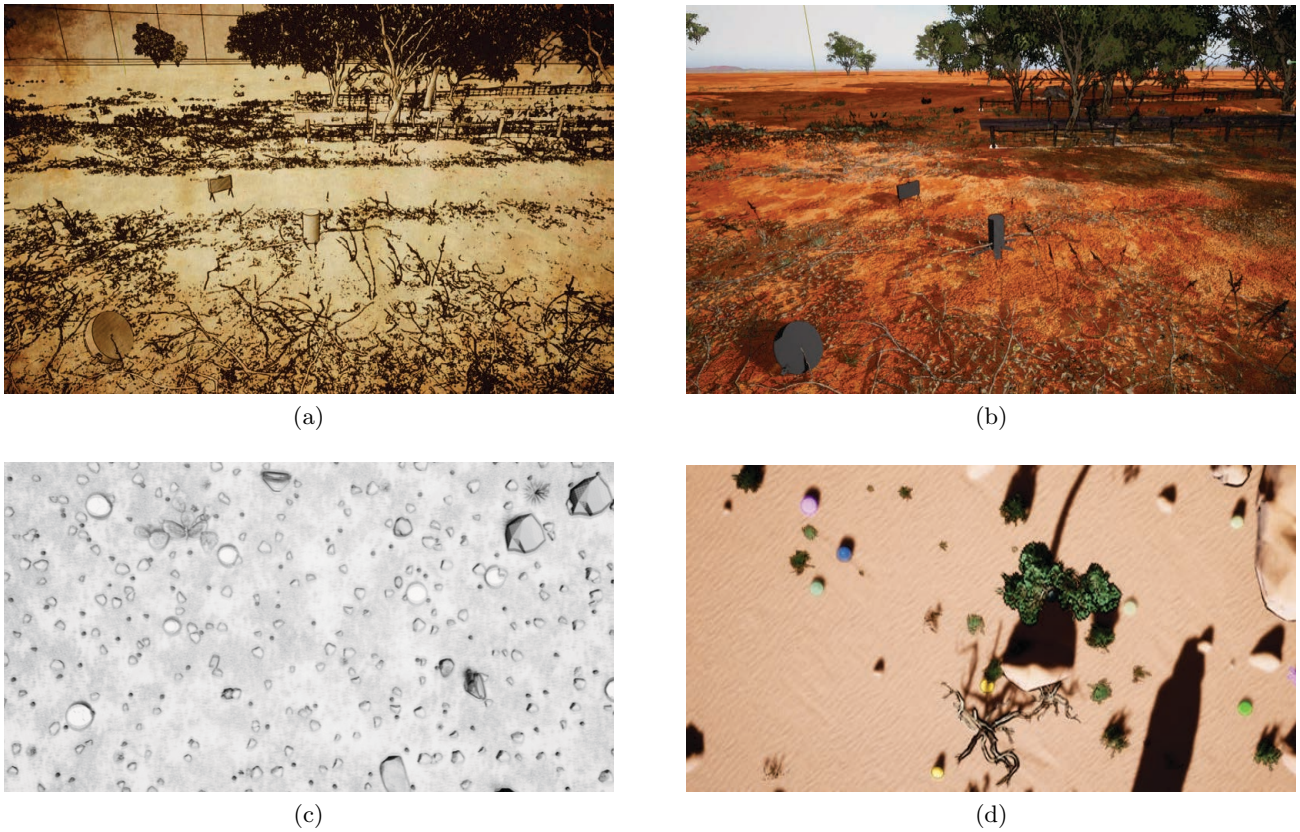
Figure 10: Example procedurally generated GEQ imagery for arid and forested environments.

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

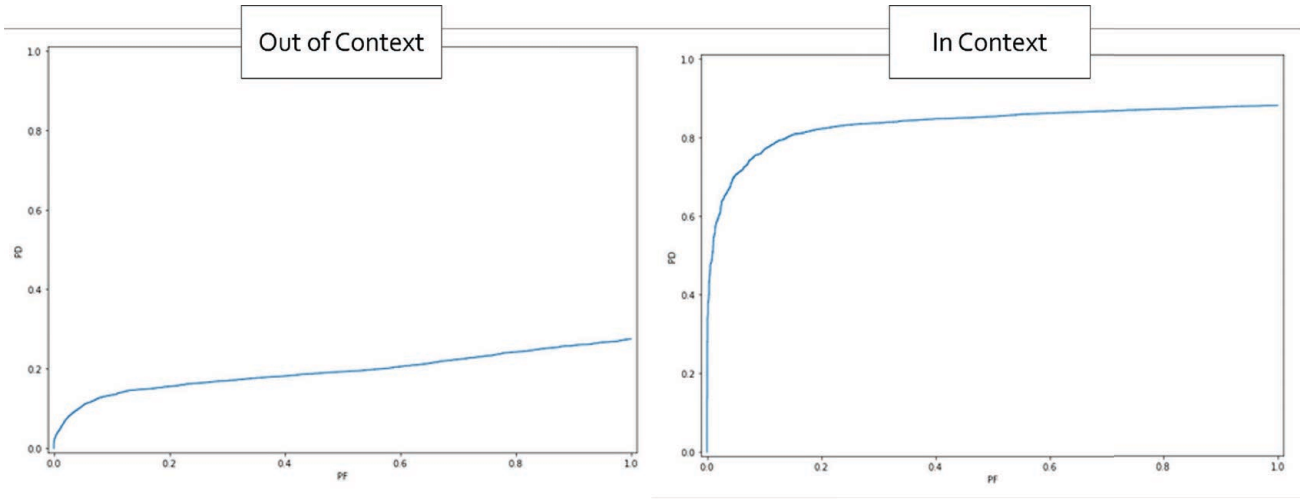


Figure 13: ROC curve results for a TOON model in "out of context" (train on solar noon and test across different times of day) and "in context" (train and test across different times of day). The y-axis shows probability of detection while the x-axis shows probability of false alarm. We do not report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

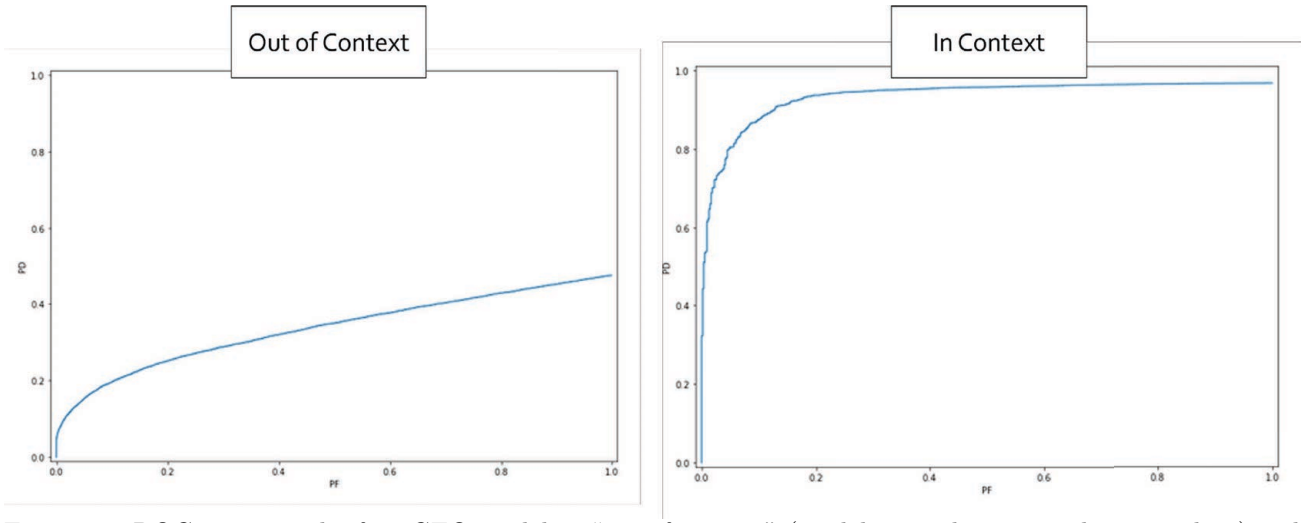


Figure 14: ROC curve results fora GEQ model in "out of context" (model trained on just solar noon data) and "in context" (model trained and tested across times of day).

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

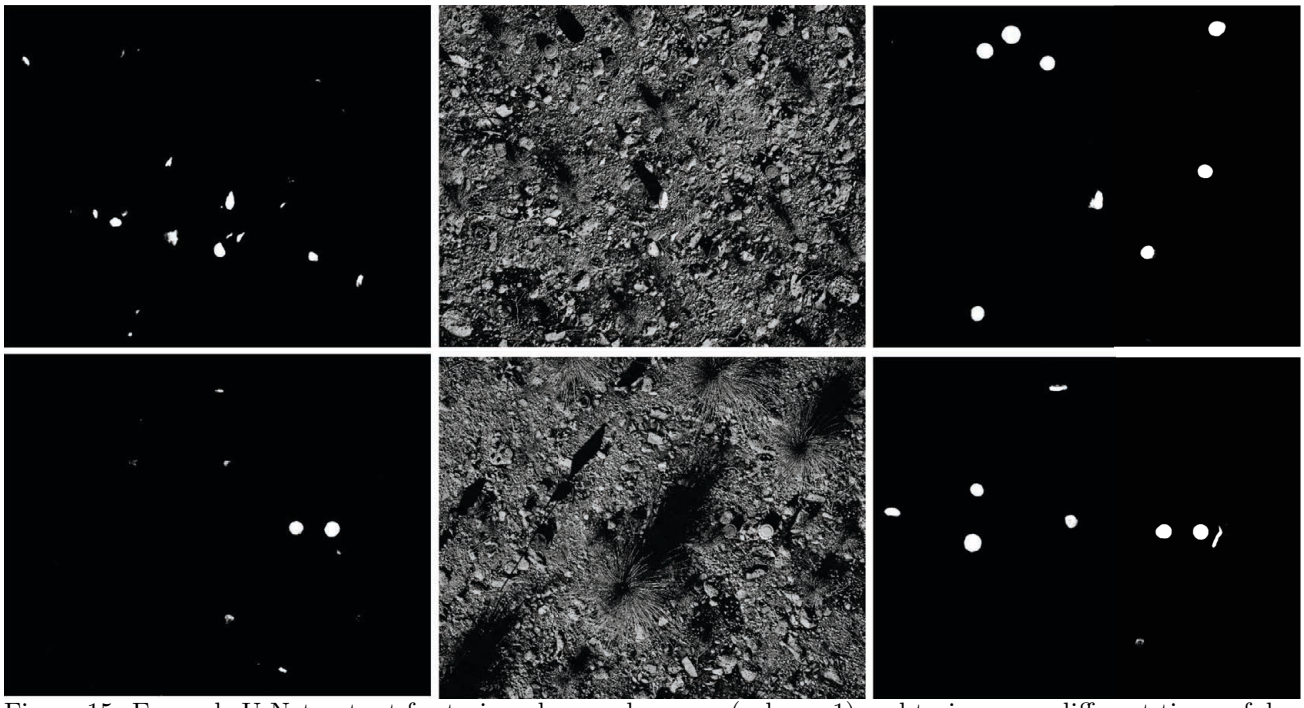


Figure 15: Example U-Net output for train only on solar noon (column 1) and train across different times of day (column 3) for a GEQ model. The middle column is example GEQ images and rows show corresponding data.

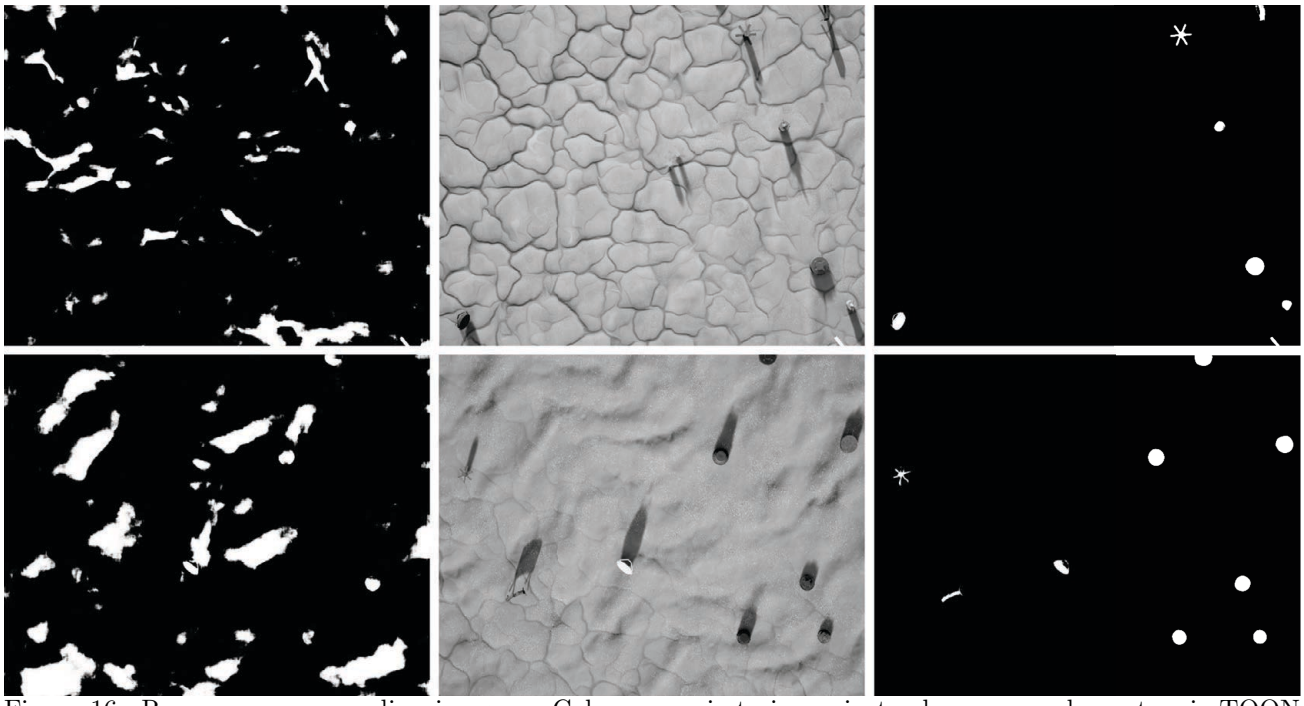


Figure 16: Rows are corresponding imagery. Column one is train on just solar noon, column two is TOON imagery, and column three is train and test across all times of day for a TOON model.

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

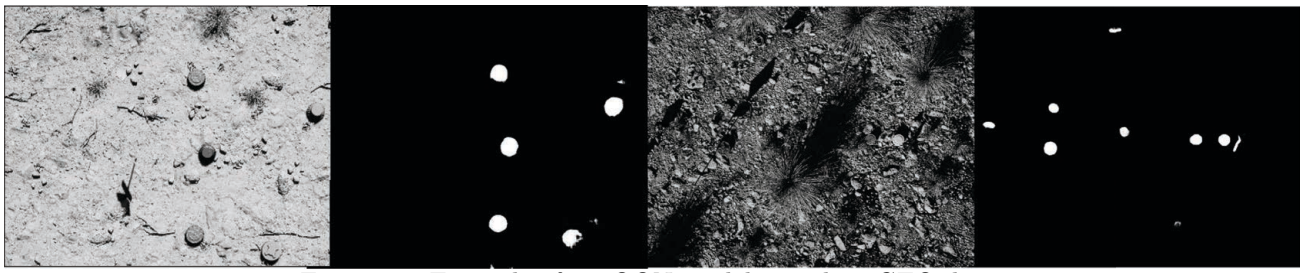


Figure 17: Example of a TOON model tested on GEQ data.

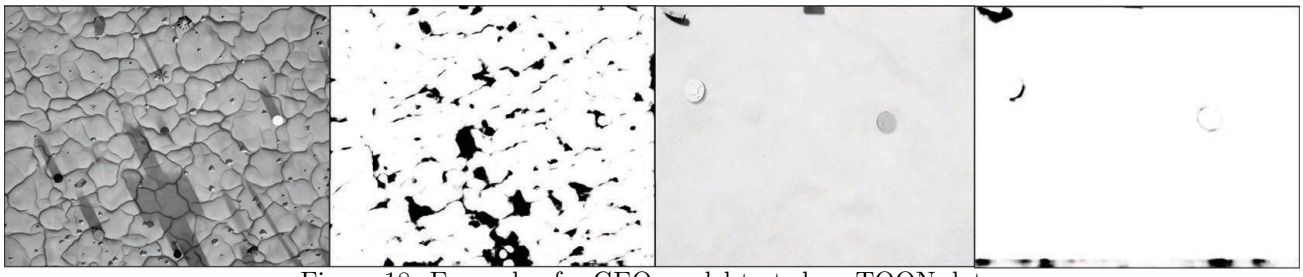


Figure 18: Example of a GEQ model tested on TOON data.

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.

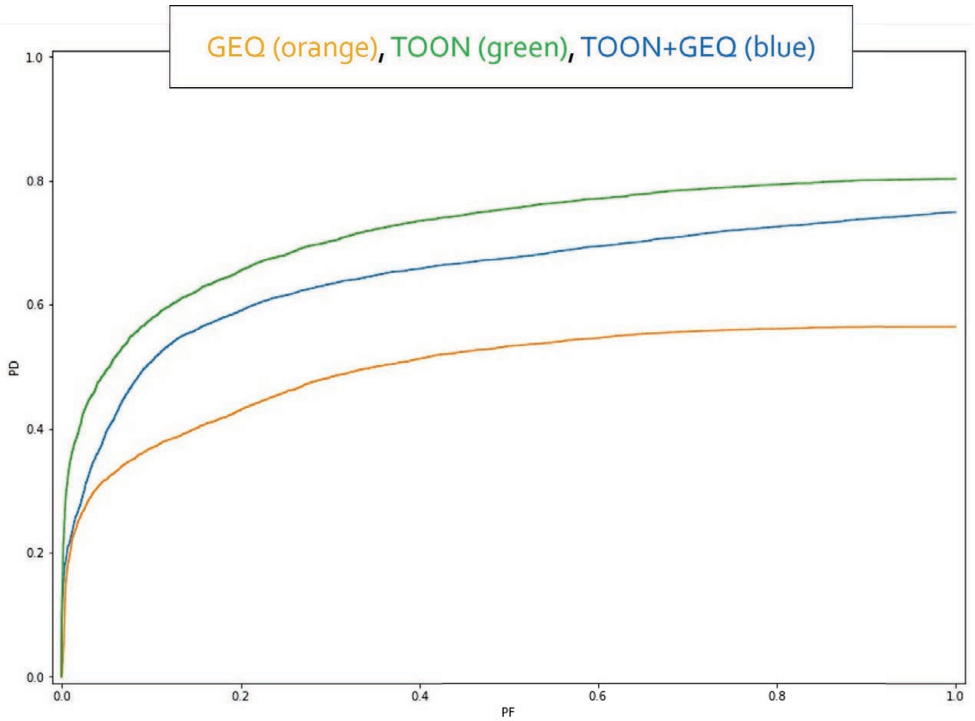


Figure 19: Quantitative ROC curve results showing how well TOON, GEQ, and TOON+GEQ trained models abstract to highly stylized or artistic imagery.

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.

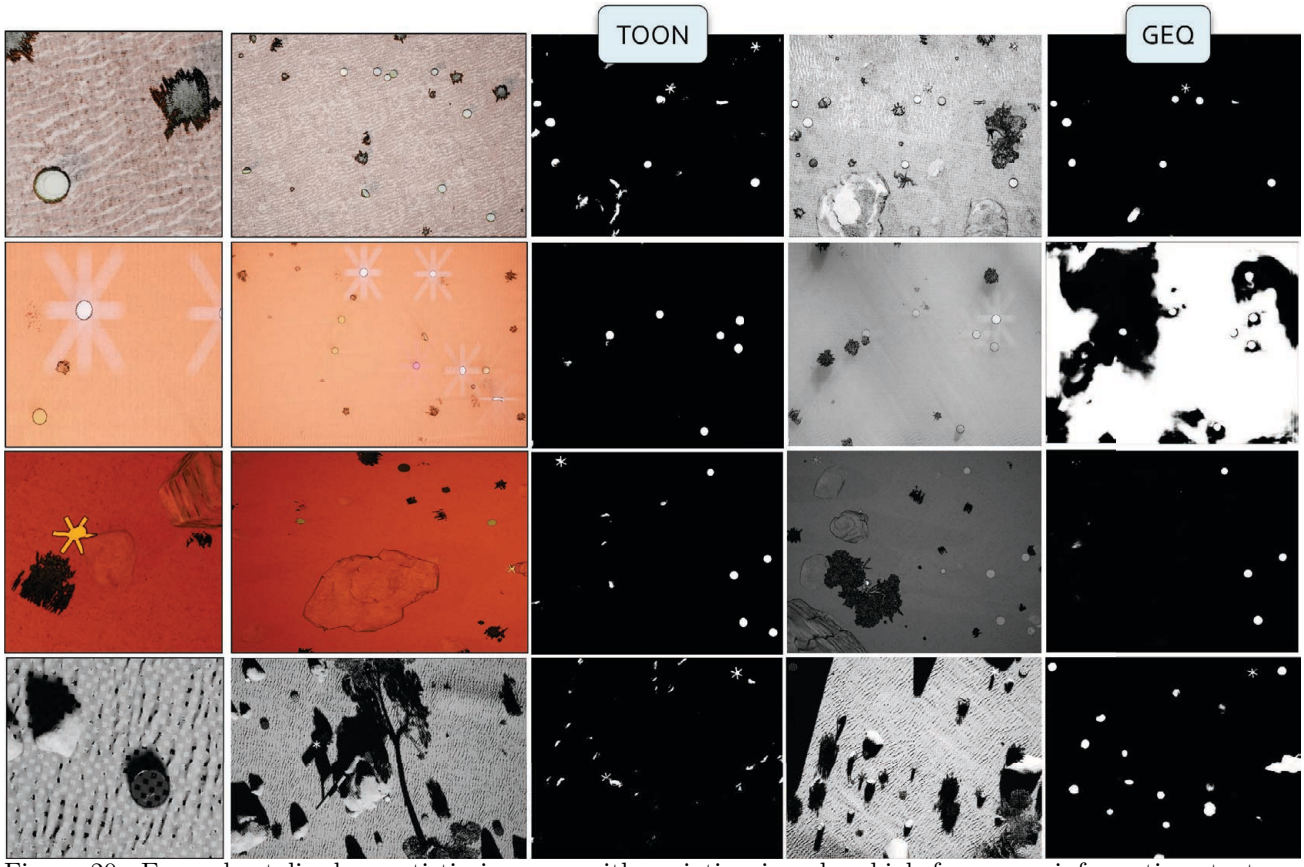


Figure 20: Example stylized or artistic imagery with variation in color, high frequency information, texture, specular highlights, etc. Column two are example effects and column one is zoomed in regions of interest. Columns three, four, and five are corresponding U-Net result pairs. Column three is TOON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

Chapter 6

Title: Explosive hazard pre-screener based on simulated data with perfect annotation and imprecisely labeled real data

Venue: SPIE Defense + Commercial Sensing

Date Published: May 30, 2022

Explosive hazard pre-screener based on simulated data with perfect annotation and imprecisely labeled real data

Madeline Kovaleski, Aaron Fuller, Jeffrey Kerley, Brendan Alvey, Peter Popescu, et al.

Madeline Kovaleski, Aaron Fuller, Jeffrey Kerley, Brendan J. Alvey, Peter Popescu, Derek Anderson, Andrew Buck, James Keller, Grant Scott, Clare Yang, Ken E. Yasuda, Hollie A. Ryan, "Explosive hazard pre-screener based on simulated data with perfect annotation and imprecisely labeled real data," Proc. SPIE 12116, Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XXIII, 121160X (30 May 2022); doi: 10.1117/12.2618792

SPIE.

Event: SPIE Defense + Commercial Sensing, 2022, Orlando, Florida, United States

Explosive Hazard Pre-Screener Based on Simulated Data with Perfect Annotation and Imprecisely Labeled Real Data

Madeline Kovaleski[a], Aaron Fuller[a], Jeffrey Kerley[a], Brendan Alvey[a], Peter Popescu[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

Datasets with accurate ground truth from unmanned aerial vehicles (UAV) are cost and time prohibitive. This is a problem as most modern machine learning (ML) algorithms are based on supervised learning and require large and diverse well-annotated datasets. As a result, new creative ideas are needed to drive innovation in robust and trustworthy artificial intelligence (AI) / ML. Herein, we use the Unreal Engine (UE) to generate simulated visual spectrum imagery for explosive hazard detection (EHD) with corresponding pixel-level labels, UAV metadata, and environment metadata. We also have access to a relatively small set of real world EH data with less precise ground truth – axis aligned bounding box labels – and sparse metadata. In this article, we train a lightweight, real-time, pixel-level EHD pre-screener for a low-altitude UAV. Specifically, we focus on training with respect to different combinations of simulated and real data. Encouraging preliminary results are provided relative to real world EH data. Our findings suggest that while simulated data can be used to augment limited volume and variety real world data, it could perhaps be sufficient by itself to train an EHD pre-screener.

Keywords: machine learning, semantic segmentation, u-net, simulation, unreal engine, pre-screener

1. INTRODUCTION

Obtaining real world datasets from *unmanned aerial vehicles* (UAVs) with quality ground truth (annotation and metadata) for training supervised learning-based *machine learning* (ML) algorithms is time and cost prohibitive. An enormous amount of effort is required to plan and coordinate a data collect. Each collect requires a skilled pilot, charged batteries, flight plans, a working UAV, acceptable weather for flying, etc. In addition, each target emplacement should be well documented with accurate GPS positions and pictures of the surrounding context. For training supervised learning ML models and for scoring, annotations are required. Accurately labeling images of EHs frequently requires the assistance of skilled experts who painstakingly draw *axis aligned bounding boxes* (AABBs) and assign class labels to objects*. Furthermore, the global COVID-19 pandemic 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. Each of these requirements and conditions places a bottleneck on the volume of quality annotated data available for training and testing.

Herein, we circumvent the real world data bottleneck by using simulated data to train, evaluate, and understand a ML-based *explosive hazard detection* (EHD) pre-screener. Specifically, our current article makes the following two contributions. First, we explore the use of modern game engine tools with perfect pixel-level ground truth and metadata to train a *neural network* (NN) based semantic segmentation model for EH pre-screening. More specifically, we investigate *cartoon* (TOON) and current generation video *game engine quality* (GEQ) rendering. These datasets possess different levels of visual information and they provide a rich platform to explore

*It is rare to get pixel-level annotations for real world aerial datasets. This is largely due to how much effort is involved and many aerial payloads have high frame rates and image resolutions (thousands of pixels in each dimension).

Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XXIII, edited by Jason A. Guicheteau, Chris R. Howle, Proc. of SPIE Vol. 12116, 121160X · © 2022 SPIE · 0277-786X · doi: 10.1117/12.2618792

Table 1: Notations and Acronyms

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

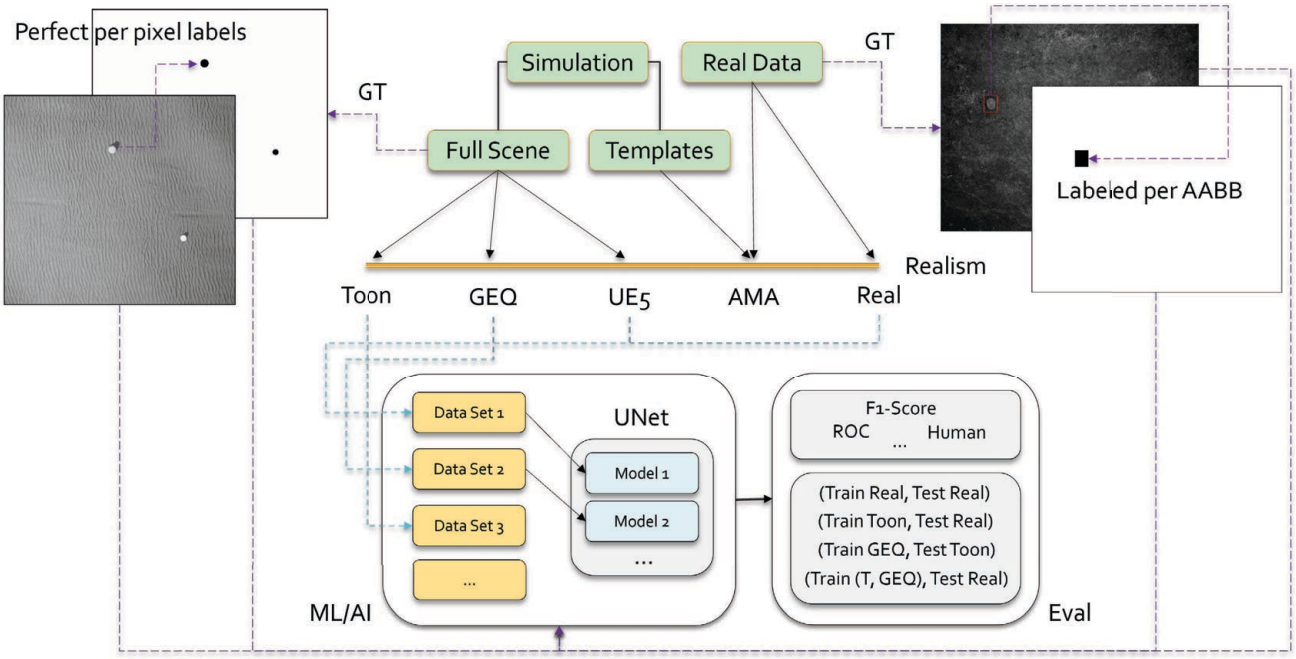


Figure 1: High-level illustration of this article. Simulated data is used to train a per-pixel pre-screener for EHD. While simulation yields perfect per-pixel ground truth, real data has ambiguous AABB labels. The first objective of this article is to explore the performance of neural segmentation networks for EHD pre-screening. The second objective is to explore performance relative to different combinations of perfectly labeled simulation data and less than ideal human labeled real data.

questions surrounding model generalizability. Second, we explore how to combine perfectly labeled simulation data with ambiguously labeled real data. The idea is to determine if simulated data can help a machine learn improved features by discovering how to make better use of ambiguously labeled human data that unfortunately possess both target and background features.

Figure 1 is a high-level illustration of the proposed article and Table 1 is notation. In Section 2, we summarize related work. Section 3 details our pre-screener, Section 4 discusses simulated datasets, Section 5 documents real world datasets, and Section 6 discusses experiments and results.

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,[1] downward looking,[2] and even side looking.[3] The reader can refer to[4] 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.

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.[5] In 2015, Chen et al.[6] used *The Open Racing Car Simulator* (TORCS)[7] 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.[8] In 2018, Martinez et al. proposed UnrealROX,[9] which used UE4 to create realistic looking indoor scenes for robots to interact with objects in simulation. In 2018, Muller et al. explored Sim4CV[10] in UE4 for generic CV research. In 2020, Drouin et al.[11] used real ortho-photos to simulate aerial data collections, and in 2021, Nouduri et al.[12] generated synthetic views from a dense 3D point cloud. In Alvey, et al.,[13] we proposed UE and AirSim based frameworks and workflows, 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,[14] and augmented reality. While Sim4CV is the closest to our current article and previous UE-based work,[13] the content used and imagery produced is not photorealistic, no detailed workflows are outlined, no supplemental online training is available, and results are simulator-focused vs. real world and simulation cross-validated.

In,[15] see Figure 2, we previously put forth a process called *altitude modulated augmentation* (AMA). AMA uses UE to render images of objects in different contexts, allowing us to render target EHs in different object poses, camera relative poses, and solar positions (which impact illumination and shadows). As Figure 2 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[15] 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.

[†] https://github.com/MizzouINDFUL/UEUAVSim

[‡] https://bit.ly/MizzouINDFUL

[§] 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.

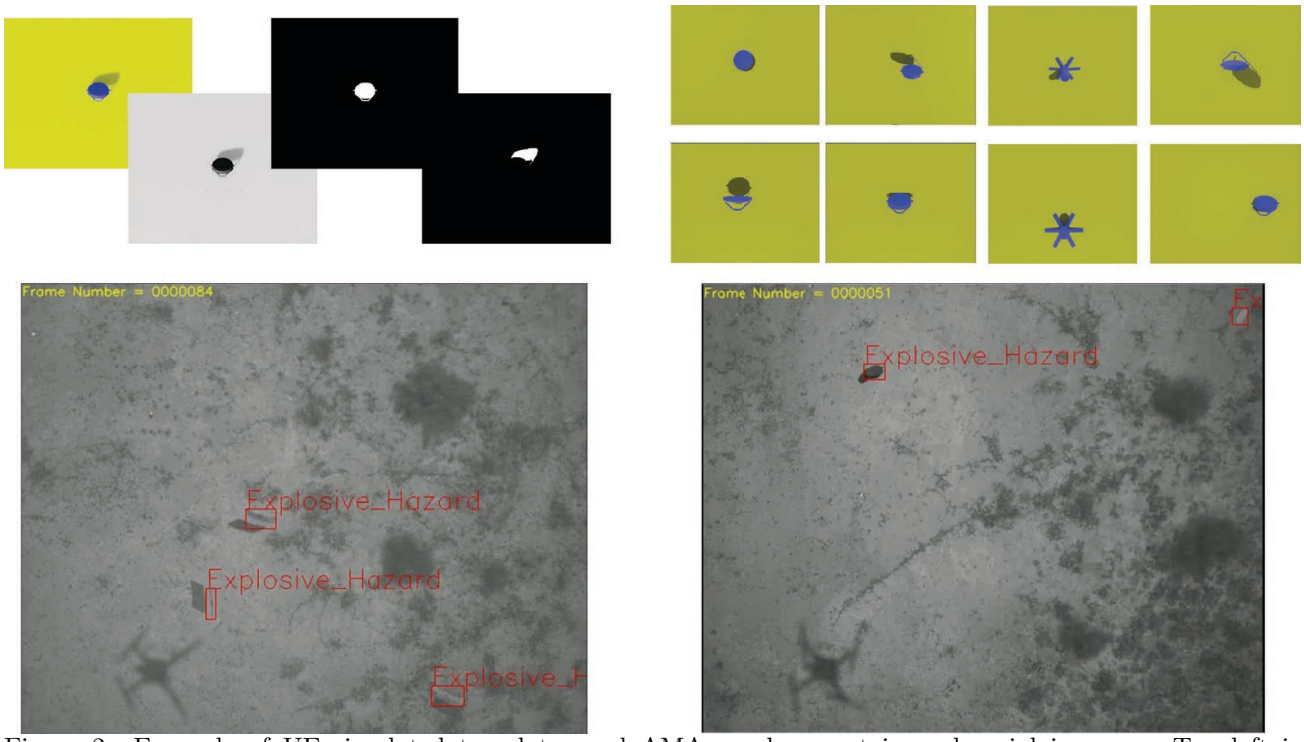


Figure 2: Example of UE simulated templates and AMA emplacement in real aerial imagery. Top left is a simulated false color RGB image (R=grayscale image, G=object pixels, B=shadow pixels) and individual channels. Top right is simulated objects at different camera poses and sun positions. Bottom left and right is AMA placing the above simulated templates into real aerial data (EHs highlighted as red AABBs).

AMA can be regarded as a type of data augmentation. As shown in,[15] the performance of AMA is notably higher than what is currently 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 that AMA is 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 volume and variety of training data with respect to a richer set of environments, environmental conditions, and platform (drone and camera) contexts. Furthermore, we desire supervised data with complete pixel-level labels.

3. EHD PRE-SCREENER

In this section we discuss our EHD pre-screener. In general, a pre-screener is a process (algorithm, feature, etc.) that nominates *alarms* (candidate targets) in data. Ideally, a pre-screener should have near a 100% *positive detection rate* (PDR) which sometimes comes at the expense of a relatively high *false alarm rate* (FAR). Strong classifiers are usually not considered pre-screeners. Most pre-screeners are relatively lightweight, can be run efficiently, and ideally result in as high of a PDR as possible; perhaps a rate comparable to that of a human.

Herein, we explore the U-Net[16] neural net architecture as an EHD pre-screener. While U-Nets have been extensively used for image semantic segmentation, they have also become the workhorse for problems like passive ranging, deep diffractive neural networks and holography, and beyond. We selected U-Nets 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[17]¶. Second, a U-Net can be engineered to model behaviors similar to attention in humans.[19] 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, then 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 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.

4. SIMULATED DATA - DATASET 1 (DS1)

As discussed above, this article is focused on the use of simulated data. The goal is to have 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,[20] VANE and ANVEL,[21] etc., we focus on UE for several reasons. It has proven longevity (decades of R&D), wide integration of assets (3D models, textures, etc.), simplicity of use and excellent online documentation. UE also supports high quality global illumination and realtime rendering capabilities (see NVIDIA *deep learning super sampling* (DLSS)). Historically, UE was created for video games, but it has found uses in film,[22] computer graphics,[23] architecture,[24] and beyond. Most importantly, high fidelity custom dynamic scenes can be conveniently purchased or produced manually in little time. 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 available, e.g., Quixel megascans 3D scanned real world objects and materials[25] and free military 3D objects,[26] as well as supporting tools for procedural content generation, e.g., large scale terrain generation with Instant Terra (UE plugin[27]) and Houdini for object/geometry, texture, terrain, animation, city generation and beyond (see UE Houdini plugin[28]). Furthermore, UE has a rich support for models, materials, effects, and more on their UE Marketplace,[29] including the artistic shaders we used herein (cel shader[30] and artistic pen and paper shader[31]).

4.1 Generated Data

The reader can refer to our 2022 SPIE article[32] for details about our procedural framework and workflow in UE to generate EH environments and automated data collections. In this section, we briefly summarize our simulated datasets. DS1 (see Table 2) is made up of three sub-datasets, which are detailed next.

Manually Created UE EHD GEQ Data In 2021, we produced an initial dataset of 720 images in the UE (see Figure 3) to train object detection and localization algorithms[15] and explore XAI.[18] This dataset is rasterization rendering based, versus global illumination or ray traced, and textures and objects are relatively low detail, e.g., no high 3D model poly counts or *physically-based rendering* (PBR) textures (albedo, roughness, normal maps, specular, etc.). The reader can see in Figure 3 that the result is similar to last generation video games. Details like bill boarded foliage and simple coloring, illumination, and shadow are present.

¶The reader can refer to[14,15,18] 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/

Table 2: Simulated Dataset 1 (DS1)

Variable	Description
Number of Images	8720 (8000 images from 2022 and 720 from 2021 collections)
Environments	arid
Foliage	rocks, bushes, trees, stumps, grass, flowers, cacti, branches
Background Texture	sand, beach, grass, rocky
Clutter	vehicles, signs, fences, tires, helicopters, tents, small buildings, crates, barrels, cones, umbrellas, trashbags, cans of soup
Drone Altitude	$[20, 40]$ meters
Drone Pose	Nadir with random pitch perturbation (± 5 °), random yaw, no roll

Procedurally Created UE EHD GEQ Data In 2022, we generated new UE EHD GEQ imagery with our procedural UE algorithm,[32] see Figure 4. Improvements to our 2021 imagery are as follows. 3D models were more detailed (higher geometry count), textures were PBR-based, and global illumination was used. From a quality standpoint, object shape, texture, color, illuminations, and shadows are much more photo-realistic. A total of 4,000 images were generated from four procedural scenes. Two scenes were captured around solar noon (2,000 images), one was captured in the early morning (1,000 images) and one in the late afternoon (1,000 images). The reason for these collects was to sample variation with respect to lux and shadows.

Procedurally Created UE EHD TOON Data Figure 5 shows example procedurally generated[32] cartoon (TOON) UE imagery from our 2022 collect. Herein, we consider TOON to be simplified object geometry (thus shape), overly saturated color, and stylized or simplified texture. Overall, it is abstract in visual appearance. Our procedurally generated imagery is similar in style to a cartoon like Garfield or whimsical and fairy tale like the Legend of Zelda. Our intention is to produce imagery that places important features like shape, color, and contrast center stage. While this information is present in real imagery, it is often overshadowed by higher frequency texture information. A total of 4,000 images were generated for two solar noon sub-datasets (2,000 images), one early morning sub-dataset (1,000 images), and a late afternoon sub-dataset (1,000 images).

All procedurally generated data was subject to the general guidelines outlined in Table 2. This consisted of arid and forested locations, common foliage in those environments, related background (aka ground) textures, and clutter common to those environments and our task at hand. Our three EH objects were pseudo-randomly placed in a dense respect across our maps. The collection platform (camera on a drone) was pseudo-randomly varied within operating conditions similar to our real world data collections. Specifically, yaw was randomized, roll was fixed, and pitch was slightly varied around nadir. Overall, our goal was not to generate simulated datasets in support of a specific EHD scenario. For example, a data collection in a specific location at a specific time of day with respect to particular occlusion conditions, materials, etc. Our simulated 2021 and 2022 datasets are a random collection of data across related environments, flight conditions, and with respect to initial EH objects of interest. In future work, we will explore restricting the procedural simulation dataset variables to help train and/or test to answer questions of interest. One main objective of this paper is to see if a wide net of random aerial EH variable sampling works relative to training a pre-screener.

5. REAL DATA - DATASET 2 (DS2)

Our real dataset consists of images with EHs collected from a low-altitude UAV. This dataset has eight flights, referred to as runs hereafter. Four of the runs were collected on one day, three a month later, and the remaining three were collected some months later. Overall, we have an extremely limited amount and variety of labeled training and test data as we begin to explore aerial-based EHD. To make matters more complicated, the drone altitude, location, month, time of day, objects, object emplacements, and other variables changed across runs. This poses great complexity with respect to performing cross validation. Most of these collections are very different. This is one of the main factors driving our exploration of simulation. It is extremely hard, if not a bottleneck, to design and collect a wide range of data collections with respect to our combinatorial explosion

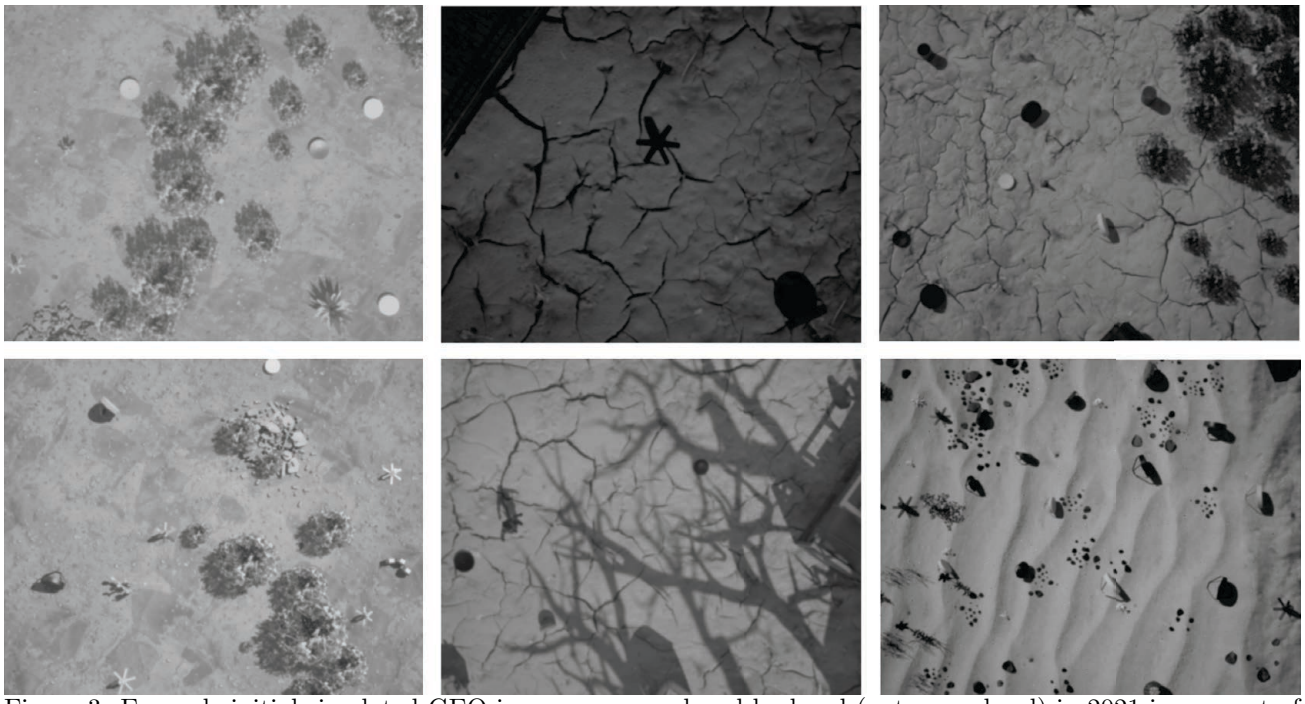


Figure 3: Example initial simulated GEQ imagery we produced by hand (not procedural) in 2021 in support of past object detection, localization, and XAI research.[15,18]

Figure 4: Example 2022 GEQ data – top is RGB, bottom is corresponding grayscale – produced procedurally[32] in support of this article. Compared to our old simulated imagery (see Figure 3), this new data has more realistic objects, textures, illumination, shadows, and the overall quality is more photo-realistic.

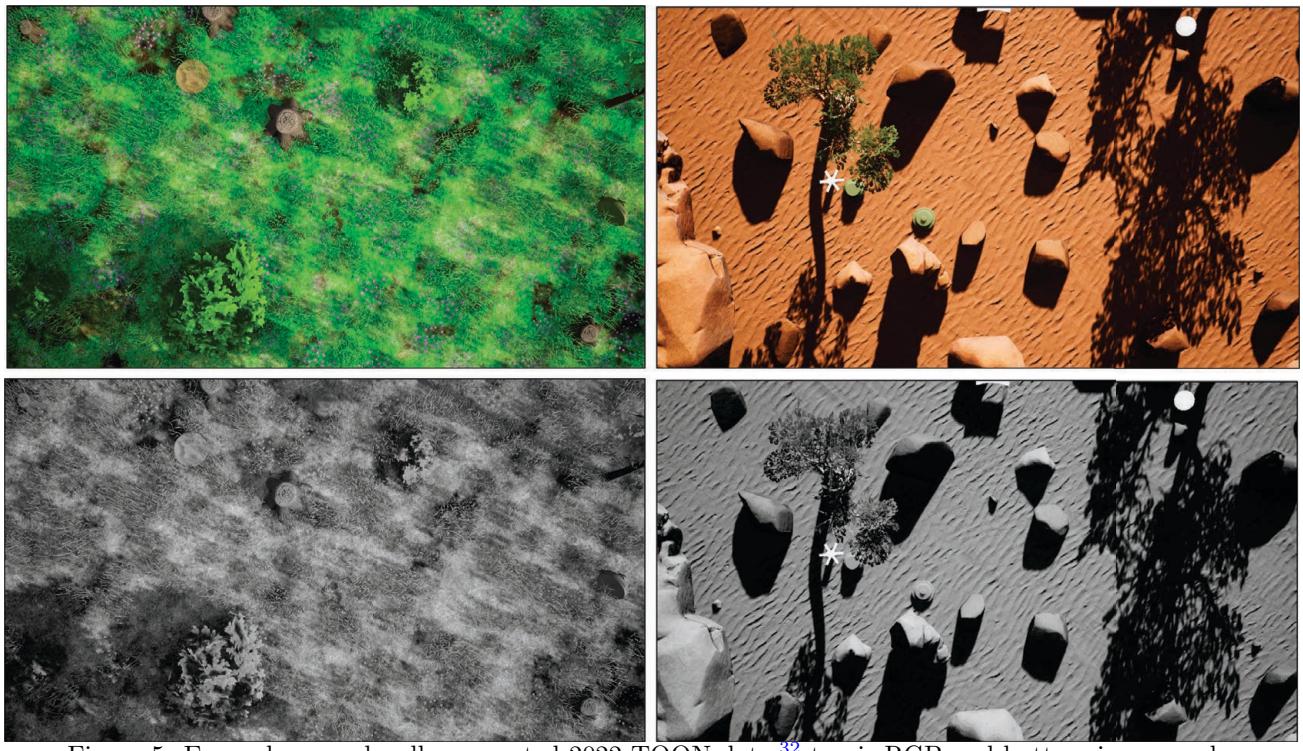


Figure 5: Example procedurally generated 2022 TOON data;[32] top is RGB and bottom is grayscale.

of object, environment, and platform/sensor EH variables**. Even if this data is collected, its an even larger undertaking to label it and provide ground truth. As a result of this real world predicament, we restrict our analysis herein to the following. We use seven runs for training and one run is left out for testing. This run (operating condition) was identified in conjunction with our collaborating funding source. The other runs are primarily of interest with respect to exploring how much we can vary these operating conditions.

Overall, a variety of EH targets exist in our dataset. Figure 7 shows examples of the four EH objects analyzed herein. Object 1 and Object 4 are mainly driven by shape, which is somewhat unique, and one is a smaller version of the other. Object 2 is driven by unique shape and Object 3, the most simple, has a very general shape, is not unique, and can easily be confused with other image content. In this article, we limit the scope of our analysis to grayscale vs. RGB imagery. That is, at this moment in time we are not considering spectral information. The resolution of the imagery explored herein is 640 x 512.

6. EXPERIMENTS AND PRELIMINARY RESULTS

This section outlines three experiments. Our objective is to learn from these preliminary results and develop intuition regarding best practices as we move forward with low-altitude aerial-based EHD. For example, should we primarily continue to request and wait on additional real world datasets to train our algorithms? Do we need to continue to refine our simulation before it can be used to train an ML/AI algorithm directly for real world data? Is our current simulation already sufficient, and if so what is the role of real world data? There are many intriguing questions in the intersection of these spaces. The point is, we are looking to better understand how well each of these processes are working individually and what utility is there in combining them.

Experiment 1 (E1): Baseline: Train on Real, Test on Real: The objective of E1 is to establish a baseline. What can be achieved using our limited variety and volume raw data?

**Example EH variables include the following. Object: type, size, pose, texture, contrast, occlusion, etc. Platform/sensor: camera/drone relative pose, altitude and standoff (number of pixels on target), speed, image resolution, focal length, spectrum, etc. Environment: season, time of day, region, foliage, clutter, etc.

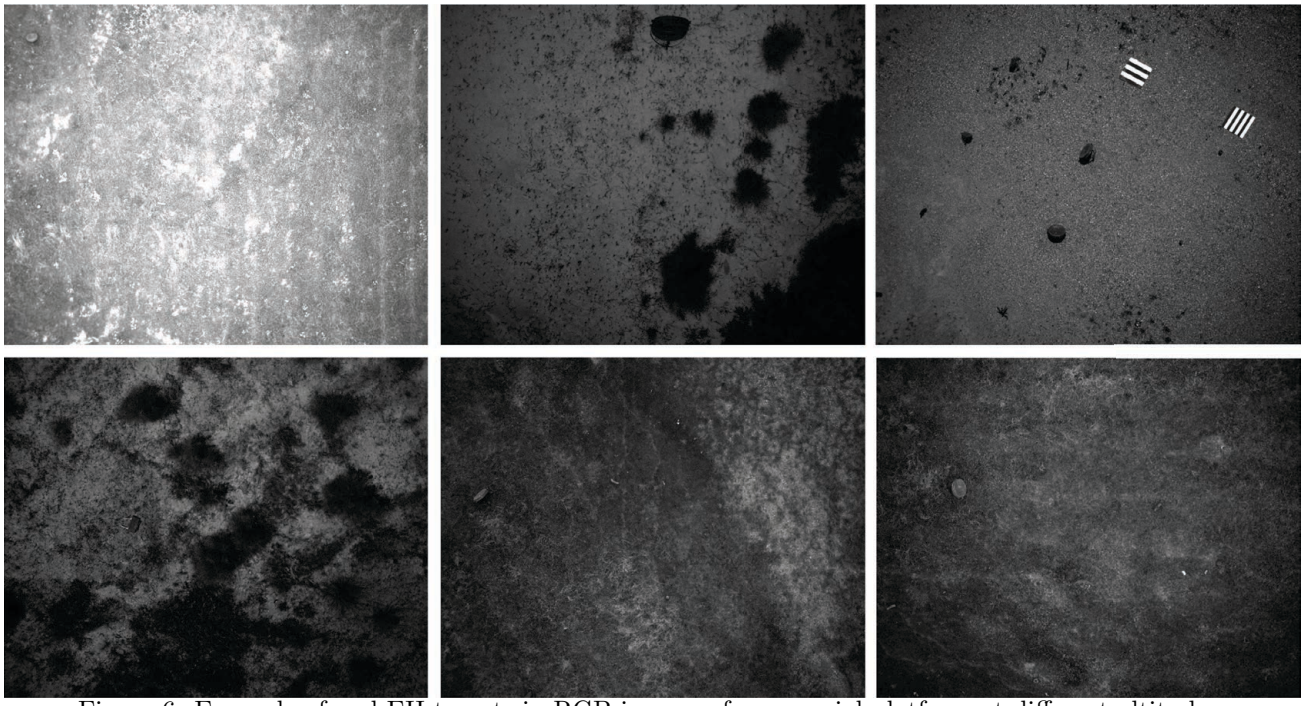


Figure 6: Example of real EH targets in RGB imagery for an aerial platform at different altitudes.

Experiment 2 (E2): Simulation: Train on Sim, Test on Real: The objective of E2 is to measure how well a simulated data trained U-Net works on real world test data. Our simulated data has the advantage that it has a much greater number of looks on target; altitudes, poses, illuminations, shadows, textures, etc. Its disadvantage is that it is synthetic and it could possess rendering artifacts (false features). Our synthetic data is a combination of arbitrarily constructed maps and procedurally random maps.

Experiment 3 (E3): Sim+Real: Train on Sim and Real, Test Real: The objective of E3 is to determine if synthetic data enhanced real data.

Figure 7 shows ROC curve results. Accompanying qualitative imagery – input imagery and U-Net results – are shown in Figure 8. The user can observe the following trends. The real data U-Net performs the worst. The second best model is Sim. It is important to note that the Sim U-Net was not fine tuned for our real data, i.e., specific site, time of day, textures, etc. Furthermore, our simulated objects are abstract representations of most of the real targets. They have the general shape and properties, but they are not scanned models nor specific instances of these objects. In particular, Object 2 was created in Blender using 1 cylinder and 7 rectangular boxes. Our Sim data also has different properties such as intensities, contrast, platform motion, sensor noise, etc. Regardless, the Sim U-Net is able to outperform the real data model.

The top performing model is Real+Sim. We do not have a mechanism to evaluate what data-driven features were learned; something generally associated with the left half of the U-Net. Nor do we have a way of measuring if these features are simulation image specific features, real data specific features, or features applicable to both. The next question is the right half of the U-Net. How are these features combined with respect to reasoning ({target, not target}) and producing the final segmentation image? We can only conjecture at this moment in time, but it appears that improved features were learned and/or the act of having both simulated and real data was useful with respect to the machine learning how to use these features on both sets of data relative to predicting and segmenting objects with similar shape and contrast. The Real+Sim model rises quickly and plateaus at less than 100% PDR due to EH objects at the edge of our imagery (see Object 1 in Figure 7).[††]

[††]Alarms near the edge an image are often discarded in detection systems as there is not enough information to reliably detect and discriminate them from clutter.

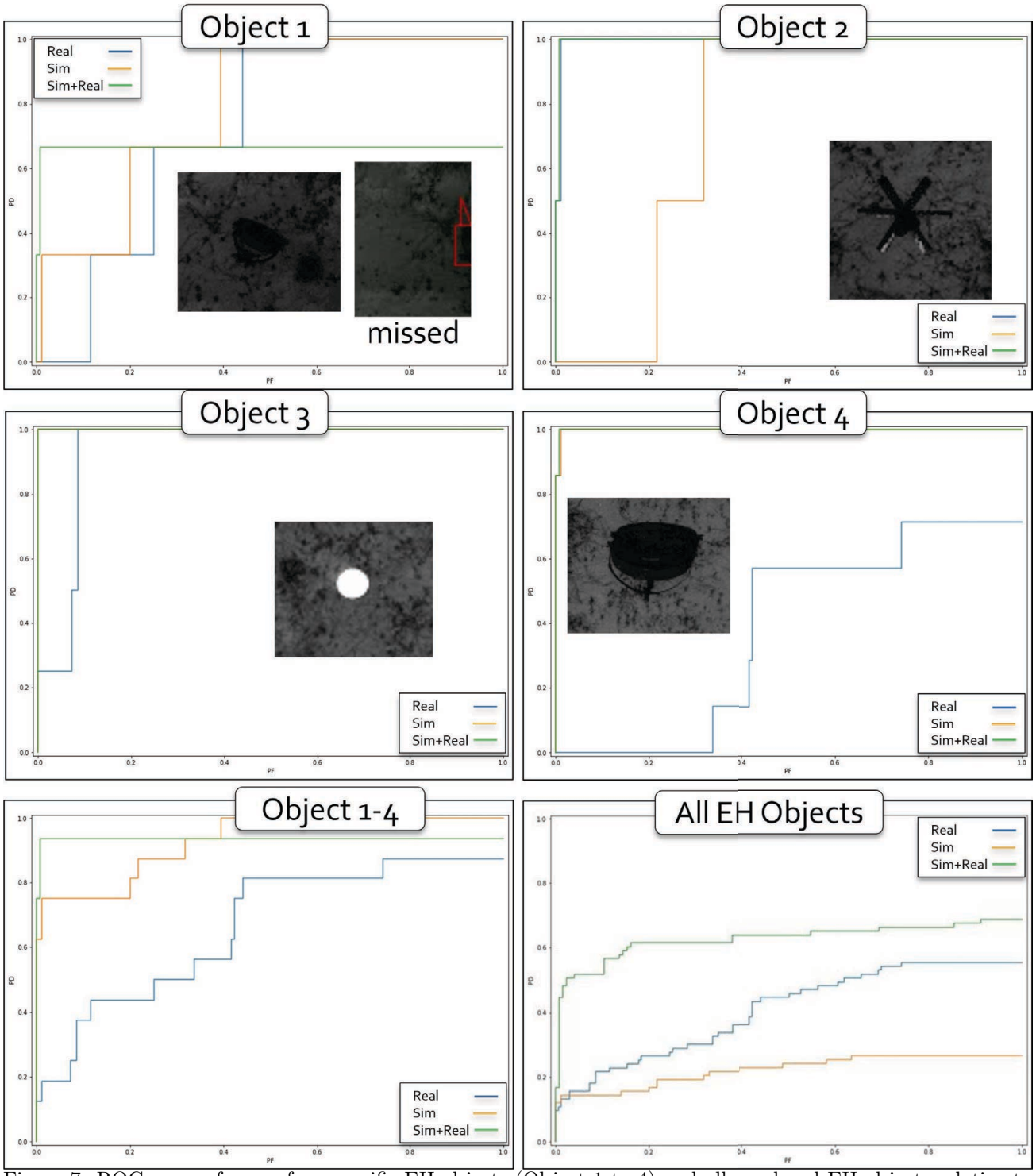


Figure 7: ROC curves for our four specific EH objects (Object 1 to 4) and all emplaced EH objects relative to our trained models; real data only trained U-Net, Sim only U-Net, and Real+Sim U-Net. Examples chips are shown in each ROC and an example Object 1 missed detection is shown. This detection was missed because it is at the edge of our image. Should it have been labeled and be included in scoring?

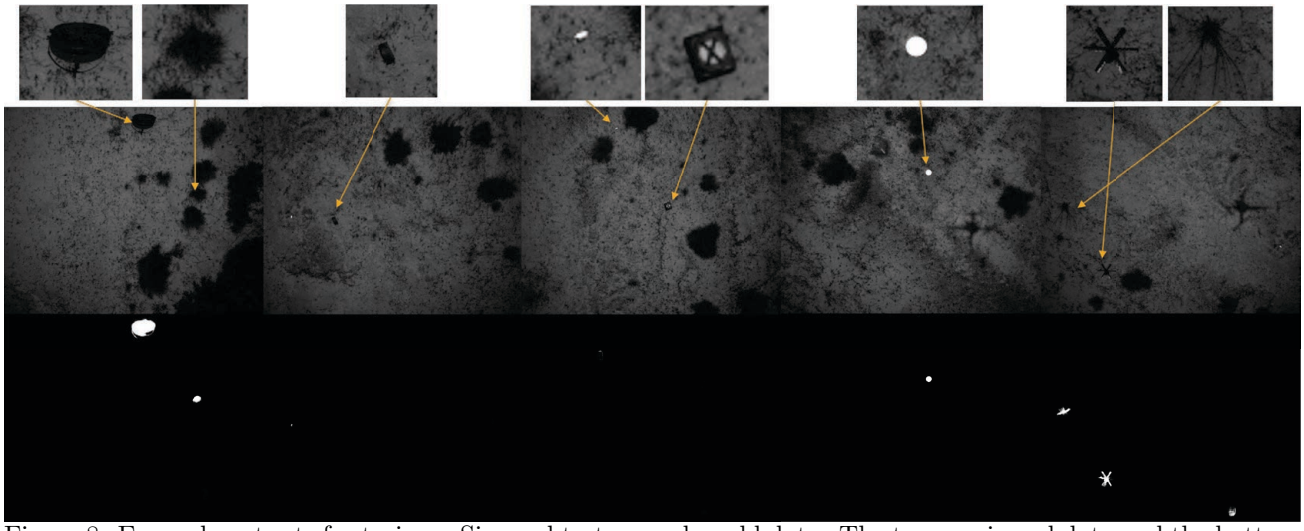


Figure 8: Example outputs for train on Sim and test on real world data. The top row is real data and the bottom row are Sim trained U-Net outputs. Example targets and FAs chips are shown.

Figure 7 also reports a ROC curve relative to scoring against all EH objects present in our data. It can be a challenge to interpret such a ROC as a good majority of these targets have too few of pixels on target and some are widely different that our four studied EH objects. We show this ROC because some of our additional EH objects, beyond Object 1 to Object 4, have similar shape and features. It is therefore interesting to observe the pre-screeners performance on this harder problem; generalizing to similar EHs. The reader can see that the system has a relatively low FAR, as the ROC curve climbs quickly, beyond Objects 1 to 4. This suggests that the combination of real and simulated data is able to enhance our U-Net and help learn more generalizable features and combination logic.

We do find it interesting, but not alarming, that in the case of all available EH objects the Sim performs the worst. Based on manual and visual inspection, we believe this can be explained as follows. The Sim data possesses just the four mentioned objects. At that, it sees them in a range of conditions, but not conditions (number of pixels on target, occlusion, etc.) consistent with the real data. It is not alarming that a real model will find a better way to make use of similar collect conditions and a specific environment. In future work we will explore if this gap can be closed with respect to a closer modeling of the environment, objects, and platform operating conditions, and/or a closer modeling of specific image statistics like intensities, contrast, etc.

Last, the reader can observe the example U-Net input (top) and output (bottom) images in Figure 8 to get a visual understanding of how well the network learns to predict our objects and their respective shapes. It should be noted that all real datasets had the human AABB labels were converted to rectangular pixel ground truth. However, the final predictions are not drawn by the U-Net as rectangles, which was the case for a U-Net trained only on real data. Instead, the Real+GEQ model has learned features that help us identify objects and ultimately more refined localization predictions. Figure 8 also shows the reader typical FAs. The Real+Sim model has relatively few FAs. However, the real model makes a greater number of mistakes on similar size, shape, and/or contrast regions, e.g., bushes, wild grass, tires, etc. An advantage of the simulation data is that it contains a greater number of close confusers.

7. CONCLUSIONS AND FUTURE WORK

In this article, we present preliminary evidence that a U-Net can be trained using simulated data from the Unreal Engine in the visual spectrum to recognize explosive hazards from a low-altitude aerial drone. Furthermore, we show that this performance is better than only using available limited volume and variety domain data. Ultimately, as one might anticipate, the combination of this data led to the best performance. However, it should be noted 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.

We are aware that this article is preliminary and it does not prove that these processes will work reliably and scale. The article is focused on a niche domain and small scale experimentation. We have also not provided any quantitative metrics beyond ROC curves to help understand what the U-Net learned. We have also not demonstrated to what degree TOON versus GEQ data, and what simulated object, environment, and platform variables are important.

There is a large body of future work ahead of us. Examples include focused experiments to assess: performance in light of non-deterministic U-Net training (what observations are real and not just a result of a single trained model); trends when/if greater amounts of real data are available (is there a trade off point where simulated data is of little to no use); to what degree does data sampled from the simulation reality spectrum impact performance. Furthermore, experiments should be performed to assess performance relative to controlled variation of underlying domain EH variables: object (e.g., size, pose, etc.), environment (e.g., time of day, occlusion, etc.), platform/sensor (altitude, focal length, etc.). This is something we are working on, but the initial framework had to be stood up. In Section 6, we also highlighted a number of questions that need more formal structured analysis. For example, did we learn simulator specific rendering features, real data image features, and/or combined or refined features. Furthermore, what *logic* was learned with respect to how to use these features and is it of benefit to present different visual abstractions of data during AI/ML training? The reader can also refer to our article on procedural generation of data in UE for a wealth of future work questions focused on the act of how to generate simulated data for a single dataset or closing the learning loop.[32]

We also did not make use of synthetically generated objects (aka templates) inserted into real data. In Alvey, et al.,[15] summarized in Section 2.2, we presented AMA. AMA had a much higher performance over real and fully simulated data relative to the YOLO object detection and localization algorithm. However, the data used in that article is our limited 720 2021 GEQ images. In future work, we need to assess if its more valuable to continue template insertion into real data for EHD or if full image simulated data is better. Clearly, AMA is limited with respect to what background imagery is available, whereas simulation can model different environments, environmental conditions, and platform contexts. While we anticipate that a combination will lead to the best performance, what simulated data should we produce?

In summary, we are encouraged by the preliminary results in this work and excited about future directions. At the time that this research was performed, the lead author was a senior in high school and the second and third authors were undergraduates in college. The point being, all of the tools and processes presented herein do not require a decade of education and a Ph.D.. Technology, specifically relatively free or low cost content, rendering tools, computational resources, and the ability to interface these technologies has the potential to unlock a rich simulated future. While simulation has been around for a long time, we are perhaps at a convergence point where it will exponentially flourish. Having possession of the truth and ability to alter things in controlled fashions is key towards our understanding of how things work and will aid our research into reliable and explainable AI/ML for domains like aerial-based EHD.

8. 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] 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] 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).

[5] 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).

[6] 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).

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

[8] 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).

[9] 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).

[10] 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).

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

[12] 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).

[13] 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).

[14] 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).

[15] 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).

[16] 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]).

[17] 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).

[18] 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).

[19] 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).

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

[21] 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).
[22] "Storytelling reimagined," (2021).
[23] "Real-time ray tracing," (2021).
[24] "UE Architecture." https://www.unrealengine.com/en-US/solutions/architecture. (Accessed: 1 March 2021).
[25] "Quixel." https://quixel.com/. (Accessed: 1 March 2021).
[26] "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).
[27] "Instant Terra Unreal Engine Plugin." https://www.unrealengine.com/marketplace/en-US/item/0c08f0ce7ac04724931565b2386012d0. (Accessed: 1 March 2021).
[28] "Houdinin Unreal Engine Plugin." https://www.sidefx.com/products/houdini-engine/plug-ins/unreal-plug-in/. (Accessed: 1 March 2021).
[29] "Unreal Marketplace." https://www.unrealengine.com/marketplace/en-US/store. (Accessed: 1 March 2021).
[30] "Cel Toon Outline Post Process Material." https://www.unrealengine.com/marketplace/en-US/item/65cd54d6adcc4b47a6c383d579beda4c. (Accessed: 1 March 2021).
[31] "Post Process Toolkit." https://www.unrealengine.com/marketplace/en-US/item/94cebc0ec72945899a494cf724f96293. (Accessed: 1 March 2021).
[32] Kerley, J., Fuller, A., Kovaleski, M., Popescu, P., Alvey, B., Anderson, D., Buck, A., Keller, J. M., Scott, G., Yang, C., Yasuda, K., and Ryan, H., "Procedurally generated simulated datasets for aerial explosive hazard detection," in [*SPIE*], (2022).

Chapter 7

Title: Characterization of Deep Learning-Based Aerial Explosive Hazard Detection using Simulated Data

Venue: IEEE Symposium Series on Computational Intelligence (SSCI)

Date Published: December 5, 2021

Characterization of Deep Learning-Based Aerial Explosive Hazard Detection using Simulated Data

Brendan J. Alvey
University of Missouri
Department of EECS
Columbia MO, USA
bja3md@umsystem.edu

Derek T. Anderson
University of Missouri
Department of EECS
Columbia MO, USA
andersondt@missouri.edu

Clare Yang
U.S. Army
DEVCOM C5ISR Center
Fort Belvoir, VA, USA

Andrew Buck
University of Missouri
Department of EECS
Columbia MO, USA
buckar@missouri.edu

James M. Keller
University of Missouri
Department of EECS
Columbia MO, USA
kellerj@missouri.edu

Ken E. Yasuda
U.S. Army
DEVCOM C5ISR Center
Fort Belvoir, VA, USA

Hollie A. Ryan
U.S. Army
DEVCOM C5ISR Center
Fort Belvoir, VA, USA

2021 IEEE Symposium Series on Computational Intelligence (SSCI) | 978-1-7281-9048-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/SSCI50451.2021.9659899

***Abstract*—Automatic object detection is one of the most common and fundamental tasks in computational intelligence (CI). Neural networks (NNs) are now often the tool of choice for this task. Unlike more traditional approaches that have interpretable parameters, explaining what a NN has learned and characterizing under what conditions the model does and does not perform well is a challenging, yet important task. The most straightforward approach to evaluate performance is to run test imagery through a model. However, the gaining popularity of self-supervised methods among big players such as Tesla and Google serve as evidence that labeled data is scarce in real-world settings. On the other hand, modern high-fidelity graphics simulation is now accessible and programmable, allowing for generation of large amounts of accurately labeled training and testing data for CI. Herein, we describe a framework to assess the performance of a NN model for automatic explosive hazard detection (EHD) from an unmanned aerial vehicle using simulation. The data was generated by the Unreal Engine with Microsoft's AirSim plugin. A workflow for generating simulated data and using it to assess and understand strengths and weaknesses in a learned EHD model is demonstrated.**

I. INTRODUCTION

If you were to go around a room of computational intelligence (CI) experts and ask them what they need to answer their most pressing questions at the moment, many of them would say, "more data!". Unfortunately, collecting real data sets from physical sensors with the variety and volume desired is often difficult or infeasible. With modern simulation, photorealistic imagery can now be rendered and accurately labeled in realtime. This is an exciting time as an abundance of high quality data is now within reach. Within the context of object detection, there are some common questions that arise. What are the limitations and expectations for a model once it is trained? What viewing and lighting conditions lead to peak detection performance? What conditions lead to poor performance? For interpretable methods such as decision trees and linear regression, you can look directly at the learned parameters and deconstruct exactly how the model will behave. However, neural networks (NNs) often outperform the more interpretable methods for the task of object detection and classification. These NNs are not black boxes, but their behavior is much more difficult to predict from weights alone. They can be explained in the sense that we can mechanically describe the different parts, how they function together, and how that ultimately results in performing a desired task. However, it is increasingly difficult to predict how a deep NN model will behave under a large variety of conditions. They are often composed of many connected layers with pooling steps and non-linear activation functions in between. Building intuition directly from the learned set of weights that represent high-dimensional nonlinear functions is not something that is feasible for large networks. Instead, by generating simulated data using the Unreal Engine, we evaluate an object detection model under a variety of conditions to assess performance and begin to answer some of these vital questions. This workflow can be expanded upon for a number of applications as discussed in the final section of this paper. A visualization of the workflow is shown in Figure 1.

Explaining artificial intelligence (AI) is, generally, one of the most difficult yet important tasks facing our field. Computer algorithms are increasingly either making decisions for us, or guiding our decisions, with no slowing down in sight. What advertisements should we be shown? Who should and shouldn't we digitally socialize with? What information is shown when we enter certain search terms? Is that a stop sign ahead that our car needs to stop for? The answers to these questions have very serious impacts on people's lives. People can trust computer algorithms if they can understand,

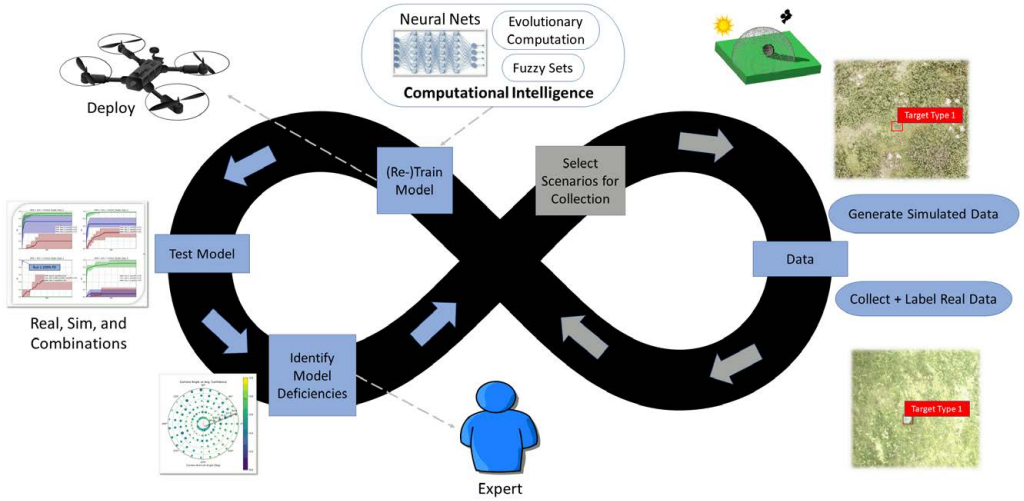


Fig. 1. Flowchart illustrating the main paper ideas relative to EHD. The objective is to support improvement and enhanced understanding of CI algorithms. Starting with an initial real, simulated, or hybrid CI model, performance is evaluated and characterized relative to a set of criteria and deficiencies (aka contexts) are identified (manually or by an algorithm). Next, this information is used to design new experiments and data can be generated using simulation, collected using a real platform, or a combination therein. Many workflows are empowered by this framework, *e.g.*, profiling a real-world model to predict failure to targeted or continuous open ended learning. Blue indicates pieces implemented and investigated and grey represents future work.

at least at some level, the logic the machine is following to make its decisions. Large black box data-driven models do not lend themselves naturally to this level of understanding and trust. "Explainable AI" (XAI) is an attempt to address and remediate issues like knowing why (or why not) an AI made a decision or predicting when a system will succeed and fail [1]. Algorithmic-based XAI methods like Local Interpretable Model-agnostic Explanations (LIME) [2] have been proposed to generate simple linear model explanations for complicated black box models. Like LIME, we also treat EHD as a black box. However, we use simulation to characterize, explain, and ultimately help improve a CI approach.

Unsupervised learning was developed to solve problems where labeled training data is non-existent. However, they tend to not be as accurate as supervised techniques, which take advantage of labeled data. A third style known as self-supervised learning attempts to indirectly optimize a solution by using existing, unlabeled data, as a ground truth. One simple example of self-supervision is an auto-encoder which attempts to learn a lower dimensional representation of data by reconstructing itself after going through a narrowing then expanding neural network (NN) [3]. Self-supervision has been shown to be successful at learning models that can predict dense disparity maps from a single image [4], [5]. These techniques generally work by taking an image, predicting the disparity map and camera pose for the image, then predicting the pose of a temporally adjacent image, and feeding back the error between the real adjacent image, and a reconstruction using the predicted disparity map and pose. While these methods have the ability to learn from the abundance of unlabeled data, the automatically generated supervisory signal may not guide the network to learn correctly. These models can also be more computationally expensive to train than supervised models. They often require additional processing to transform network outputs back into image space in order to compute loss then backpropogate, instead of being able to compute the loss directly. Self-supervised methods often struggle to learn well at first and require transfer learning from pre-trained models. The need for self-supervision results directly from the lack of abundant, accurately labeled training data, which is otherwise preferred. Within the context of depth or disparity prediction, accurate dense ground truth does not exist at a variety of ranges. Thankfully, using simulation can provide this for photorealistic imagery easily. Virtual cars can be driven over and over again with variations in scene and lighting conditions to quickly generate training data that is currently impossible to obtain in the real world. Obtaining accurately labeled EHD data may not be as difficult as obtaining accurate dense depth maps, but it is still very challenging and, thankfully, something that simulation can help us with.

Explosive hazards (EHs) have been detected with a number of sensor modalities, *e.g.*, ground penetrating radar (GPR) [6]–[8], sonar [9], electromagnetic induction (EMI) [10]–[12], infrared [13], RGB [14], and acoustics [15], [16]. EHs come in a variety of shapes and sizes. They can be buried underground, placed above ground, or even in the ocean. Identifying and removing EHs is an important but difficult task. Each sensor type comes with advantages and disadvantages. The number of EHs is staggering and their devastating impact on innocent civilians, including children, should be prevented [17]. The task is better suited for autonomous, unmanned systems, allowing for a large distance between an operator and potentially dangerous EH. Unmanned aerial vehicles (UAVs), *e.g.*, *drone* equipped with cameras and positional sensors, are especially well suited for the task of surface EHD due to their far standoff distance relative to target. Their maneuverability is also a

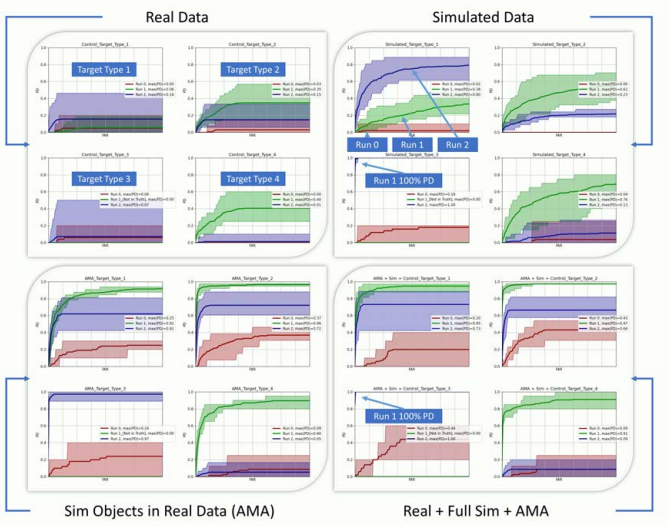


Fig. 2. Receiver operator curve (ROC) characterizing detection performance. Each ROC is scored on a single target type. There are four target types in total. Each curve is scored over a run of data. There were three runs of real testing data used. Top Left: Model trained only on real aerial explosive hazard data. Top Right: Model trained on simulated data only. Bottom Left: Model trained on AMA generated images only. Bottom Right: Model trained on real data, simulated, and AMA generated images. For each data set, 10 models were trained with the same learning parameters. The area between the minimum and maximum PD for each set of models is shaded. The average PD across the models is plotted in bold.

useful feature for controlled collections of imagery.

Images of EHs have previously been generated by our group using the Unreal Engine. Target and shadow template images were extracted by placing targets on a solid background. These template images were then put into our Altitude Modulated Augmentation (AMA) pipeline where simulated targets were inserted into real images of environments containing EHs. By using purely simulated data, and AMA generated data, detection performance was improved for a real EH data set [18]. Figure 3 shows the performance gains that can be achieved by images generated using simulation, especially with a limited training set. Data augmentation techniques have been shown to significantly improved detection results [19], [20]. They attempt to create novel imagery from existing data sets. Our method creates novel imagery using simulation which then could be used in combination with exiting augmentation techniques to improve overall detection results under a wide range of conditions. The receiver operator curves shown in figure 2 summarize our results from a previous paper involving EH detection. We compared models which were trained on different data set; either real, simulated, AMA, or mixed, and found that simulation could be used to improve EH detection on a real data set given a limited real training set. For each data set, 10 models were initialized with random weights and trained with same learning parameters. The variance between these models is shown by shading the are between the minimum and maximum percentage of targets detected (PD)s. The average PD across a set of models is plotted in bold.

Fig. 3. Example images from the "Mountain Grassland Environment 2x2 km" from the Epic Marketplace used for our experiments. With this environment, photorealistic rending of a large scene with a variety of trees, foliage, grass, gravel, and rocks, is easily done.

II. Generating Simulated Explosive Hazard Data

The gap between simulated and real data is closing. Modern rendering tools such as realtime ray-tracing, high-resolution textures, and post-processing effects, are enabling photorealistic imagery to be generated in real-time entirely from simulation [21]. One of the most popular and widely used software for rendering and creating interactive virtual experiences is Epic's Unreal Engine [22]. Combined with Microsoft's AirSim plugin [23] for Unreal Engine, we program a remote agent to fly around in virtual worlds and collect simulated EH data.

Collecting an abundance of controlled, labeled data is an important but challenging task in the real-world. Hand labeling data is tedious, time-consuming, and sometimes requires subject matter experts. Although there are software solutions which attempt to aid and speed up annotation such as Microsoft's VoTT [24], the process is still relatively slow and potentially erroneous due to the human factor. Unfortunately this problem scales linearly as well — if you need 10x more data labeled than before, you will need 10x more human labor to label it. Besides computing power and speed, obtaining accurate ground truth for data is one of the biggest and most common bottlenecks in data science and machine learning. While fully capable UAVs are becoming much more easy and accessible, accurately labeling the data they collect is not an easy or automated task. Simulation is useful for this problem. Having instant access to pixel perfect labels for segmentation, accurate dense depth values, and many other otherwise difficult-to-obtain ground truths, is invaluable.

There is an art to constructing virtual environments in Unreal Engine. Instead of creating convincing landscapes ourselves, we purchased content from the online marketplace. Instances of EHs were scattered across a grassland mountain environment, which was purchased from the Epic Marketplace [25]. This environment contains a large, pre-built, realistic landscape, shown in Figure 3 below. Thankfully, quality content like this is available for purchase which allows us to spend a minimal amount of time designing and modeling scenes.

Likewise, modeling EHs is not something we have much expertise in so we opted to purchase existing 3D models from TurboSquid [26] and Free3d [27]. Four EH models were

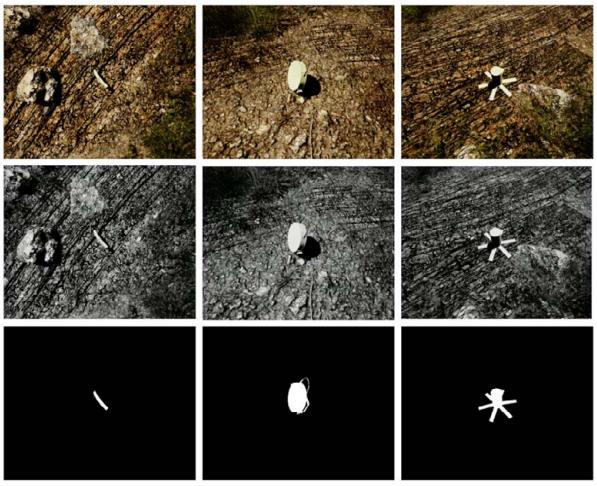


Fig. 4. EH images rendered via AirSim using the "Mountain Grassland Environment 2x2 km" map. Top: RGB renderings. Middle: grayscale imagery. Bottom: target mask, extracted using AirSims's semantic segmentation.

selected and placed in various locations in the environment. The target models vary significantly in size and shape. Target type 1 is a flat, circular object. Target type 2 is a larger version of the target type 1. Target type 3 has a vertical cylindrical body. It is supported by 6 flat, rectangular, metal feet. The final target, target type 4, is a small, rectangular, slightly concave object. These models were previously used to create 2D target templates that were inserted into real images to generate training data to improve EHD models [18]. The models were imported into grassy mountain environment, assigned a physics asset, and configured to have physics enabled. This allows objects to be placed slightly above the ground so that when the simulation is started, they fall realistically into place, taking into account collisions with the floor and surrounding objects. For each object type, we placed 10 instances at various locations in the map. The locations were chosen to provide a variety of background textures, shapes, and objects. Example renderings of these target models in the simulated environment at various standoff distances as well as sun and camera angles are shown in Figures 4 and 5. AirSim was used to control a virtual camera in the Unreal Engine and capture images. AirSim enables the manipulation of objects as well, which allows us to move the sun to any angle. Images were generated at a resolution of 2560×2048 and down-sampled 4x to 640×512 This was done to decrease aliasing and improve overall image quality. A post-processing object was also used to disable automatic exposure camera settings. Exposure settings are also fixed in AirSim in order to generate images which accurately capture the change in effective luminance caused by different sun elevation angles.

III. Characterizing Detection Performance

DNNs have become the tool of choice for state-of-the-art object detection, and there is a huge variety of architectures. DNNs usually require learning a large number of parameters (weights) as well as hyper-parameter tuning (learning rate, momentum, batch size, etc.) to obtain ideal performance. This is what gives these networks both flexibility in terms of possible behaviors, and difficulty in terms of interpretability. With so many knobs to turn, it quickly becomes infeasible for a human to look over a table of weight values and determine how exactly a network will behave. Instead, we must turn to other methods to characterize the ability of a given network to perform a particular task. For the task of aerial object detection, we would like to know under what conditions does a trained model do best. Under what conditions does it fail to meet expectations or behave unexpectedly? Without the ability to understand and predict the network behavior directly, we instead treat the network as a black box. Images go in and detection declarations come out.

Fig. 5. EH images rendered directly in the Unreal Engine. Left: model placed on a gravel road with surrounding grass and trees. Right: model placed in a grassy environment up close.

It is impossible to create an image object detector that works under every possible set of conditions. If the camera is too far away or the lighting is poor, an object may simply be unable to be detected, and there is nothing we can do about it. However, typically there are a set of operating conditions that define the reasonable expectations for a given model. For example, a truck detector may be expected to be able to find red and green consumer pickup trucks on sunny days, when the sun is at least 30° elevation. Once a model is trained, there are always questions as to how well it will preform. In this paper we choose a somewhat naive approach, which is to simply to create simulated data sampled within the bounds of the defined operational conditions, then to use that data to evaluate the learned models.

Aerial images of the exact same object will look very different depending on the camera and lighting conditions. For example, a thin rectangular box laying flat will look like a typical rectangle when viewed from nadir, but from a steep elevation angle it may be very difficult to see and only appear as a thin line. Standoff distance, sun azimuth, sun elevation, camera azimuth, and camera elevation are all controlled and logged exactly using AirSim with the Unreal Engine when generating imagery. To systematically generate simulated imagery for evaluating model performance, the space surrounding emplaced targets is sampled along a hemisphere grid. The radius of the hemisphere is changed to alter the average number of pixels on target and the relative level of detail. Figure 6 shows a visualization of this hemisphere sampling and some example of images collected using it. A plain white, grey, or black plane can be chosen as the background to

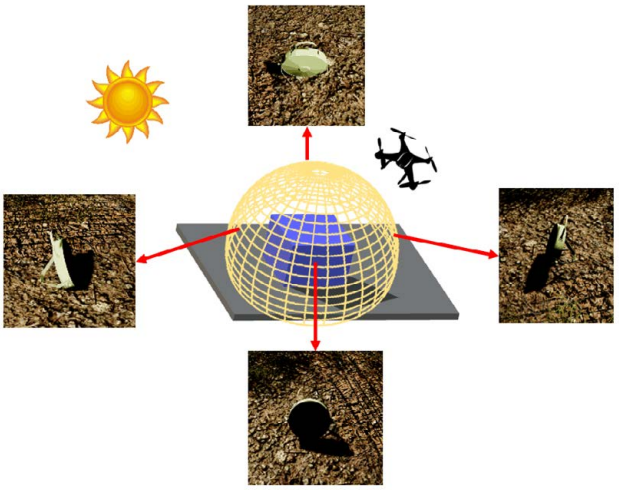


Fig. 6. Hemisphere Image Sampling: Images are captured at various camera azimuth and elevation angles at a fixed standoff radius from the center of the target. Examples are of an EH model emplaced into the Mountain Grassland environment, at a camera elevation angle of 45° off-nadir.

get a baseline for performance. The scene can be adapted to the analysis desired. Objects can be placed in front of target objects to characterize occlusion tolerance. Naturally if an object is occluded at certain camera angles, the detection performance will drop off, but can we find out by how much and when? Different backgrounds and target emplacement orientations will also likely effect detection performance, again how significant is this? We can now generate imagery quickly which can help to answer these questions.

Alongside the simulated images, detailed ground truth JSON files are also outputted. They contain bounding box declarations, and associated metadata such as camera and sun angles for each image. Once a model is trained, the performance can be characterized by running the simulated images through the detector, scoring the results, and visualizing them. Unfortunately, with each variable to consider, we add another dimension and further complicate visualization. If we want to analyze the the detection confidence as it varies with the sun and camera angles, given a fixed standoff radius, then we already have 5 variables to consider (confidence, sun azimuth, sun elevation, camera azimuth, and camera elevation).

To visualize all of these at once, we first start by holding the sun angles constant and letting only the camera angle change. We have provided a graph key to try to help interpret the detection confidence figures shown in this paper in Figure 7. We plot the confidence as a function of camera sun and azimuth angle. This allows us to, at a glance, determine which viewing angles are optimal and which are challenging. Those angles that have relatively lower confidences should be investigated and possibly selected for sampling to improve future detection models. Building on these plots, we visualize performance across sun angles. Following a similar geometric encoding as the camera positions, the sun position is also encoded. Each camera angle plot as on the left of Figure 7 becomes a sub figure in a sun angle plot such as on the right of Figure 7. The center of each sub-figure on the right depends on sun angle.

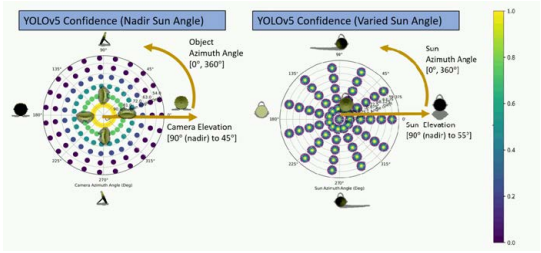


Fig. 7. Graph Key: Left figure shows an example confidence scoring as a function of camera angle. Right figure shows an example confidence scoring as a function of sun angle. Target template images are put on top of their corresponding collection parameters. The confidence values vary from 0 to 1 in this plot and were generated just for illustrative purpose. If these confidence values were generated from a model, they would represent a model biased towards nadir detection and invariant to sun position.

Starting from the right side of the camera angle figure, the collection parameters are 0 camera azimuth angle and a somewhat steep 45° camera elevation angle. Moving around the graph clockwise, directly above we see the target from right side. This results in fewer pixels on target and a significantly different profile. The target templates near the center of the left figure are collected from a nadir camera angle at the 4 main cardinal azimuth angles. Likewise for the sun angle comparison figure on the right side, templates are placed to visualize the different sun angles and their effect on target presentation. Furthest to the right, the target template was collected when the sun was directly behind the target and at a relatively low elevation angle. Moving around the chart clockwise, directly above we show the associated target template when the sun was directly to the right of the target, causing a shadow to the left. The confidences in the graph key and the two toy example figures are completely synthetic and are not intending to represent any real model results.

IV. Experimental Results

Ideally, an object detector would be able to recognize its learned object in every possible context it could ever be seen in. Realistically, an object detector will only be optimized to work well within the contexts that are provided in the training data. It is important to assess the limitations and expectations for a model to know where we can expect it to preform reasonably well and where it will break. Using the characterization visualizations described in the previous section, we qualitatively analyze the performance of this model across camera and sun angles. We can also compare the detection performance across different environments.

An EHD model was previously trained on a combination of real, simulated, and AMA generated images [18]. It is a custom trained YOLOv5 [28] model. This model was trained on a very limited data set and was crafted to preform well within a relatively narrow range of operational parameters. The training set consisted of only nadir images collected from

at a fixed altitude during the time of day when the sun was near directly overhead.

The first scene that was setup for hemisphere sampling is intended to function similarly to an impulse response for the detector, although this was not the eventual outcome. We start with a completely empty space. An infinite, white, horizontal plane is added as a platform. A single directional light is set to function as an atmospheric sun. A post-processing volume with infinite extent is used to disable the auto-exposure feature. This was done in order to preserve changes in brightness at different sun elevation angles. To render accurate shadows, ray-tracing and cascaded shadowing was also enabled.

Images were collected by regularly sampling a hemisphere centered at the pivot point for the object's mesh. The camera elevation angles only varied from $45°$ to $90°$, the camera azimuth angles varied the full $360°$. The sun elevation angles were only varied from $55°$ to $90°$ and the sun azimuth angles were also varied the full $360°$. The radius of the hemisphere was varied from 20m to 100m. The number of images needed to fully sample the hemisphere increases quickly with finer sampling. For these experiments we used 24 Camera Azimuth Angles*6 Camera Elevation Angles*12 Sun Azimuth Angles*6 Sun Elevation Angles=$10,368$ images per hemisphere. If you want to explore another variable like target color or orientation then you have to multiply the number of images per hemisphere by however many variations of the new variable there are. If you want to investigate 10 different colors, that requires over $100,000$ images at this sampling resolution. Future work may address this issue with some form of bounded stochastic or intelligent sampling.

Figure 8 summarizes the characterization results for a pre-trained Yolov5 EHD model [18]. This model was trained on a variety of EHs ranging in terms of size, shape, and composition. The detection model does not discriminate between different types of targets even though the results presented only represent a characterization of detection performance for a single target type. The figure shows how the model begins to fail at detecting the target when the number of pixels on target gets to be too large. At a 20m simulated altitude, the target looks too different from training images at steep elevation angles, causing a drop in detection performance. Surprisingly, the model is able to detect the target reasonably well even at a 100m altitude. With so few pixels on target, we believe this is primarily due to the model having been trained on smaller target types. Those smaller target types have likely influenced the model to cue off of small contrast and edge features. Looking across altitudes, the back of the target tends to have lower detection performance and a larger variance. The bottom chart demonstrates how detection performance varies as a function of the background environment. The highest confidence detections can be found on a benign background at 30m. When the target was placed on a grass background, detection confidence dropped from the back of the target at lower camera elevation angles. Similar trends were found when the target is placed on gravel, except the poor detection was exaggerated across more azimuth camera angles.

V. Conclusion and Future Work

The visualizations in the previous section demonstrate how simulated imagery can be used to characterize the detection performance of an aerial CI-based EHD model. It is important to put the results shown here in context. The model evaluated was only trained on a limited data set. It was trained on several different target types and does not discriminate between them, even though we only evaluated a single target type. This model we selected to evaluate was simply *"A"* model pre-trained among many and is not intended to represent the specific behavior of any particular platform or system. In addition, the results shown only represent detections on simulated imagery. We expect that insights gained from these characterizations will translate into the real-world but verification is necessary. The next step is to take trends that are identified using simulated data and investigate if the same trends are present when testing on real aerial EH imagery. In addition, we only tested targets placed at the center of images. Another question that could be answered with simulation is how invariant is the model to translation or rotational changes.

An automatic training workflow could be constructed using the characterizations described in this paper. Given an initially trained model and a set of operation parameters, first the performance can be evaluated using simulated data. Deficiencies can be identified–*e.g.*, by thresholding confidence values or doing something more sophisticated such as clustering or rank ordering–and then data can be automatically generated to specifically match the deficiencies identified. For example, long shadows at low sun elevation angles could be identified as causing poor detection. This could be from one of two things. Either the problem is inherently more difficult (requires more computational complexity) or the model is lacking training data. If it is the former, then new techniques must be developed or learned which is outside the scope of this paper. If it is the later then we are in luck and can generate simulated images specifically to address the shortcomings of the current model. These images can then be fed back into training, either starting from scratch or transfer learning. This forms a feedback loop whereby real data could be continuously collected and evaluated.

A diagram of the workflow is found in Figure 1 at the beginning of this paper. The sections colored in blue represent parts of the loop that we have investigated and implemented. The greyed sections represent future work, which will now be briefly described. Now that a platform and method for evaluating models and identifying deficiencies has been established, we would like to use this. Data can be intelligently generated to specifically address CI model deficiencies. For example, if a model is identified as being unable to detect a target well with low sun elevation angles (long shadows) then we can randomly generate examples to help improve the model. Depending on the machine learning system used, the new data may be used to train an entirely new model or transfer learn from existing weights. Our method does not take into account false alarms and the quality of detections. Future work is needed to

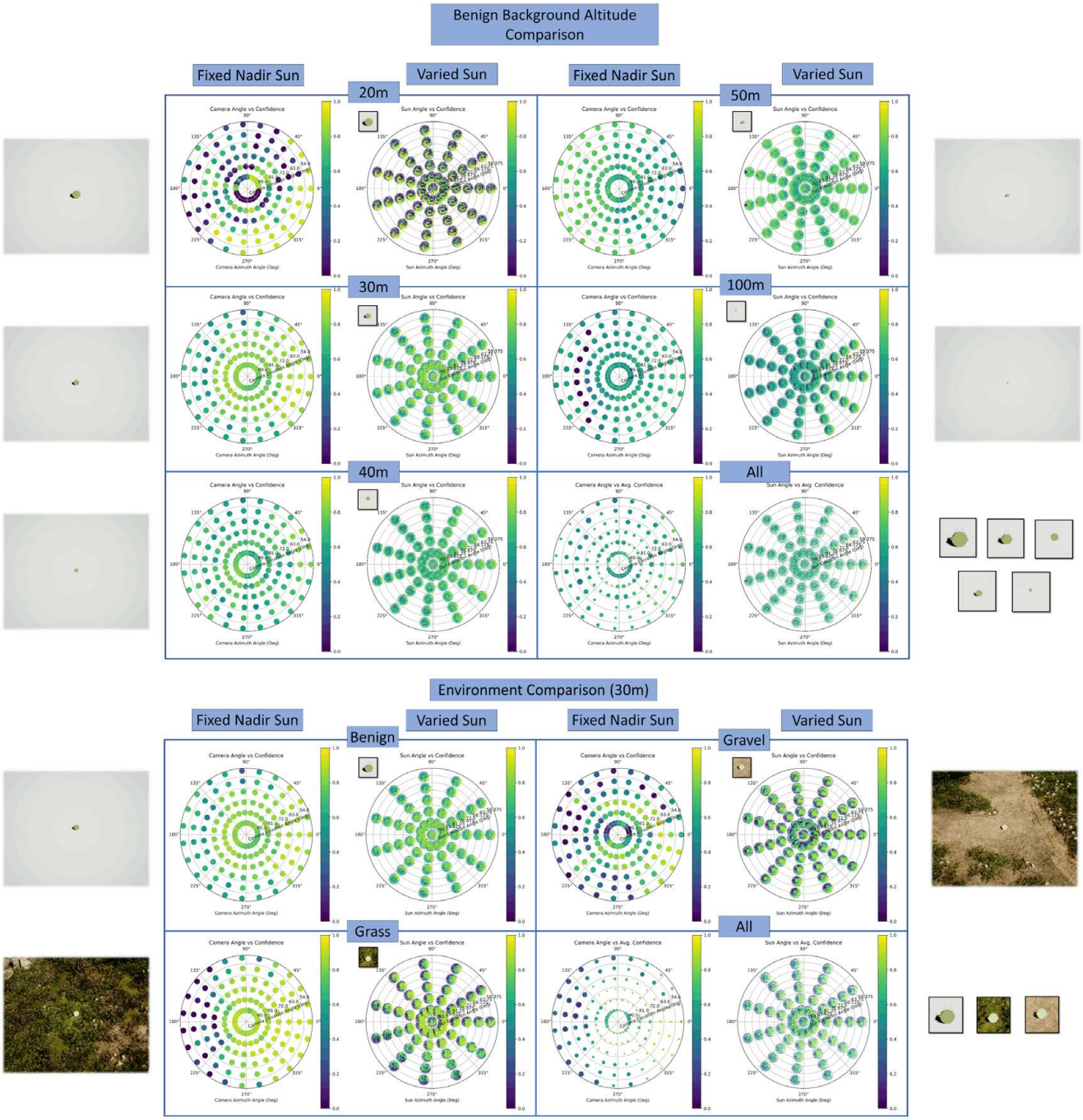


Fig. 8. Visualizations of confidence as it varies with camera and sun angles. Top chart shows results across several different simulated altitudes. The "All" graph summarizes the results; the color is determined by the average confidence across the different altitudes and the size of the circle corresponds to the variance across the runs. A larger circle implies a higher variance across the variable that was being tested (altitude or environment). *e.g.* A higher variance across the tested altitudes is plotted with a larger circle. The bottom chart shows results across several different environments. The surrounding pictures are sample renderings from each hemisphere data set.

include these factors to get a better representation of model quality. As the field of computation intelligence continues to use increasingly complex tools to solve its problems, the need to systematically characterize the performance of these machines also increases. Using simulation allows us to scale both training and testing data in terms of volume, accuracy, and variety to levels that would be infeasible with manual collections.

REFERENCES

[1] M. Turek, "Explainable artificial intelligence (xai)." [Online]. Available: https://www.darpa.mil/program/explainable-artificial-intelligence

[2] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," 2016.

[3] A. Gogna and A. Majumdar, "Semi supervised autoencoder," in *Neural Information Processing*, A. Hirose, S. Ozawa, K. Doya, K. Ikeda, M. Lee, and D. Liu, Eds. Cham: Springer International Publishing, 2016, pp. 82–89.

[4] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras," *CoRR*, vol. abs/1904.04998, 2019. [Online]. Available: http://arxiv.org/abs/1904.04998

[5] C. Godard, O. M. Aodha, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," *CoRR*, vol. abs/1806.01260, 2018. [Online]. Available: http://arxiv.org/abs/1806.01260

[6] R. E. Smith, D. T. Anderson, J. E. Ball, A. Zare, and B. Alvey, "Aggregation of Choquet integrals in GPR and EMI for handheld platform-based explosive hazard detection," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXII*, S. S. Bishop and J. C. Isaacs, Eds., vol. 10182, International Society for Optics and Photonics. SPIE, 2017, pp. 300 – 314. [Online]. Available: https://doi.org/10.1117/12.2263005

[7] L. E. Besaw, "Detecting buried explosive hazards with handheld GPR and deep learning," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXI*, S. S. Bishop and J. C. Isaacs, Eds., vol. 9823, International Society for Optics and Photonics. SPIE, 2016, pp. 187 – 197. [Online]. Available: https://doi.org/10.1117/12.2223797

[8] S. Harris, K. C. Ho, and A. Zare, "On the use of log-gabor features for subsurface object detection using ground penetrating radar," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXI*, S. S. Bishop and J. C. Isaacs, Eds., vol. 9823, International Society for Optics and Photonics. SPIE, 2016, pp. 161 – 169. [Online]. Available: https://doi.org/10.1117/12.2223324

[9] G. J. Dobeck, "Fusing sonar images for mine detection and classification," in *Detection and Remediation Technologies for Mines and Minelike Targets IV*, A. C. Dubey, J. F. Harvey, J. T. Broach, and R. E. Dugan, Eds., vol. 3710, International Society for Optics and Photonics. SPIE, 1999, pp. 602 – 614. [Online]. Available: https://doi.org/10.1117/12.357082

[10] C. H. McCurley, J. Bocinsky, and A. Zare, "Comparison of hand-held WEMI target detection algorithms," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIV*, S. S. Bishop and J. C. Isaacs, Eds., vol. 11012, International Society for Optics and Photonics. SPIE, 2019, pp. 221 – 240. [Online]. Available: https://doi.org/10.1117/12.2519454

[11] B. Alvey, A. Zare, M. Cook, and D. K. C. Ho, "Adaptive coherence estimator (ACE) for explosive hazard detection using wideband electromagnetic induction (WEMI)," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXI*, S. S. Bishop and J. C. Isaacs, Eds., vol. 9823, International Society for Optics and Photonics. SPIE, 2016, pp. 58 – 64. [Online]. Available: https://doi.org/10.1117/12.2223347

[12] J. Dula, A. Zare, D. Ho, and P. Gader, "Landmine classification using possibilistic K-nearest neighbors with wideband electromagnetic induction data," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XVIII*, J. T. Broach and J. C. Isaacs, Eds., vol. 8709, International Society for Optics and Photonics. SPIE, 2013, pp. 395 – 406. [Online]. Available: https://doi.org/10.1117/12.2016490

[13] D. T. Anderson, J. Farrell, K. Stone, J. M. Keller, and C. Spain, "Fusion of anomaly algorithm decision maps and spectrum features for detecting buried explosive hazards in forward looking infrared imagery," in *2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2011, pp. 1–8.

[14] D. T. Anderson, O. Sjahputera, K. Stone, and J. M. Keller, "Causal cueing system for above ground anomaly detection of explosive hazards using support vector machine localized by k-nearest neighbor," in *2012 IEEE Symposium on Computational Intelligence for Security and Defence Applications*, 2012, pp. 1–8.

[15] N. Xiang and J. Sabatier, "Laser doppler vibrometer-based acoustic landmine detection using the fast m-sequence transform," *IEEE Geoscience and Remote Sensing Letters*, vol. 1, no. 4, pp. 292–294, 2004.

[16] J. Sabatier and N. Xiang, "An investigation of acoustic-to-seismic coupling to detect buried antitank landmines," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 39, pp. 1146 – 1154, 07 2001.

[17] "Protecting children from explosive weapons," May 2019. [Online]. Available: https://www.unicef.org/protection/protecting-children-from-explosive-weapons

[18] B. Alvey, D. T. Anderson, J. M. Keller, A. Buck, G. Scott, D. Ho, C. Yang, and B. Libbey, "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*, S. S. Bishop and J. C. Isaacs, Eds., vol. 11750, International Society for Optics and Photonics. SPIE, 2021, pp. 76 – 90. [Online]. Available: https://doi.org/10.1117/12.2586342

[19] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[20] K. Zhang, Z. Cao, and J. Wu, "Circular shift: An effective data augmentation method for convolutional neural network on image classification," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 1676–1680.

[21] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha, and B. Guenter, "Photorealistic image synthesis for object instance detection," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 66–70.

[22] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com

[23] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *CoRR*, vol. abs/1705.05065, 2017. [Online]. Available: http://arxiv.org/abs/1705.05065

[24] Microsoft, "microsoft/vott: Visual object tagging tool: An electron app for building end to end object detection models from images and videos." [Online]. Available: https://github.com/microsoft/VoTT

[25] "Mountain grassland environment 2x2 km in environments - ue marketplace." [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/mountain-grassland-environment-2x2-km

[26] d. International and d. International, "3d m18a1 claymore anti personnel - turbosquid 1642128," Oct 2020. [Online]. Available: https://www.turbosquid.com/3d-models/3d-m18a1-claymore-anti-personnel-1642128

[27] "Mon 100 russian bomb 3d model." [Online]. Available: https://free3d.com/3d-model/mon-100-russian-bomb-3432.html

[28] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, A. Hogan, lorenzomammana, yxNONG, AlexWang1900, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, F. Ingham, Frederik, Guilhen, Hatovix, J. Poznanski, J. Fang, L. Yu, changyu98, M. Wang, N. Gupta, O. Akhtar, PetrDvoracek, and P. Rai, "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4154370

Chapter 8

Title: Simulated Photorealistic Deep Learning Framework and Workflows to Accelerate Computer Vision and Unmanned Aerial Vehicle Research

Venue: IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)

Date Published: October 11, 2021

2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW) | 978-1-6654-0191-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCVW54120.2021.00435

Simulated Photorealistic Deep Learning Framework and Workflows to Accelerate Computer Vision and Unmanned Aerial Vehicle Research

Brendan Alvey, Derek T. Anderson, Andrew Buck, Matthew Deardorff, Grant Scott, James M. Keller
University of Missouri
Columbia MO 65211, USA
(bja3md@mail, andersondt@, buckar@, msdrm8@mail, scottgs@, kellerj@)missouri.edu

Abstract

Deep learning (DL) is producing state-of-the-art results in a number of unmanned aerial vehicle (UAV) tasks from low level signal processing to object detection, 3D mapping, tracking, fusion, autonomy, control, and beyond. However, barriers exist. For example, most DL algorithms require big data, but supervised ground truth is a bottleneck, fueling topics like self-supervised learning. While it is well-known that hardware and data augmentation plays a significant role in performance, it is not well understood which data augmentations or what real data need be collected. Furthermore, existing datasets do not have sufficient ground truth nor variety to support adequate controlled experimental research into understanding and mitigating limitations in DL algorithms, models, data, and biases. In this article, we address the combination of photorealistic simulation, open source libraries, and high quality content (models, materials, and environments) to develop workflows to mitigate the above challenges and accelerate DL-enabled computer vision research. Herein, examples are provided relative to data collection, detection, passive ranging, and human-robot teaming. Online video tutorials are also provided at https://github.com/MizzouINDFUL/UEUAVSim.

1. Introduction

A recent report by the Grand View Research [1] estimated that the global market size of AI was $39.9 billion in 2019, with a projected 42.2% compound annual growth rate until 2027. Investors range from government (e.g., billions committed by the US [2]) to commercial (e.g., one billion from Tesla and SpaceX [3]). For example, Tesla is heavily investing in areas like large scale machine learning (ML) based computer vision (CV) from cameras, which has required them to perform large scale human assisted labeling of petabytes of data [4]. This data dependency is not new. Numerous companies have emerged and raised tens of billions to label data for ML [5]. The point is, data-driven AI/ML is on the rise for a variety of applications from cybersecurity to social media, remote sensing, biomedical, healthcare, smart cars, and beyond. At that, data is the *new oil* and new ideas are needed to generate and label data.

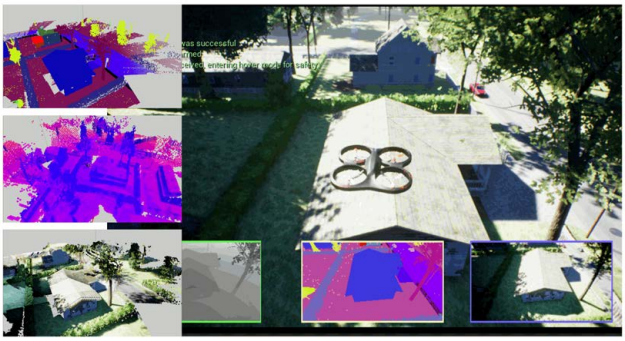


Figure 1. Example of a simulated aerial dataset for a rural neighborhood in AirSim and the Unreal Engine, automatically generated ground truth, voxel data, and 3D point cloud.

While recent AI/ML progress is impressive, a number of deep limitations have become clear. Examples include: dependency on large typically annotated for supervised learning volume/variety data; cost and time bottleneck of collecting (annotated) data; and understanding what data to augment or collect; and how to conduct controlled experiments in this complex landscape to study, profile, and understand algorithm, model, data limitations and biases. In this article, we explore photorealistic simulation to address these challenges. *Our contribution is a workflow for collecting controlled photorealistic simulated data with associated metadata via simple to use open source tools to assist deep learning (DL) CV research for unmanned aerial vehicles (UAVs).* Multiple examples are provided to help the reader apply and generalize these ideas to new contexts and code with video tutorials are provided at https://github.com/MizzouINDFUL/UEUAVSim.

978-1-6654-0191-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCVW54120.2021.00435

The use of simulation for data generation, algorithm training and testing, design-space exploration, sensor verification and validation, virtual and augmented reality, etc., is not new. What is new is a convergence in the maturity, realism, and availability of relatively simple to use tools and web-based tutorials that allow controlled, wide breadth simulation-enabled computer vision and DL research for individuals who are not computer graphics nor gaming experts. This is a potential game changer. Examples are boundless, e.g., accelerating research by enabling greater exploration, generating larger heterogeneous datasets, unit testing, improved ground truth, reducing time and cost barriers of collecting poorly labeled uncontrolled real-world data, and closed-loop simulated for systematic analysis and life long AI/ML learning. Just as hardware and data changed the face of DL, flexible open source photorealsitic simulation has the potential to be the next leap.

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. While Sim4CV is the closest work to our current article, the content used and imagery produced is not photorealistic, no detailed workflows are outlined, no supplemental online training is available, and results are simulator-focused vs real world and simulation cross-validated.

2. Modeling and Rendering : Unreal Engine

In this section, we highlight the Unreal Engine (UE) [14] for generation of photorealistic environments and pristine computer vision and UAV (meta)data production. Historically, UE was created for video games, but it is now used for film [15], computer graphics [16], architecture [17], and beyond. Figures 2 and 3 show existing free and purchasable online content, Figure 4 shows free high quality real-world scanned data, and Figure 5 shows an online store of purchasable content. 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.

The reader can refer to [21] for a video-based introduction to producing photrealistic scenes in UE via Quixel Megascans. Users can use AirSim (Section 3) to generate data or the Movie Render Queue (MRQ) in UE [22] (Figure 6). We recommend the MRQ over AirSim–to gain full access to lumen, ray tracing, anti-aliasing, and more photorealistic image generation options–and that users manually script a UAV flight using a Cine Camera Actor [23] (with parameters like spatial resolution, FOV, focal length, and more) as keypoints or tracks in the Sequence Editor [24].

Figure 2. Example of urban and rural scenes–prices range from free to a few hundred dollars–on the UE Marketplace [18].

Figure 3. Example of large outdoor area from the UE Marketplace [18] and seasonal variation for train/test data variation.

Figure 4. Example of photorealistic scenes by Quixel [19] (real-world scanned models and textures) content (over 15,000 available), made free to UE users.

Figure 5. Example Turbosquid [20] FBX content (500,000+ models) for rapid creation of dynamic environments.

We recommend this path to obtain the best imagery, and it is arguably the quickest way to get up and running for a controlled non-autonomous UAV dataset. For readers that need a higher degree of scene dynamics, the UE Blueprint Editor can be used to script based on a clock, spatial position of objects, collisions, or other triggerable events. If a reader requires depth or per-pixel object (or instance) IDs, the MRQ can be used. However, recording metadata like UAV state is not trivial; it requires a custom Blueprint.

3. Autonomy and Control : AirSim

Microsoft's AirSim [25], see Figure 1, is an open-source robotics simulation platform, for UE4. A number of extensions have been proposed, including simulating limited infrared (rather than standard RGB imagery) [26] and CinemAirSim [27], a camera-realistic robotics simulator for cinematographic purposes. The main advantage over an existing robotics simulation environment like Gazebo [28] is the maturity of UE with regard to photorealistic content creation (i.e., ray tracing, 3D modeling, physics, scripting, etc.). While AirSim has been used to date for a few tasks like deep reinforcement learning [29], drone racing [30], and autonomous cars [31], research has focused on setting up and validating the simulation environment, i.e., sensors, physics simulation of a car and drone, etc. AirSim is currently supported in C++ and Python.

While at first AirSim appears to be the winner for all DL and UAV research, there are drawbacks. To start, the cost of entry is arguably higher, backed by significantly less documentation and tutorials. Second, AirSim only has access to a subset of UE functionality. This can be prohibitive if the reader desires advanced scripting and non offline MRQ ultra photorealistic imagery. At the moment, AirSim seems best suited for tasks like testing control and reinforcement learning algorithms vs generating realistic looking data to train, test, and profile methods for object detection, passive ranging, tracking, etc. However, as we discuss in our use case sections, the flexibility and rendering quality from AirSim might prove to be good enough for many applications.

Our group uses AirSim for a few specific tasks: (i) autonomy scenarios that are too complex to be *implemented* in UE (e.g., requiring core engine modification and compilation or ridiculous UE Blueprint designs); and (ii) streaming custom (meta)data to an augmented reality device (e.g., HoloLens). An example of (i) is our multi-sensor DL-based EHD from UAVs [32, 33]. That research requires real-time algorithms (detection, tracking, and control that runs on an embedded device, e.g., NVIDIA Jetson), dynamic UAV interrogation of an object and scene, and exploration. While it is possible that some subset could be implemented in UE, an AirSim backbone simplifies life. An example of (ii) is real-time streaming of large voxel or point cloud data and an agent's internal *mental map* (objects, spatial relations, etc.) for human-robot teaming and augmented reality (AR) [34] (examples in Figures 16 and 17). In order to achieve this custom feat, we had to develop a workflow that routes data from UE and AirSim through ROS to Unity for real-time interaction on the HoloLens. In summary, we claim that AirSim is maturing and it is helpful for tackling tasks that are too cumbersome or not possible in UE directly. However, if the reader desires a pre-planned UAV flight with RGB imagery, odds are it can be achieved faster and to higher quality directly in UE.

4. Related Open Source Libraries

Not all DL and UAV tasks can be supported in UE and AirSim alone. In addition to libraries that our group is developing, we make use of a few open source tools. First, we use the Robotic Operating System (ROS) [35] for contexts involving augmented reality. While UE supports streaming to the HoloLens directly, we have found Unity to be a more natural fit for working with ROS and AR. Our workflow is to use UE and AirSim for simulation and data collection, and to transmit that data to Unity for processing via ROS message passing. A major reason for supporting this functionality is that under the hood, our team is constantly switching between data coming from a real drone, camera on the computer, and simulation. We are constantly changing where data is routed, e.g., NVIDIA Jetson for real-time UAV operations, a desktop, server, etc. The point is, ROS enables a great deal of flexibility, re-routing, and interfacing of DL and UAV information across I/O devices.

To work with 3D data outside of UE and Unity for custom algorithms and processing, we have made significant use of the Open3D library [36]. Open3D supports basic point cloud and voxel operations, and provides a Python OpenGL rendering context to help facilitate rapid prototyping and visualization (see Figure 7). Although Open3D can *display* point clouds with millions of points, it lacks many of the optimizations that would enable efficient processing and manipulation of large scale point clouds and voxel data. For this, we recommend OpenVDB [37] (by DreamWorks

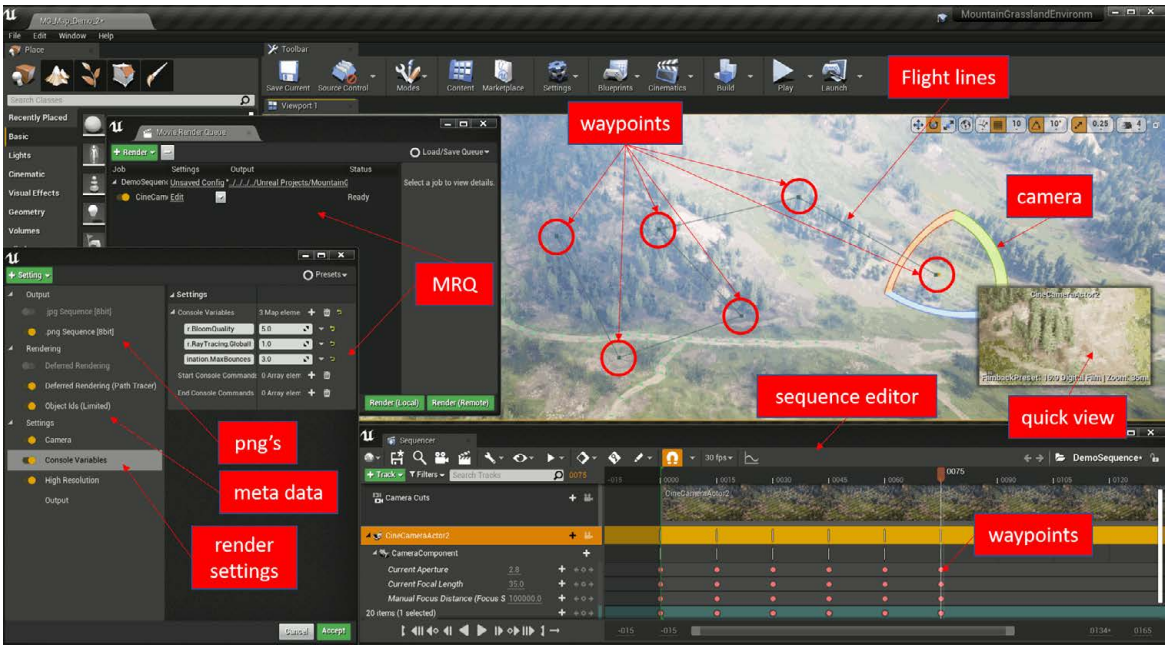


Figure 6. Example UAV lawnmower flight pattern data collection specified manually in UE. See article for additional details.

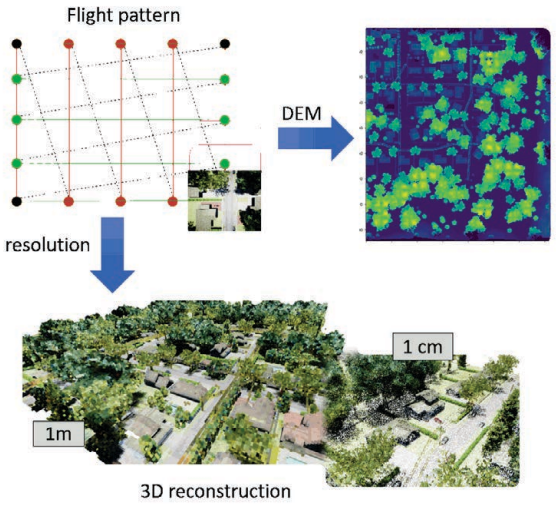


Figure 7. Example of a 3D point cloud and digital elevation model generated for a waffle flight UAV pattern in AirSim and UE.

Animation), a hierarchical data structure and suite of tools for the efficient storage and manipulation of sparse volumetric data discretized on 3D grids. In a big open outdoor scene (see Figure 6), we used OpenVDB to stream over 36,864,000 points in a 4.637 billion voxel space at 33 fps on a desktop PC. This brings us back to a theme of the current article. Our field is driven in part by tools, and we are at a convergence point due to low-to-no cost tools like UE, AirSim, PyTorch, Open3D, OpenVDB, ROS, etc.

5. Framework, Workflow, and Case Studies

This section illustrates our overall framework (Figure 8) with four example workflow use cases: (1) offline UAV data collection, (2) training of an online DL-based real-world object detector, (3) offline training, augmenting, and scoring a DL-based 3D passive ranging algorithm, and (4) online autonomy in support of human-robot teaming and AR.

5.1. Case 1: Offline UAV Data Collection Workflow

Case 1 highlights a workflow for collecting a low altitude UAV dataset with RGB imagery and ground truth (depth and per-pixel object or instance IDs). We assume that the reader has an environment loaded in the UE Editor; e.g., the Modular Neighborhood Pack [38] from the Unreal Marketplace [18] or a custom scene they perhaps built following Quixel's photorealistic tutorials [21]. Next, the user needs to create a Cine Camera Actor [23], set its camera settings (FOV, image resolution, etc.), select Add Level Sequence [24], and add the Camera Actor to the Track. The reader can then manually or via the Details panel move (translate) the camera (Cine Camera Actor) to a desired start location and rotation (e.g., nadir) and add a keyframe.[1] The reader needs to repeat this translation and rotation for each waypoint (intermediate location) in the flight pattern at desired temporal offsets[2] in the Sequencer timeline.[3] Next, the reader needs to add the MRQ Plugin, select their desired render settings

[1] UE operates by default in centimeters.

[2] A simple strategy is to regard each unit as a frame

[3] UE will automatically interpolate object properties between keyframes; but the user can override it if desired.

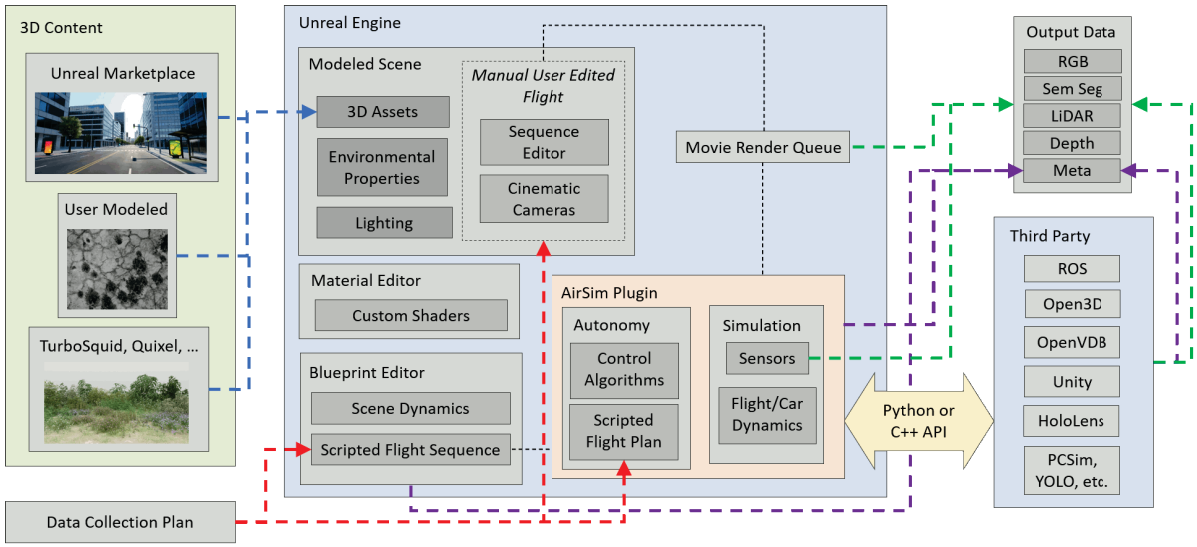


Figure 8. Example of overall predominantly open source photorealistic simulation framework discussed herein. Blue lines show scene modeling, red shows ways to carry out controlled data collection, green shows data generation, and purple shows corresponding metadata.

[22], and specify render options to generate per-pixel IDs and depth in addition to the default RGB imagery. Just like in the real world, the reader will need to plan their data collection, i.e., determine a desired ground sampling distance (GSD) based on camera FOV, number of pixels, UAV speed, altitude, etc. The resulting data, which is output into folders specified in the MRQ, are now ready for processing, i.e., detection, tracking, 3D mapping, etc. If 3D data is needed and it is not the focus of the readers research, then an open source structure from motion (SfM) or multi-view stereo (MVS) library like COLMAP [39, 40] can be used.

A different workflow is needed for AirSim. First, the user needs to create an appropriate AirSim configuration file (`settings.json`) with desired platform ("SimMode": "Multirotor"), sensors (e.g., RGB and LiDAR) [41], and sensor settings. Next, the user will create a simple multirotor drone controller in Python [25]. The reader can specify each location for the drone (client.simSetVehiclePose), which operates differently based on what is selected for "SimMode" in `settings.json`. "Multirotor" mode will use the AirSim control logic to move from waypoint to waypoint, while "ComputerVision" will instantaneously take the drone to a specified location. The point is, the user can program where to go, what data to collect (e.g., RGB imagery and depth via client.simGetImages, LiDAR via client.getLidarData, etc.), etc. It is important to note that AirSim makes it easy to poll the drone flight metadata[4], e.g., location and roll, pitch and yaw (client.simGetVehiclePose), which can easily be written out to file. Furthermore, as all this information is known, the reader can use it to generate 3D data (e.g., o3d.geometry.PointCloud.create_from_rgbd_image in the Open3D library). While AirSim provides great flexibility, the rendering quality is not as good as UE. However, in many of our DL studies, e.g., detectors like YOLO and Monodepth2 for PR, AirSim data often good enough.

5.2. Case 2: Real-World EHD Workflow

In [33], we demonstrated how to train a YOLOv5 [42] (object detection and localization algorithm) DL model in UE for a UAV equipped with RGB and IR cameras for explosive hazard detection (EHD). Specifically, in [33] we documented the generation of full scene EHD imagery in UE (Figures 9 and 10). Our goal was to increase our number of training samples across different environments, targets, and emplacement contexts for this otherwise class imbalanced under-sampled domain. While we set up our UE scene to look like environments of interest, no attempt was made to model a specific EHD target, environment, or clutter. The goal was to quickly set up a data collection that is not re-substitution and allows us to test if our algorithms generalize.[5] In that article, we also explored exporting images from 3D target objects and their shadows (see Figure 11). The goal is to leverage all possible environmental imagery and to increase the number of looks of targets in those contexts that is otherwise too expensive and time consuming to collect. This approach is deemed particularly useful because many DL detection algorithms only make use of imagery with labeled targets in it. As a result, many frames that are good to learn from go unused during train-

[4]To the best of our knowledge, this has to be done *externally* to the Sequencer Editor in UE by scripting a Blueprint if not using AirSim.

[5]Generalizability here refers to differences in real targets vs 3D EHD targets, differences in scenes, 3D models, and textures, and ultimately, subtle differences between ray tracing-based data generation vs a real sensor.

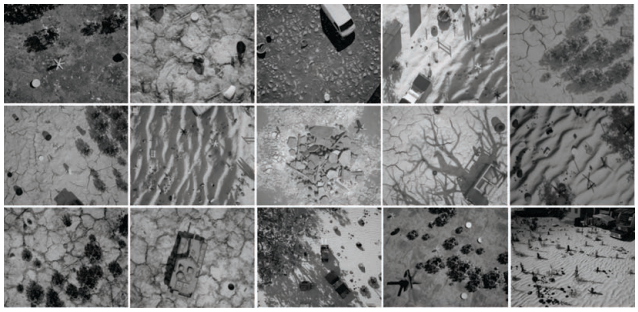


Figure 9. Example non-photorealistic UE scenes and imagery we built by mixing existing content to train an EHD DL detector [33].

Figure 10. Example of increased photorealism in UE for EHD.

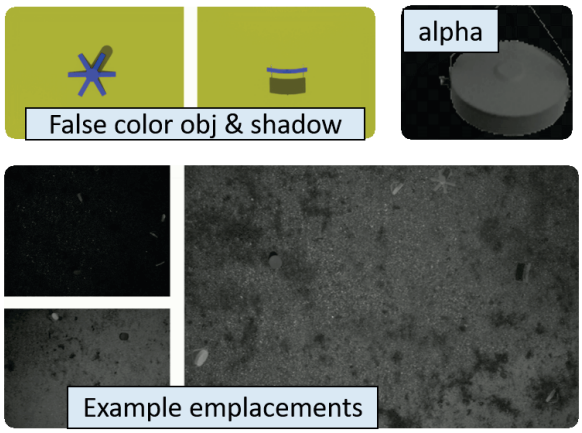


Figure 11. Example false color imagery and resultant alpha transparency EHD target templates from UE placed into real UAV drone imagery. Our insertion technique [33] combines UE and real UAV metadata to intelligently insert EHD templates.

ing. Therefore, this approach not only allows us to increase our training data, it allows us to fully make use of all collected UAV data.

Figure 12 is a summary of our DL-based object detection results as receiver operating characteristic (ROC) curves.[6] The reader can see that the real data model is the worst, followed by full simulated data, simulated objects inserted into real data, and the combination of all four. Our suspicion, which needs to be experimentally backed by more experiments, is that the training data has the least amount of variety, the full simulated data has more background, target,

[6]These results are not offline and hypothetical. These models are run in real-time on a Jetson for UAV demonstrations by our US government collaborators.

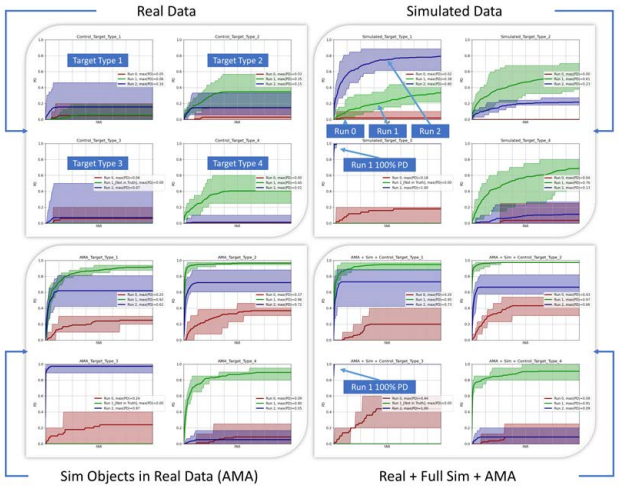


Figure 12. ROC results for YOLOv5 trained using real data, UE full simulated data only, real data with UE templates, and all combinations [33]. ROCs are divided into performance on four EH targets across three runs, and shading shows max, min, and average (the bold color lines) performance across a set of YOLOv5's. The y-axis are classifier accuracy and the x-axis is the number of mistakes (false alarms (FAs)). We do not publish FA rate units nor training set specifics due to the sensitive nature of EHD. The reader can still clearly see relative performance.

and clutter variation, the templates increased target object instances and increased real non-target background usefulness, and the combination of these is clearly the most beneficial. We also suspect that while our fully simulated data is not photorealistic, this might be to our advantage. That is, many *false correlations* are not present; the data highlighted the most relevant features like shape and contrast. Future DL and UAV research will be needed to understand and demonstrate just where simulation is most useful, e.g., backgrounds/targets/clutter not seen in training data, edge cases that will likely never be encountered in real data, increasing sampling in support of real data, etc. It should be noted, cases with very low performing ROC curves were associated with high altitudes and too few of looks on target, which also plagued a human observer, meaning lack of performance had less to do with the overall algorithm.

Last, in [32] we used UE for rapid research testing of UAV and environment contextual metadata driven fusion using fuzzy integrals. That is, on the fly construction of multi-algorithm and multi-sensor data fusion. Coordinating a UAV EHD collection is an expensive and time consuming process; designing the collection, acquiring ground truth, data collection with trained pilots, etc. Our aim is to bootstrap UE simulation for rapid testing of ideas to better inform our collaborators what to collect, vs our traditional take on iteration and trial and error. Figure 13 is a simple EH scene with the UAV at different altitudes, times

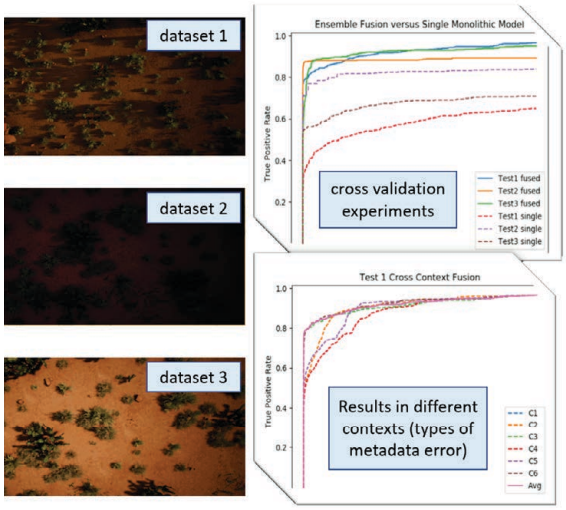


Figure 13. UE images and ROCs showing simulation predicted multi object detection algorithm fusion gain and degradation with respect to different types and amounts of metadata error. As in Figure 12, x-axis units are not reported due to sensitivity of EHD. The reader can still see and gauge relative algorithm performances.

of day, shadows, object emplacements, occlusions, etc. As discussed in [32], this setup aided our research process by allowing for quick specification and controlled variation of UAV and environment parameters to help us study the impact of fusion vs single algorithm processing and its degradation with respect metadata error; something not typically possible in real-world data collections due to factors like cost, time, and ability to collect accurate ground truth.

5.3. Case 3: Passive Ranging (PR) Workflow

Next, we highlight the use of UE simulation for PR. Existing PR datasets (e.g., Cityscapes and KITTI) are video sequences from a car. However, KITTI also provides a sparse LiDAR ground truth, which is incomplete, range limited, and error prone vs the perfect information we get in UE. Herein, we highlight results for the self-supervised Monodepth2 DL algorithm [43]. While metrics like Abs Rel, Sq Rel, RMSE, and log base 10 are frequently used for PR, real-world truthing makes answering certain questions hard if not impossible, e.g., long distance ranging, error as a function of object or feature type, etc. In general, lack of ground truth in real-world PR data is what has led to the use of self-supervised learning and hard to optimize loss functions involving photometric error (e.g., SSIM) and pose estimation error (e.g., cycle-consistency). Lack of quality truth also has big financial implications, like discussed earlier like Tesla's human data labeling and curation.

Figure 14 shows example imagery from a training and test UE dataset. While we showed quantitative results in Section 5.2, in this section we show qualitative results

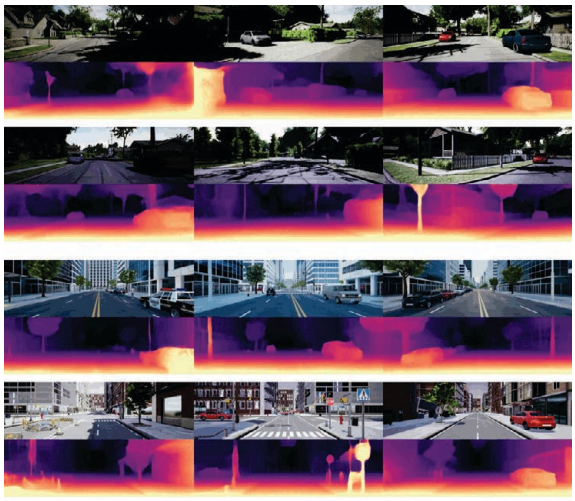


Figure 14. Monodepth2 output on UE data.

in Figure 14 for a KITTI real-world trained Monodepth2 model on simulated data. These examples are not "cherry picked", Monodepth2 does a great job on simulated UE data. The reason why we did not strive for extreme photorealistic imagery (use of AirSim vs generation in UE directly) is because we wanted to test if less than perfect imagery from UE4, which looks good to a human, is already convincing enough to a PR algorithm. It should be noted that there is little "overlap" between our arbitrary selected urban and rural simulated scenes and real KITTI environment. After performing these experiments, we then (Figure 15) trained new Monodepth2 models (i) from our UE dataset, (ii) the KITTI dataset, and (iii) a combination of simulated UE data transfer learned with KITTI data. As the reader can see, the simulated data does well on KITTI (on various objects but also the horizon) and the combined model does best. We know these results are qualitative and preliminary. A future detailed study is needed to understand where simulated data can be used to advance PR.

5.4. Case 4: Human-Robot Teaming Workflow

The point of this last section is to demonstrate a more complicated example of the proposed workflow. This case study uses AirSim to get data from UE and to control the UAV. 3D data and imagery is managed via Open3D in Python and is transmitted to a C# Unity client (which can be on the same or a different computer) via ROS. The rendered output is sent over WiFi to the Microsoft HoloLens headset for real-time interactive AR (see Figure 17). While UE and AirSim are used for data generation and UAV control in this example, we have also been able to use this AR interface to control a real DJI drone. To control the drone (real or simulated), the user can grab a blue arrow hologram (bottom left image in Figure 17) and move it around (twist command messages are sent via ROS). The upper left image

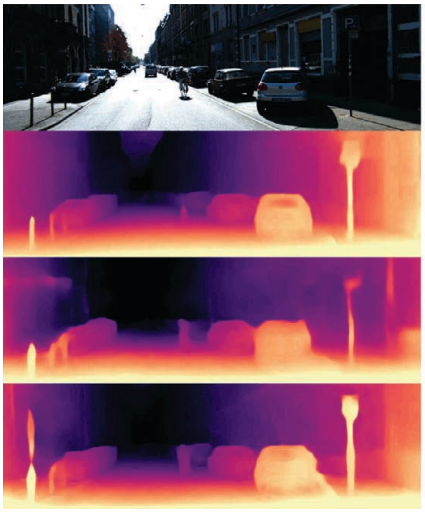


Figure 15. Monodepth2 on real, simulated, and combined data.

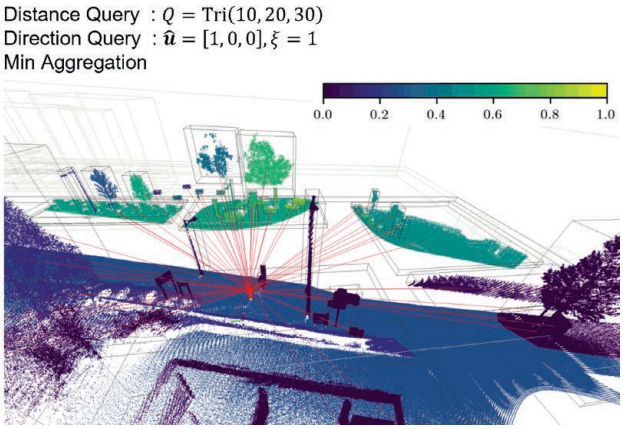


Figure 16. Attributed relation graph (ARG). Objects are outlined by bounding rectangles. Human is shown in red. Also shown are linguistic queries based on distance, direction, object type, and neighboring relations. Brighter colors are query satisfaction; e.g., user telling a UAV "find all close objects in front of this person."

is a HoloLens virtual monitor that shows the streaming UE or real UAV video data. The bottom right image shows the user selecting a region of interest to go to and/or enlarge for AR interrogation. The top right image shows the enlarged point cloud and RGB real-time streaming feed. This case study shows how this workflow for data collection and real-time DL algorithm training, testing, and evaluation can be easily adapted using open source libraries into an interactive demo for human-robot (in this case a UAV) teaming.

Figure 16 is an example of rendering simplified metadata vs raw data for use in AR. The tracked person, a reference object, is rendered in red, the point cloud is color coded based on the degree to which points satisfy a linguistic query (aka the person can talk to the UAV), and segmented objects are shown with axis aligned bounding boxes. Linguistic spatial queries (e.g., "close and to the left") need to be specified in the reference object, UAV (relative to its heading), or user's viewpoint (reference frame). The point is, real time streaming and our UAV's "mental map" (representation of scene) can be processed, visualized, and interacted with via the outlined tools and workflow.

6. Conclusions and Future Work

In summary, DL has changed the landscape of AI/ML for application specific tasks at the expense of a dependency on large collections of labeled supervised data. This is a bottleneck for many domains like UAVs. While the field is constantly in search of new innovate theory, practical advancements are equally welcome. Herein, we outline a framework built on open source tools and example workflows are demonstrated for offline data collection, online object detection, 3D mapping, and human-robot interaction via AR. In order to close the gap and facilitate reproducible research, online video tutorials for each

Figure 17. Example UE data in the Microsoft AR HoloLens. Real time video feed on a virtual AR monitor, real time point cloud streaming, and dynamic interrogation UAV data.

case study are provided at https://github.com/MizzouINDFUL/UEUAVSim. As we demonstrated, quantitatively and qualitatively, research can be accelerated, models can be improved, and hybridization's of real world and photorealsitic simulated data are of utility. While preliminary, our article shows that the convergence of these tools–for academia, but also industry–are extremely promising and worth the time investment. In future work, we will continue to explore interesting fringes of sim in offline, online, and closed loop ways to improve ML/AI for applications like UAVs in ways that cannot be realistically achieved due to financial, time, and/or practical real-world limitations. In order to concrete the role of photorealistic simulation, akin to how data augmentation has become an every day tool to DL researchers, new workflows and studies with quantitative metrics need to be performed.

References

[1] "Artificial intelligence stocks: The 10 best AI companies," 2021. [Online]. Available: https://money.usnews.com/investing/stock-market-news/slideshows/artificial-intelligence-stocks-the-10-best-ai-companies

[2] "Final report: National security commission on artificial intelligence," 2021. [Online]. Available: https://www.nscai.gov/wp-content/uploads/2021/03/Full-Report-Digital-1.pdf

[3] "Elon Musk and tech heavies invest 1 billion in artificial intelligence," 2021. [Online]. Available: https://money.cnn.com/2015/12/12/technology/openai-elon-musk/

[4] "Tesla AI chief explains why self-driving cars don't need LiDAR," 2021. [Online]. Available: https://venturebeat.com/2021/07/03/tesla-ai-chief-explains-why-self-driving-cars-dont-need-lidar/

[5] "If data is the new oil, these companies are the new Baker Hughes," 2021. [Online]. Available: https://fortune.com/2020/02/04/artificial-intelligence-data-labeling-labelbox/

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

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

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

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

[10] P. Martinez-Gonzalez, S. Oprea, A. Garcia-Garcia, A. Jover-Alvarez, S. Orts-Escolano, and J. Garcia-Rodriguez, "UnrealROX: An extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation," *ArXiv e-prints*, 2018. [Online]. Available: https://arxiv.org/abs/1810.06936

[11] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4CV: A photo-realistic simulator for computer vision applications," *Int. J. Comput. Vision*, vol. 126, no. 9, p. 902–919, Sep. 2018. [Online]. Available: https://doi.org/10.1007/s11263-018-1073-7

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

[13] K. Nouduri, K. Gao, J. Fraser, S. Yao, H. AliAkbarpour, F. Bunyak, and K. Palaniappan, "Deep realistic novel view generation for city-scale aerial images," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 10 561–10 567.

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

[15] "Storytelling reimagined," 2021. [Online]. Available: https://www.unrealengine.com/en-US/solutions/film-television

[16] "Real-time ray tracing," 2021. [Online]. Available: https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/RayTracing/

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

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

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

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

[21] "Creating Photoreal Cinematics with Quixel," https://www.unrealengine.com/en-US/onlinelearning-courses/creating-photoreal-cinematics-with-quixel, (Accessed: 1 March 2021).

[22] "How to Use the Movie Render Queue for High-Quality Renders," https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/RayTracing/MovieRenderQueue/, (Accessed: 1 March 2021).

[23] "Using Cine Camera Actors," https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/Sequencer/HowTo/CineCameraActors/, (Accessed: 1 March 2021).

[24] "Sequencer," https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/Sequencer/Overview/, (Accessed: 1 March 2021).

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

[26] S. Shah, "AirSim-W: A simulation environment for wildlife conservation with UAVs," in *ACM SIGCAS*, June 2018. [Online]. Available: https://www.microsoft.com/en-us/research/publication/airsim-w-a-simulation-environment-for-wildlife-conservation-with-uavs/

[27] P. Pueyo, E. Cristofalo, E. Montijano, and M. Schwager, "CinemAirSim: A camera-realistic robotics simulator for cinematographic purposes," 2021.

[28] "Gazebo," http://gazebosim.org/, (Accessed: 1 March 2021).

[29] T.-C. Wu, S.-Y. Tseng, C.-F. Lai, C.-Y. Ho, and Y.-H. Lai, "Navigating assistance system for quadcopter with deep reinforcement learning," in *2018 1st International Cognitive Cities Conference (IC3)*, 2018, pp. 16–19.

[30] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "AirSim drone racing lab," *arXiv preprint arXiv:2003.05654*, 2020.

[31] S. Shah, A. Kapoor, D. Dey, and C. Lovett, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and Service Robotics*, pp. 621–635, November 2017. [Online]. Available: https://www.microsoft.com/en-us/research/publication/airsim-high-fidelity-visual-physical-simulation-autonomous-vehicles/

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

[33] B. Alvey, D. T. Anderson, J. M. Keller, A. Buck, G. Scott, D. Ho, C. Yang, and B. Libbey, "Improving explosive hazard detection with simulated and augmented data for an unmanned aerial system," in *SPIE*, 2021.

[34] A. R. Buck, D. T. Anderson, J. M. Keller, R. H. L. III, and G. Scott, "A fuzzy spatial relationship graph for point clouds using bounding boxes," in *FUZZ-IEEE*, 2021.

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

[36] "Open3D," http://www.open3d.org/, (Accessed: 1 March 2021).

[37] "OpenVDB," https://www.openvdb.org/, (Accessed: 1 March 2021).

[38] "Modular Neighborhood Pack," https://www.unrealengine.com/marketplace/en-US/product/modular-neighborhood-pack, (Accessed: 1 March 2021).

[39] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[40] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[41] "AirSim Settings," https://microsoft.github.io/AirSim/settings.html, (Accessed: 1 March 2021).

[42] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, yxNONG, A. Hogan, lorenzomammana, AlexWang1900, A. Chaurasia, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Durgesh, F. Ingham, Frederik, Guilhen, A. Colmagro, H. Ye, Jacobsolawetz, J. Poznanski, J. Fang, J. Kim, K. Doan, and L. Yu, "ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4418161

[43] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," 2019.

Chapter 9

Title: Metadata enabled contextual sensor fusion for unmanned aerial system based explosive hazard detection

Venue: SPIE Defense + Commercial Sensing,

Date Published: Aril 12, 2021

PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Metadata enabled contextual sensor fusion for unmanned aerial system-based explosive hazard detection

Matthew Deardorff, Brendan Alvey, Derek Anderson, James Keller, Grant Scott, et al.

Matthew Deardorff, Brendan Alvey, Derek T. Anderson, James M. Keller, Grant Scott, Dominic Ho, Andrew Buck, Clare Yang, Brad Libbey, "Metadata enabled contextual sensor fusion for unmanned aerial system-based explosive hazard detection," Proc. SPIE 11750, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXVI, 117500C (12 April 2021); doi: 10.1117/12.2586340

SPIE.

Event: SPIE Defense + Commercial Sensing, 2021, Online Only

Metadata Enabled Contextual Sensor Fusion for Unmanned Aerial System-Based Explosive Hazard Detection

Matthew Deardorff[a], Brendan Alvey[a], Derek T. Anderson[a], James M. Keller[a], Grant Scott[a], Dominic Ho[a], Andrew Buck[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

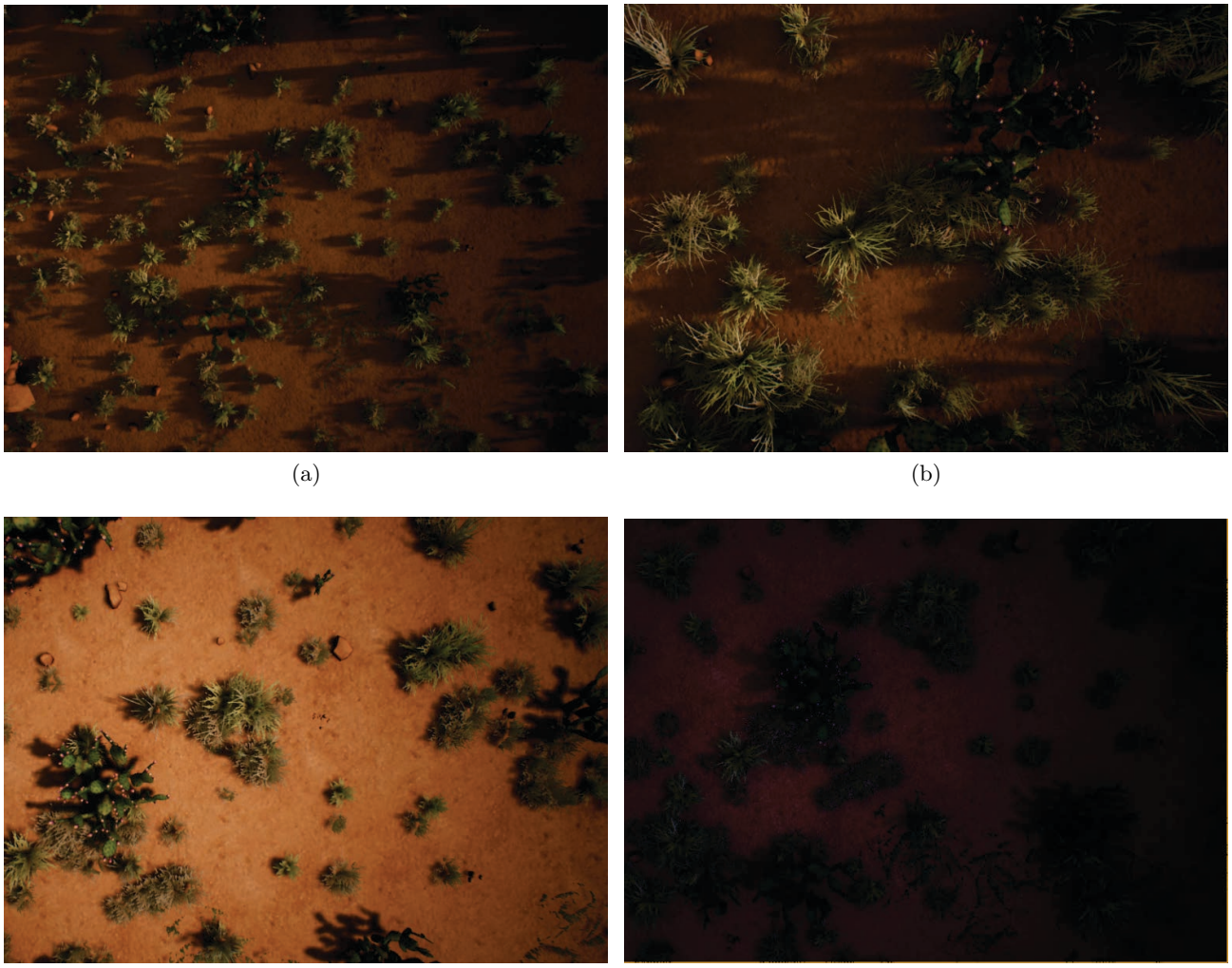
Numerous real-world applications require the intelligent combining of disparate information streams from sensors to create a more complete and enhanced observation in support of underlying tasks like classification, regression, or decision making. An often overlooked and underappreciated part of fusion is context. Herein, we focus on two contextual fusion challenges, incomplete (limited knowledge) models and metadata. Examples of metadata available to unmanned aerial systems (UAS) include time of day, platform/sensor position, etc., all of which have a potentially drastic impact on sensor measurements and subsequently our decisions derived from them. Additionally, incomplete models limit machine learning, specifically under-sampling of training data. To address these challenges, we investigate contextually adaptive online Choquet integration. First, we cluster and partition the training metadata. Second, a single machine learning model is trained per partition. Third, a Choquet integral is learned for the combination of these models per partition. Fourth, at test/run time we compute the degree of typicality of a new sample to our known contexts. Fifth, our trained integrals are decomposed into a bag of underlying aggregation operators and a new contextually relevant operator is imputed using a combination of the metadata clustering and observation statistics of the integral variables. This process enables machine learning model selection, ensemble fusion, and metadata outlier detection, with subsequent mitigation strategy identification or decision suppression. The above ideas are demonstrated on explosive hazard detection using surrogate data simulated by the Unreal Engine. In particular, the Unreal Engine is used because it provides us with flexibility to explore the proposed ideas across a range of diverse and controlled experiments. Our preliminary results show improved performance for fusion in different contexts and a sensitivity analysis is performed with respect to metadata degradation.

1. INTRODUCTION

The task of detecting and classifying explosive hazards (EH) from unmanned aerial systems (UAS) is a difficult one, in part due to the drastically varied environments and platform conditions one can expect to operate in/across. Detection in a hot desert at noon is a significantly different problem than detection in a frozen tundra at night. Detection from a UAS with a nadir sensing angle at 10 feet is different from a UAS with a sensor slant angle at 100 feet. Furthermore, the sensors used on UASs–e.g., RGB, IR, LiDAR, multi-spectral, etc.–all experience different sensor phenomenology depending on the environmental conditions and material properties of sensed objects. Figure 1 is an example that highlights environment and UAS (platform) variation. Herein, we refer to the above variations as *contexts*, as they are sources of information which we can use to enhance the performance of an underlying task like EH detection. In this paper, we propose an online and adaptive ensemble-based fusion scheme for EH detection that is driven by environment and platform metadata.

Before we delve into UAS-based EH detection (EHD), we briefly discuss related efforts. EHD technologies vary drastically. An early and well-known technique is the so-called "metal detector", which can be used to detect metal 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

Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXVI, edited by Steven S. Bishop, Jason C. Isaacs, Proc. of SPIE Vol. 11750, 117500C
 · doi: 10.1117/12.2586340



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

Figure 1: Detection and localization algorithms are tasked with understanding objects in a variety of *contexts*, requiring robustness across factors like scale, color, illumination, and texture. Often, even a single location can look very different depending on platform altitude, look angle, time of day, etc. However, this information often goes unused in algorithms. The proposed contextual fusion scheme attempts to determine proper strategies based on metadata features which help inform context.

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 a UAS platform for EHD. Advantages of UAS, versus hand-held or ground vehicle deployment is it keeps humans at safer standoff distances and a UAS can in theory act like each of the above technologies. That is, it has the potential to search wide areas and dynamically interrogate regions of interest, likely through the use of a squad or swarm of UASs, with different sensors at different look angles. In this article, we limit our analysis to the use of a single UAS with multiple imaging and position sensors.

Adaptive fusion is not a new idea. For example, in Ref. 4, Hichem, Gader, et al. proposed a creative

algorithm, context extraction for local fusion (CELF). We highlight and discuss this algorithm because we also use the Choquet integral (ChI). In particular, Hichem combined the fuzzy C-means (FCM) clustering algorithm and the ChI. They formulated a single joint optimization. Hichem partitions the input space based on the features to be fused. Our work, aka the current article, differs as we initialize contexts based on otherwise unused metadata features such as altitude and temperature and learn independent operators from subsets of data. We also approximate the entire integral, aka all the underlying capacity variables, of which there are 2^N for N inputs. Hichem instead focuses on the "densities", i.e., the capacity defined on only the singletons, and an imputation strategy (the Sugeno λ fuzzy measure). We focus on learning the entire capacity because the tuples beyond the singletons capture interaction between sources. This is something we expect to occur and it can result in performance gain. Last, Hichem's fusion is driven by clustering. Herein, we exploit our recent integral transfer learning[5,6] and data-driven eXplainable AI (XAI) methods.[7,8] XAI allows us to identify what parts of a model (integral) were not approximated (sufficiently) from training data. Integral transfer learning allows us to transfer fusions, or parts of fusion, across integrals. In contrast to prior methods, our proposed method provides a method to measure similarity of new samples relative to prior contexts and determine the sufficiency of the fusion operator being imputed. These similarity methods are important as our goal is to optimize the aggregation operator based on information previously observed.

We also explore the use of the Unreal Engine[9] to create synthetic imagery. The graphical fidelity offered by these simulated environments proves as a useful surrogate for the otherwise difficult task of assembling large amounts of varied, UAS-captured data. Advantages include training real models from simulated data and rapid prototyping and experimenting with ideas that can later be transfered to real world experiments and solutions.

1.1 Machine Learning Models Derived from Limited Data Sample Sets

Data is king in modern machine learning. The performance of neural networks and other supervised learning models are intimately linked with the kind, quality, and diversity of training data provided. In a perfect world we could assume that good quality data can be obtained with enough time and patience, but this is rarely the case. It is in our interest to develop well performing classification models that have been exposed to only a limited amount of training data.[6,10] This is especially relevant in the domain of UAS based vision as it can be difficult to obtain large amounts of appropriate aerial data.

The problem of limited training data is one which informs how the rest of this architecture is structured, and must be considered at every step. One known problem caused by small amounts of training data is *overfitting*. Overfitting occurs when a learning model memorises the solutions to training data but is unable to generalize to data it has not seen before. This is in part due to the training data lacking adequate diversity, as the training data does not represent all possible variations of data that might be discovered. Our current article attempts to mitigate the overfitting problem by relying on a collection of niche experts, as opposed to a single model which can universally solve the problem. As a result, models are only expected to perform well on data similar to what has been seen before and their use is restricted when performance expectations are low.

Another problem encountered due to limited training data is specifically tied to our fusion operator of choice, the ChI. As described later, the ChI partitions the input space based on the sorting of the inputs, where each unique sort results in a different method of combination. Because of this, we ideally would observe every possible sort in the training data so that an optimization algorithm is able to estimate all the values that are required. This is often not the case for a number of reasons. First, it can simply be difficult to encounter each sort through random chance. For N inputs there exist $N!$ possible sorts, meaning an adequately sized training set is required just to get one sample from each sort. A second reason the observed sorts are important is specifically tied to the domain of fusing *strong learners*. A strong learner is a learning model which usually produces only extreme (strong) values. For example, a strong classifier would only label detections as 0 (not the class) or 1 (is the class), but would rarely label something as 0.5 to denote uncertainty. Due to this, it is often the case that all models to be fused agree on a class label of either 0 or 1. Thus, the only sort encountered is the default from when all values are the same. This heavily biases the ChI training procedure, as it is possible that nearly all observed data consists of only a few unique walks. If unencountered sorts ever show up later in testing data, the operators will be poorly optimized to handle them. Our method attempts to mitigate this problem by transferring learned values from an integral that has observed a particular sort.

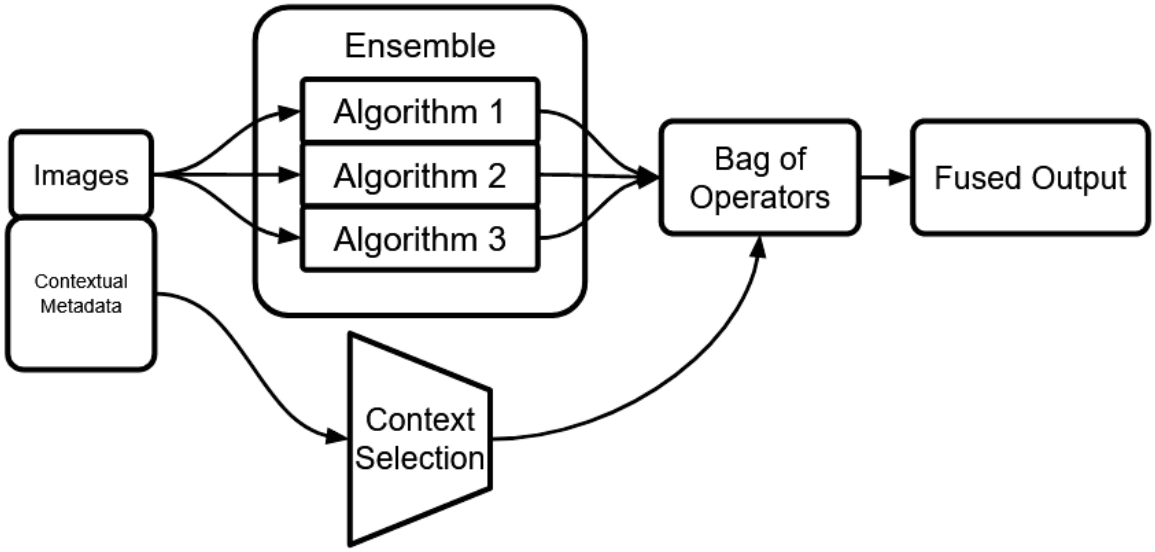


Figure 2: The general flow of images and metadata in our ensemble. Multiple algorithms are treated as sources of evidence to be fused together, while metadata such as altitude, temperature, and time of day inform the system how to construct the best possible aggregation operator.

1.2 Ensemble of Neural Networks

A common technique to mitigate the reliance on a single black-box neural network is to train multiple networks which operate in parallel on the data. While this goes by different names in the community, we refer to it herein as an ensemble neural network. Each of the networks produces its own estimate of the target value, before each of the estimates are aggregated back into a single score. The precise method of aggregation depends on the architecture, though our method uses the ChI, in part due to it's capability of producing human-readable explanations of the fusion. This article creates an ensemble of networks with homogeneous architectures trained on varying subsets of data. A different popular technique in ensemble architectures is to vary the architecture of the individual networks (depth, number of parameters, etc.) but provide each network with complete data. The reader can refer to Refs. 11–14 for our recent publications on ensembles of heterogeneous architecture neural networks for broad area scanning, land classification, and object detection in remote sensing. Figure 2 illustrates the flow of data and metadata in the ensemble architecture proposed herein. The pieces of this ensemble are described in greater detail in following sections.

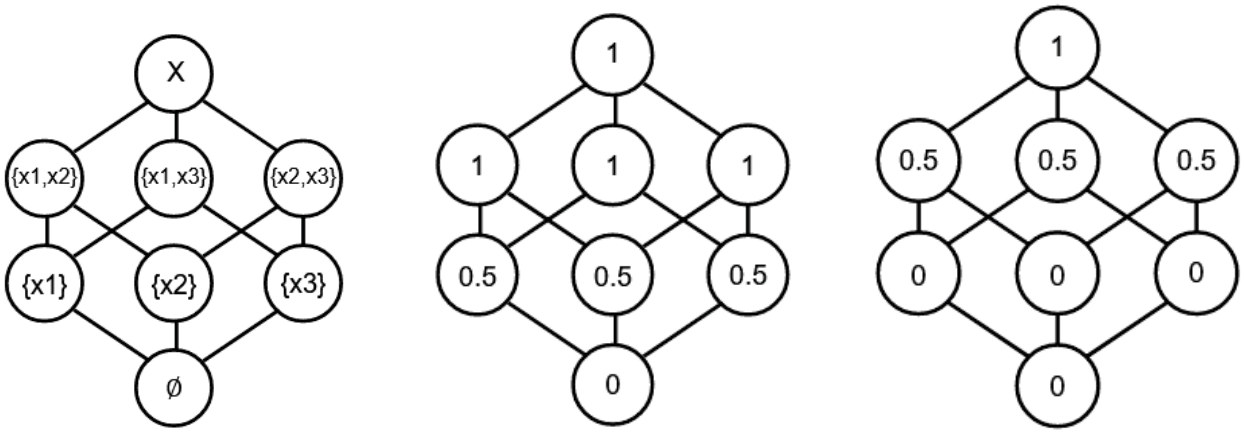
2. METHODS

2.1 Fuzzy Measure and Fuzzy Integral

The fuzzy integral (FI) is a well studied tool in information fusion which defines a family of nonlinear operators. The integral is evaluated on a fuzzy measure (FM), $g : 2^X \rightarrow R^+$, which is a function that has two properties on finite X: (i) (boundary condition) $g(\emptyset) = 0$, and (ii) (monotonicity) if $A, B \subseteq X$, and $A \subseteq B$, then $g(A) \leq g(B)$. The Choquet integral (ChI) is a *type* of FI,[15] given by

$$\int \mathbf{h} \circ g = C_g(\mathbf{h}) = \sum_{j=1}^{N} h_{\pi(j)}(g(A_{\pi(j)}) - g(A_{\pi(j-1)})), \tag{1}$$

where $\mathbf{h}$ is the integrand ($h(\{x_i\}) = h_i$ is the input from source i), $A_{\pi(j)} = \{x_{\pi(1)}, \ldots, x_{\pi(j)}\}$, $g(A_{\pi(0)}) = 0$, and π is a sort such that $h_{\pi(1)} \geq h_{\pi(2)} \geq ... \geq h_{\pi(N)}$. In our case, $h_{\pi(j)}$ is the jth largest return out of all algorithms, and $a \subseteq A$, $g(a)$ denotes the "worth" of a subset of algorithms. Thus, the ChI fuses evidence from each source based on the worth of a subset of sources.



(a) Set-Labeled Hasse (b) Optimistic Fusion (c) Pessimistic Fusion

Figure 3: Hasse diagrams depicting different strategies of fusion for $N = 3$ inputs. An optimistic fusion like the one depicted in 3b averages the two largest input values. A pessimistic operator 3c averages the two smallest values. In algorithm fusion it is common to see pessimistic operators due to their redundancy as all algorithms must agree on a high value, i.e., unanimous consent.

It is relevant to note that the ChI is an operator which can be learned from data using various solvers. For example, in Ref. 16 we use quadratic programming (QP), in Ref. 11 we proposed constraint free full FM gradient descent optimization for supervised neural networks, and in Ref. 17 we proposed an evolutionary algorithm for efficient genetic operators on non-convex optimization surfaces. In this paper, we use the QP to learn a number of ChIs, trained on subsets of the data, which can be selected from based on what we believe the context to be.

A convenient way to visualize the ChI is in the form of its underlying Hasse diagram, where nodes in the diagram represent the g values of the power set of A in lexicographic order from bottom to top, left to right. For example, if $X = \{1, 2, 3\}$ the lexicographic ordering of the power set $P(X)$ is

$$P(X) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

The edges in the diagram represent monotonicity constraints, meaning nodes in the upper layers are greater than or equal to the nodes connected below them. A *walk* up the Hasse diagram refers to a given sort on $\mathbf{h}$ and the resulting path taken from the bottom of the diagram to the top. Therefore, each walk defines a unique fusion operation the ChI is capable of. Figure 3 depicts example diagrams which are possible fusion strategies for three sources ($N = 3$).

2.2 Context Matters

The discrete ChI described above partitions the input space based on the sorting of inputs and each partition results in a different fusion operator. One way to think of these partitions is that they provide *context* as to what operator is most appropriate. An example interpretation of this for our current paper on UAS-based detection of EHs using multiple neural net algorithms is: "if Algorithm 1 has the greatest return, listen primarily to it. Otherwise, take the average of all the algorithms". We call this kind of context the *internal context*, as it is based solely on the data that is directly being fused. Specifically, Equation 2.1 informs us that each internal operator context is a linear convex sum (LCS) function, when $g(\emptyset) = 0$ and $g(X) = 1$.

However, there is more than just internal context in problems such as ours. Consider the task of EHD from a UAS. There are wildly different conditions in which the UAS might be flown, such as high altitudes, low altitudes, bright days, or dark nights. These are all normal operating conditions for such a system, yet the sensory feedback in each of these conditions will be distinct. As a result, the algorithms that we use on this data must be robust to these variations. This article aims to better handle this kind of context, what we call the *external context*, of our fusion problem. Our method attempts to identify these unique external contexts by clustering the metadata

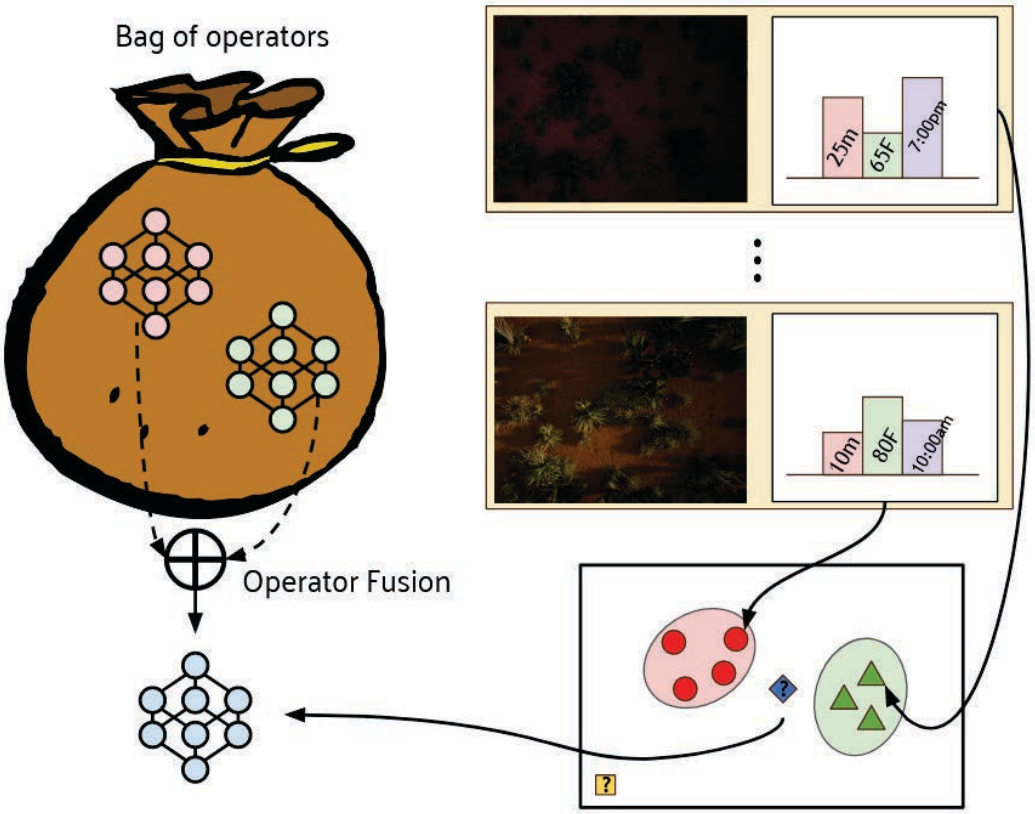


Figure 4: Illustration of the propose methodology. Metadata feature vectors are generated from training data and they are clustered to define initial contexts (red circles and green triangles). In this diagram we show an example image and the prototype per cluster. Next, a different ChI that combines a set of neural network classifiers is built per context (red and green Hasse diagrams). When a new sample (observation) belongs to a known context, the appropriate ChI is used. However, if a new sample, e.g., blue diamond, does not belong to a known context but it is similar to known contexts, then a new operator is built on the fly. In the event that a new sample is extremely different from anything that we have seen before, e.g., the yellow box outlier, then the system can decide to take no action or an operator can be built if the system is expected to always operate.

obtained from the UAS (platform) and environment. Specifically, we use the GPS reported altitudes, recorded temperatures, and time of day as initial features to identify unique external contexts. Figure 4 illustrates our scheme of clustering metadata and using them to train unique fusion operators.

2.3 Metadata Feature Encoding

As the following metadata features will be used to inform the system of which context to associate the data with, it is important to consider their encoding. A common problem resulting from the use of disparate feature types is that one feature can dominate the space, e.g., have notably higher magnitudes. In our case, if we assume that the range of observed temperatures is on average larger than the range of observed altitudes, then the distance between temperatures will predominantly drive the distance measure. While there are other ways to handle this using techniques such as categorical encodings, a simple solution is to normalize the values (denoted as z below) on a scale of [0,1] based on minimum and maximum observed values,

$$z_{scaled} = \frac{z - z_{min}}{z_{max} - z_{min}}. \tag{2}$$

Special attention should also be paid to how the time of day is encoded, as it is a cyclical feature. Consider what happens if time of day was encoded as a scalar value in the range 0 to 23 hours (12am to 11pm). If we measure

the distance from $t = 23$ to $t = 1$ (11 : 00pm to 1 : 00am), the Euclidean distance is 22, though clearly those two time periods are only two hours apart. A simple yet clever way to avoid this problem is to split the time feature into two values given by

$$t_{sin} = \sin(\frac{2\pi t}{23}), t_{cos} = \cos(\frac{2\pi t}{23}). \tag{3}$$

When these two values are plotted as (x, y) pairs in the range $[0, 23]$, the result is a circle. This makes it a more appropriate encoding for use with Euclidean distance, as it now mimics distance on an actual clock, i.e., $t = 0$ and $t = 23$ are adjacent, while any two values offset by 12 hours maximize the distance. Note, in this paper we explore a few metadata. In future work we will investigate the inclusion of more metadata and their respective pleasing semantic conditioning.

2.4 Determining Initial Contexts Through Clustering

As already discussed, our ensemble of neural networks is driven by context. To this end, we cluster the training metadata features into an initial set of contexts via the possibilistic c-means (PCM) algorithm.[18] The PCM is a mode seeking method that operates on a finite set of M samples $Z = \{\mathbf{z}_1, ..., \mathbf{z}_M\}$ relative to a specified number of c clusters. Unlike the k-means clustering algorithm, which is a crisp partitioning technique (i.e., every sample belongs to one, and one only, cluster), the PCM is a mode seeking algorithm. The PCM returns c clusters, which depending on the choice of underlying metric (e.g., Euclidean, Mahalanobis distance, GK metric, etc.) results in c prototypes, e.g., $C = \{\mathbf{c}_1, ..., \mathbf{c}_c\}$, and a partition matrix $[U]_{ik} = u_{ik}, i = 1, ..., C, k = 1, ..., M$, where u_{ik} is the typicality of sample $\mathbf{z}_k$ to cluster i. Unlike the k-means algorithm, the PCM allows samples to belong fully to multiple clusters and outliers can now be represented and detected. The PCM typicality degrees are especially useful at evaluation time, as it gives us a degree to which we believe a new data point belongs to a known context (cluster). As described later, we adapt our fusion strategy for new data (UAS observations) based on how similar it is to what has been seen before. In our implementation of the PCM, we use Euclidean distance and we initialize the cluster centers with the fuzzy c-means algorithm output because it helps us estimate PCM bandwidth parameters and it provides robustness over random initialization.

A problem ever present in all clustering algorithms is determining an optimal value for c. Herein, we use the fuzzy partition coefficient[19] (PC), an internal cluster validity index. The PC attempts to measure how well a set of data was partitioned based on the membership values of each class given by

$$F_c(U) = \frac{tr(U * U^T)}{M}, \tag{4}$$

where U is the fuzzy partition matrix segregated into c classes, $*$ is matrix multiplications and $tr()$ is the trace, or sum of squared diagonals. A desired c can be selected from an index like the PC by looking for the maximum (or minimum) index value, or a trend (e.g., elbow) in the c plot. While the PC is used herein, it should be noted that there are more sophisticated internal (Xie and Beni index, Dunn, DBI, etc.) and external (Rand, etc.) cluster validity measures in the community, e.g., see Ref. 20. If the reader desires to implement and use the methodologies contained herein, we recommend that a more robust cluster validity index be used.

2.5 Realtime Fusion

The above sections describe a set of offline computations on training data. The result is a set of contexts, neural classifiers (one per context), and subsequent aggregation operators (one ChI per context). This section outlines an online (aka runtime) selection mechanism to determine what contexts a new sample belongs to based on the typicality values provided by the PCM algorithm.

The selection process we developed breaks down into three distinct cases. The first case is when the data to be evaluated is highly typical of one and only one existing context, meaning we believe we have an appropriate fusion operator to use. The second case occurs when the data to be evaluated is highly atypical compared to all known contexts, meaning we are operating in an unknown context and will subsequently resort to using a default fusion operator. Herein, we explore the idea of a system taking an action, but a user could instead take no action because we are unable to predict how the system will respond. The third and final case occurs when

the data to be evaluated belongs to more than one context. This is a case where we will fuse multiple operators together to create a more appropriate adaptive fusion scheme.

The selection process above can be defined for data point z_i, where u_{ij} is the typicality of z_i in cluster j, and α and β are user defined upper and lower typicality thresholds respectively. The selection function is

$$\mathbf{g} = \begin{cases} g_k & u_{ik} \geq \alpha \text{ and } \forall j, j \neq k, u_{ij} < \alpha \\ g_{\text{default}} & \forall j, u_{ij} < \beta \\ combine(\mathbf{u}, g_1, ..., g_N) & \text{else,} \end{cases}$$

where condition one says pick a single FM/ChI when we are in a known context. Condition two is how we respond to a metadata outlier and condition three outlines the fusion of our fusions from metadata. The above scheme still leaves a few questions. First, what operator do we choose in case two? This is the case where the system is exposed to what appears to be a thus-far unseen context. We explore multiple methods, including a simple mean average of the network confidences and averaging the operators from all previously observed contexts. The simple mean is a natural place to start, as it credits equal worth to each of the sources to be fused. This is useful as it makes no assumptions about the worth of individual sources in unseen contexts, though this is also the method's weakness. If there is a clear pattern in the fusion strategies consistent across all contexts then the simple average will disregard this, throwing away information that could be useful as a default fusion strategy. The second method explored attempts to handle this problem by averaging the set of trained ChI operators. Here we define the average of a set of operators to mean calculating the average value on a per-node basis in the Hasse diagram. This produces a fusion scheme which retains any dominant fusion strategies common across contexts, while maintaining the monotonicity constraints required by the ChI. If there is no obvious fusion scheme across all contexts (such as being generally optimistic or favoring a particular source), this averaged operator produces an operator that is in a way smoothed, and pulled closer to an operator that resembles the mean*.

This solution inspires the combination method we use for case three (see Figure 5). In this case, our clustering is tight enough that a new data point can reasonably be considered to be in one of multiple contexts. To resolve the ambiguity, the operators in question are combined through a weighted average where the weight is determined by the relative strength of the typicality values. The $combine(\mathbf{u}, g_1, ..., g_N)$ function above is

$$\mathbf{g} = \sum_{i=1}^{N} \frac{u_{ik}}{T(\mathbf{u})} g_i, \tag{5}$$

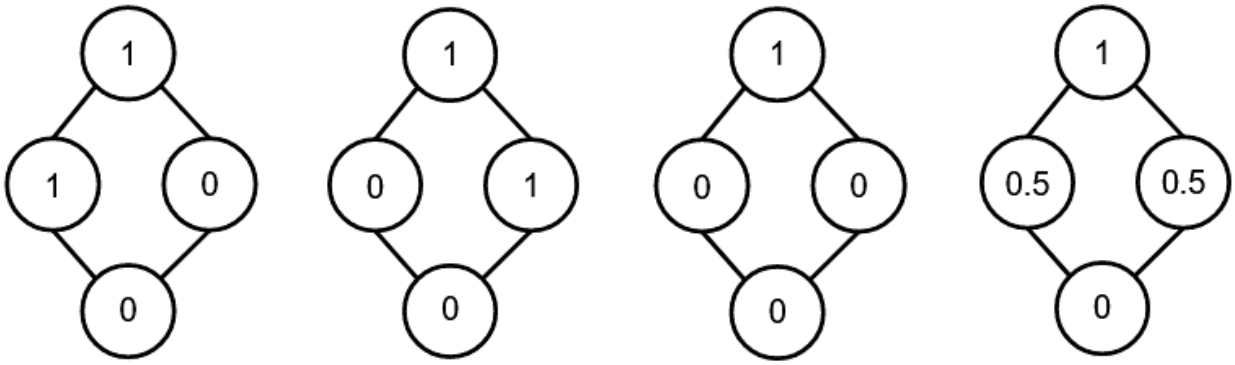
where γg ($\gamma \in [0, 1]$) is defined as $\gamma g(A), \forall A \in 2^X$, $g = g_i + g_j$ is defined as $g(A) = g_i(A) + g_j(A), \forall A \in 2^X$, $T(\mathbf{u})$ is the sum total of typicalities for sample k, $T(\mathbf{u}) = \sum_{i=1}^{N} u_{ik}$. Other possible ways to aggregate FMs include the operators we outlined in Ref. 21 relative to evolutionary optimization, a simple t-norm like the minimum or product on each variable, or a set of t-norms and t-conorms outlined in Ref. 22 by Yager.

On a final note, we wish to comment that this is an initial study. That is, the above method only exploits metadata cluster membership values. This helps us build a new operator on-the-fly based on how similar the sample is to our past contexts. In future work we will also look at the internal context in each ChI (runs in the Hasse diagram) and combine it with our probabilistic estimate of how well that operator was supported.[7,8] The idea being, there is no point relying on an operator that has not been sufficiently learned from data. Instead, a method like our ChI transfer learning[5] or similar should be engaged to derive an appropriate data informed internal operator. Last, these two disparate concepts need be combined.

3. PRELIMINARY EXPERIMENTS AND ANALYSIS

In this section we explore our proposed methods on a set of synthetic imagery meant to imitate changes in sensor phenomenology we might expect from use in different UAS environments. Imagery was generated using the Unreal Engine, as it allows automatic data-labeling and complete control of environment parameters. This provides us needed flexibility to explore ideas like adaptive fusion. That is, our methods are not bottle necked by

*Assuming uniformly random FM g values



(a) First operator (b) Second operator (c) Minimum combine (d) Average combine

Figure 5: What is a reasonable scheme to combine FMs? 5a and 5b signify fusion schemes which listen entirely to a single source, a result that is likely to happen in our system if a given algorithm performs especially well in a certain context. If we combine based on a minimum operator or allow the quadratic solver to recompute on all data, the result is 5c. This operator is very pessimistic and will require both algorithms to agree on an answer, something that may be unlikely to happen. 5d is the result of a node-wise average, and maintains a degree of worth for individual algorithms.

real world factors like time and ultimately expense of collecting and labeling EH data. In our experiments we use the You Only Look Once version 5 (YOLOv5) network architecture[23] for EH object detection and localization, as it provides estimates of bounding boxes and confidences to fuse across, with a well documented implementation for easy training. We compare our method against a general model which has been exposed to all training data and is not a part of an ensemble. We examine what happens when the system is exposed to contexts that are not present in the training data, as well as good strategies for combining existing operators when the metadata is ambiguous between multiple existing contexts.

It should be noted that we are intentionally not disclosing which environments, EH targets, and EH emplacement strategies we simulated. The targets and environments were determined in conjunction with our US Army Night Vision and Electronic Sensors Directorate (NVESD) collaborators. The targets are above ground objects (versus buried), they have moderate-to-low clutter (e.g., are often partially obscured by natural objects like a bush), we use a generic (aka similar to what you would find on the commercial market) RGB camera, and we believe that objects have enough pixels on target for detection. The goal was not to push the system to extreme breaking points, i.e., camera spectra being insufficient to detect an object or too few of pixels to even have an object that can be detected and discriminated from clutter. The point is to create a challenging and real world achievable problem that we can push to the point of failure and to compare the different avenues outlined herein. Furthermore, the UAS platform conditions were nadir (looking straight down) and a few arbitrary altitude variations were explored. In this article we do not consider all common scenarios, e.g., varying factors like aircraft speed and subsequent motion blur that would result. The point is, exact altitudes, environments, camera parameters, and etc. are just a surrogate herein to test the proposed algorithms. These experiments and environments are not real, but they are set up to mimic similar conditions to data that we have seen to date; meaning they are not overly simple and unrealistic. This paper is not a documentation of YOLOv5 for EHD on specific use cases. We would not report that information due to the real-world EH threat. In summary, what the reader can take away from the following experiments is the relative performance of the algorithms, their variations, and sensitivities.

3.1 Metadata Enabled Fusion versus Single Model

We start by evaluating the proposed algorithms on six sets of synthetic training data, where each dataset represents a different context that a UAS could experience during normal operation. As mentioned above, our goal is generality and diverse training data, not specific operational experiments. Specifically, the training runs

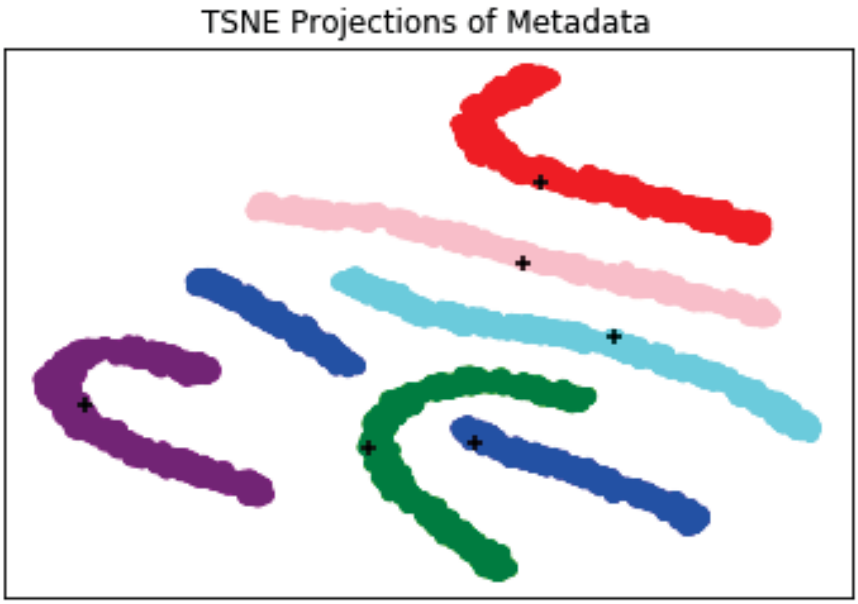


Figure 6: The metadata of our training sets reduced from four to two dimensions by TSNE. Color-coding is provided by PCM assigned clusters. Cluster centers are marked with a plus.

consist of "high" and "low" altitude variants at solar noon (aka no shadows and ideal radiance conditions), afternoon (long shadows, darker), and night (very dark, most difficult) data. Below, these training datasets are referred to as "data set 1", "data set 2", and etc., and test datasets are simply referred to as "test 1" (Test1), "test 2" (Test2), etc. The test data sets contain otherwise unseen data (i.e., not resubstituion) with targets that are under heavy occlusion, shadows, and extreme angles so that they should be sufficiently difficult tests to evaluate our method. We feel like there is no loss of generality in our paper, as one can see performance in context, out of context, and with respect to outlier observations. As each training run has associated higher dimensional metadata (four dimensions herein) that we visualize using the dimensionality reduction techniques t-distributed stochastic neighbor embedding[24] (TSNE). Figure 6 shows the metadata from the six training runs, along with PCM assigned clusterings.

While the clusters are clearly separable in our experiments, when transitioning to the real world we do expect the data to be noisier. This can lead to a larger bandwidth parameter in the PCM algorithm, which will ultimately cause typicalities to increase across the board as the algorithm becomes more relaxed in what it considers a cluster. In short, these experiments are less likely produce the third case in the context selection procedure, as it is unlikely for a given point to have high typicalities across multiple clusters. It can also be noted that the reason the clusters manifest as rope-like in the projection is due to the time of day feature increasing linearly through time while the other features are pulled from a normal distribution. Furthermore, we study this separable problem because it mimics the way that many real world collections occur. That is, data is often collected for a short number of consecutive days in a specific geographic area. We would not expect to have data from twenty four hours a day at all geographic locations. Last, it is our strong belief that if the proposed algorithms do not work for the scenarios explored herein, it is unlikely that they will work for the more challenging scenarios. And as stated above, an advantage of the Unreal Engine is we can generate a lot of data, of which all attributes are known. This is rarely the case in the real world as labeling can be sparse and error prone and documentation is never complete nor perfectly accurate (e.g., amount of cloud cover, temperature at each geospatial location, etc.).

Figure 7 shows the relative performance of our ensemble network (solid lines) compared to a single, out-of-the-box YOLOv5 model (dotted lines). Ideally, a good receiver operating characteristic (ROC) curve, where the x-axis is mistakes and y-axis is the positive detection rate, is the "zero FAR, one PD" (which is almost always achievable in real datasets). Most often, people look for "quick rises" (increase in PD with little to no mistakes)

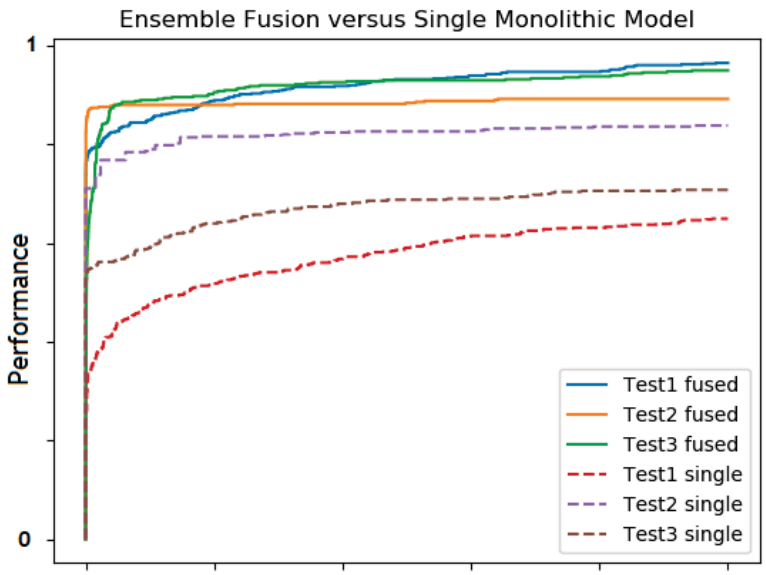


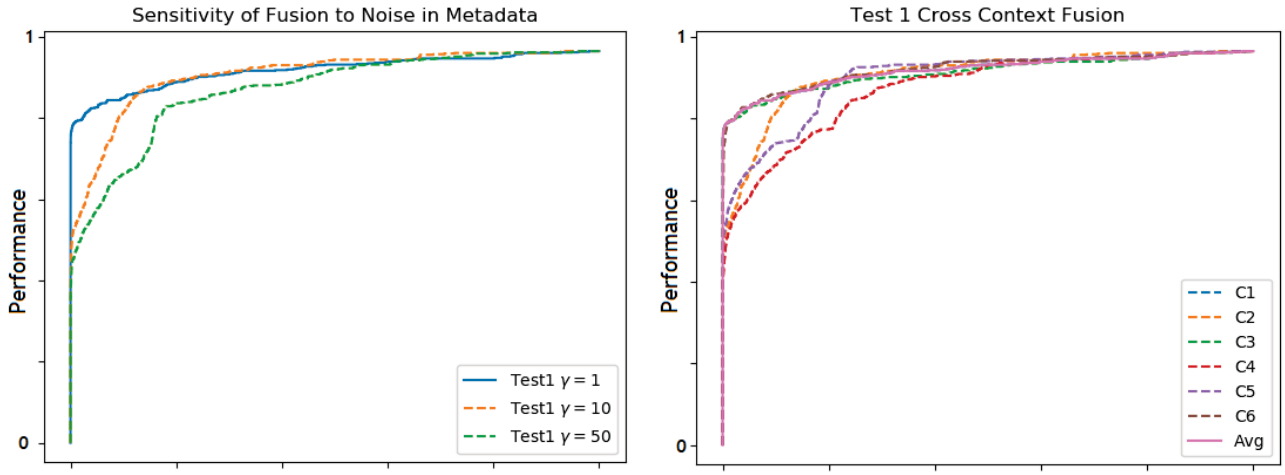
Figure 7: The proposed adaptive fusion scenario compared to a basic YOLOv5 architecture on three test scenarios. Test sets comprised of seen and unseen contexts.

versus plateaus (no detections but more false alarms) or "linear climbs" (aka you have a 50% chance of calling something target or a false alarm). Thus, on the three test contexts evaluated, our ensemble method performs better than the standard model in all cases. This is similar to what we observed relative to fusing, with a fixed versus adaptive strategy, a set of heterogeneous neural networks for land classification and object detection in remote sensing.[7,12–14] It should be noted that the total amount of data to train the ensemble is the same as the single model, though the single model was responsible for learning solutions across all of that data, while the ensemble was free to optimize a smaller subset of that data. While unproven, we believe this experimentally highlights the need to strike a balance between generalizeable models that perform well on all sorts of data and models that are experts in a more limited domain.

3.2 Sensitivity to Noise in Metadata

The above experiment is useful in the regard that it helps us understand operation in ideal scenarios. However, our method is reliant on additional data (metadata) provided by the UAS platform and/or environmental metadata. A benefit of using simulation is that we have complete control over the fidelity of this data. That is, we can simulate noise and other errors, which are likely to appear when the algorithm is used in the real world. To better understand the robustness of our method to such errors, we construct the folowing two sensitivity experiments.

Figure 8a depicts the degradation of fusion performance as noise is introduced to the associated metadata. Again, we are not disclosing which metadata (altitude, time of day, etc.) lead to the biggest degradation due to the sensitive nature of EHD. Specifically, our metadata was generated based on normal distributions with varying levels of standard deviation. The gamma variables in figure 8a are scalar multipliers to the base standard deviation, resulting in more erratic (and less representative) metadata. Thus, $\gamma = 1$ is a single standard deviation (normal operating conditions), $\gamma = 10$ is 10 and $\gamma = 50$ is 50 times more noise, respectively. Semantically, the simulated types of errors lead to incorrect identification of current contexts, or relying on the generated default operator. It can be noted that the training clusters present in the synthetic experiments are nicely separable and spaced apart. It is unclear (future work) how these algorithms will work in the case of extremely close contexts. If the metadata is a good context identification scheme then contexts would be expected to be distinct and separate in space. However, if context, and or collected data are very close, e.g., model for 1pm and another model for 1:30pm, then we might expect that the trained classifiers and fusions should be similar. The point is, further analytical studies with performance characterization or experiments need to be performed in order to understand the impact of adaptive fusion for such scenarios. In summary, this experiment (Figure 8a) informs



(a) Noise induced in metadata (b) All fusions compared on a single test run

Figure 8: Accurately capturing context data is important for the performance of our system. Excessive noise in the data can lead to incorrect operator selection as seen in 8a. The entire range of learned operators can also be evaluated on a single test run such as in 8b.

us that there is indeed an impact, but if our simulation is a close model to the operating conditions of a UAS for EHD, metadata noise is a concern but detection is not significantly impacted; we still get 40% detection with no error and do better than chance from that point on.

In Figure 8b, our second experiment, we consider what happens when the selection process consistently identifies an incorrect context. Whereas the last experiment showed us random controlled variations, this experiment tests extreme cases. This experiment is achieved by comparing the correct fusion operation to the rest of operators generated in the training process. These experiments results in six unique contexts being identified, thus, we can perform the fusion operation tailored to each of the six contexts. It should be no surprise that when the system is intentionally given incorrect contextual information, the performance of the fused algorithm is lowered, e.g., a morning model is used to detect EHs later in the day at a higher altitude. In general, we can see that the correct fusion operator (the generated weighted average) results in the best possible performance, though a few of the existing operators manage to perform equally well.

3.3 Explainability of Fusion

In this section we explore if there are any additional benefits of the proposed methodologies. One our underlying goals was to realize a trustworthy system versus opaque model. Our aim is to make a mathematical system that can be extended in future work, e.g., factor in additional advanced EHD or physics knowledge. The system outlined herein is really "level 1 visual intelligence". That is, the system processes imagery and tries to learn robust low-level spatial and spectral features. Ideally, the system outlined herein is not the entire system, but one stage in the detection and understanding pipeline. The end goal being a "higher level environmental understanding" algorithm. To that end, one of the primary benefits of using the proposed ChI fusion strategy, as opposed to a black-box neural network is that the ChI can be opened up and examined after the fact to determine what fusion strategies were prevalent. That is, the ChI is a centralized and explicit model versus a distributed and implicit neural network. However, we remark that in Ref. 11 we put forth a way to encode and optimize a full ChI as a neural network, without loosing interoperability. In this section we take a look at what was learned in the previous experiments. We only report a subset of explanations. The reader can refer to our past works[7,8,11,25] for our wider set of XAI fusion tools that generate statistical, graphical, local, and linguistic explenations.

In Murray et al.,[7,8] data-centric indices were proposed as a way of evaluating the kind and quality of data that was used to train the ChI. Of particular note is the walk-visitation calculation which describes what part

of the Hasse diagram is well supported by data, i.e., how many (and which) internal contexts has a trained ChI observed and approximated. This can be used to identify "missing data", or perhaps more appropriately labeled "missing model variables." The trends reported herein are consistent across contexts, therefore we select and focus on the arbitrary Context 1. In this context, only 19% of the total possible walks received even a single piece of support. With this being a six source fusion, $6! = 720$ sorts are possible on the data, though only 137 of those were seen. Therefore, the integral is only approximately 20% approximated, which is not good. However, as we discuss in our fusion for remote sensing work,[7,8,12,13] most real world datasets do not have sufficient volume nor diveristy. While many datasets claim to have both, our prior work showed that even for the higher volume datasets, their estimated fusion model values are frequently less than 20%. Meaning, our simulated scenario has arguably more diversity than we would encounter in practice. This makes sense to us, as the Unreal Engine lets us produce more data across different context.

Furthermore, of the walks that were taken, 65% of the time the data took the sort (1, 2, 3, 4, 6, 5), meaning that one walk almost completely dominates the operator. This is a common thing to encounter when training a ChI. That is, this is the default sort order. Usually that means that a bulk of the data is in agreement, is all saying the same thing. This is a typical behavior of strong learners, which we might expect from the YOLOv5 algorithm. Furthermore, this most prevalent walk corresponds to a **max** operator, meaning this particular operator tends to be optimistic. This is a difficult run in the Hasse to interpret. There is an fundamental entanglement that we cannot break apart. That is, this run is both the default sort order run and a valid case of what happens when algorithm 1 is more confident than algorithm 2, followed by 3, and so forth. As such, did we need the max to solve the latter or did it simply pick the max because it was an arbitrary selection when all algorithms say the same thing. Furthermore, the densities (values of the lowest level in the Hasse diagram) are $g(\{x_1\}) = 0.99, g(\{x_2\}) = 0.07, g(\{x_3\}) = 0.99, g(\{x_4\}) = 0.77, g(\{x_5\}) = 0.48, g(\{x_6\}) = 0$, showing that the fusion is often based entirely on the largest confidence value, as long as the largest confidence does not come from source two or six. Further analysis[7,8,11,25] would be required to separate these variables to determine which are supported by data and we should trust.

This trend continues for most of the other contexts. Due to specific algorithms performing very well in each context (as the training procedure is therefore resubstitution), the fusion operators in those contexts weight those sources heavily. In future work we will look to sample and study validation data to minimize this effect. However, this is not the case for the default operator that was learned, as it was exposed to all data and it did not perceive a clear superior in the sources. As described in section 2.5, we explored multiple methods to generate default operators including an average aggregation and retraining on all data. While it would be difficult to display the full Hasse diagrams here (visually and with respect to page count), the operator that was retrained on all data resembles a **min** operator, while the average aggregation resembles a **mean**. This means that it is nearly impossible for the retrained operator to produce a high output, as all six algorithms would need to agree on a detection with a high confidence (something that rarely happens.) While it may seem a bit counter-intuitive to do all of this machine learning only to end up with something similar to what could be guessed at from the start (using a mean to aggregate sources), we believe that encouraging a less pessimistic model generalizes better to unseen data. The reader needs to keep a few things in mind. First, this is not conclusive and it is not a proof. It is merely an observed behavior of our experiments and experimental setup; i.e., simulated scenes, trained YOLOv5 classifiers, quadratic solver, etc. Thinking beyond our experiments, it is reasonable to expect that a high quality model trained for a specific context could prove to be *optimal*; versus the unachievable single model trained on all possible data or its ensemble approximation. Furthermore, we might expect that a mean like operator is an, on average, least worse strategy for outlier metadata scenarios. Last, when contexts truly overlap, something not explored yet, a mixture of models could prove to be a more robust approximation; similar to the performance gain we observed using fusion in Figure 7. Last, if we assume that each of these models and fusions are derived at least in part from data, then the models will always fundamentally have missing pieces. The point is, the above conclusion from our papers experiments are in no way conclusive. They are an experimental observation that we can use as intuition to set up the next and better approach.

4. CONCLUSION AND FUTURE WORK

This article proposes a metadata enabled adaptive fusion scheme for UAS-based EHD which attempts to discover the underlying contexts data was collected in to better inform the fusion operation. The offline determination

of what constitutes a context is made by the possibilistic c-means (PCM) algorithm, a clustering algorithm which provides typicality values that describe to what degree a data point is typical of a given cluster. Once the training metadata is clustered, a set of context specific YOLOv5 location and detection classifiers are built, one per cluster. Finally, a Choquet integral (ChI) aggregation operator is trained for each context.

At evaluation time, the typicality values provided by the PCM allows the system to make an intelligent decision of whether or not an appropriate fusion scheme has already been trained. If the new data is sufficiently similar to an existing context then we are able to use the associated operator directly. If the new data is highly atypical from all previous contexts or similar to multiple contexts then the system uses a default strategy or it creates a new fusion scheme on the fly based on weighted interpolations of existing operators.

We evaluated the above methods on a set of synthetic imagery generated in the Unreal Engine, a process which allows us to circumvent the otherwise tedious process of obtaining large amounts of varied UAS data. Our results showed that there is benefit across the board in taking a metadata driven ensemble of our context dependent classifiers. Furthermore, we showed that while our system, as expected, is sensitive to metadata perturbation, the resultant ROC curve performance is still encouraging. Last, we showed additional sensitivity analysis experiments where we intentionally tried to destroy the algorithm. We note that this scenario is rare and might never be encountered in practice. However, the experiment reinforced our expected behavior of the system. That is, when out of context classifiers are used, performance is not ideal. However, our metadata fused result remains resilient. In summary, these preliminary experiments are encouraging.

In the future, we will evaluate how well simulator informed models transfer to real environments. We will apply intuition developed through our setup and experiments to an in depth investigation for each component in a real UAS scenario, e.g., metadata, its similarity, context prediction, etc. for GPS, IMU, and environmental factors. Furthermore, we only achieved a first step of adaptive fusion herein. That is, we use clustering to inform the construction of an on the fly fusion operator. In future work we will advance this model to include the factors discussed above, like internal context and its degree of approximation in a ChI for a given context. We want to advance this adaptive fusion method to let us produce operators that are similar to prior operators when we understand their performance. This may involve transferring solutions in and across models and metadata clustering typicality. Our goals are to determine if a well-trained model in a context outperforms an ensemble and to find optimal operators for addressing outliers. We intend to move away from experimentation and to rely on analytical proofs when possible to achieve this end. While preliminary experiments and the method are encouraging, there remains a great deal of future work.

5. ACKNOWLEDGEMENTS

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] Abdallah, A. C. B., Frigui, H., and Gader, P., "Adaptive local fusion with fuzzy integrals," *IEEE Transactions on Fuzzy Systems* **20**(5), 849–864 (2012).

[5] Murray, B., Islam, M. A., Pinar, A., Anderson, D., Scott, G., Havens, T., Petry, F., and Elmore, P., "Transfer learning for the choquet integral," 1–6 (06 2019).

[6] Kakula, S. K., Pinar, A., Islam, M. A., Anderson, D., and Havens, T., "Novel regularization for learning the fuzzy choquet integral with limited training data," *IEEE Transactions on Fuzzy Systems* , 1–1 (2020).

[7] Murray, B., Islam, M. A., Pinar, A., Anderson, D., Scott, G., Havens, T., and Keller, J., "Explainable ai for the choquet integral," *IEEE Transactions on Emerging Topics in Computational Intelligence* **PP**, 1–10 (07 2020).

[8] Murray, B., Islam, M. A., Pinar, A. J., Havens, T. C., Anderson, D. T., and Scott, G., "Explainable ai for understanding decisions and data-driven optimization of the choquet integral," in [*2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*], 1–8 (2018).

[9] Epic Games, "Unreal engine."

[10] Pinar, A. J., Havens, T. C., Islam, M. A., and Anderson, D. T., "Visualization and learning of the choquet integral with limited training data," in [*2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*], 1–6 (2017).

[11] 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).

[12] Scott, G. J., Hagan, K. C., Marcum, R. A., Hurt, J. A., Anderson, D. T., and Davis, C. H., "Enhanced fusion of deep neural networks for classification of benchmark high-resolution image data sets," *IEEE Geoscience and Remote Sensing Letters* **15**(9), 1451–1455 (2018).

[13] Scott, G. J., Hurt, J. A., Marcum, R. A., Anderson, D. T., and Davis, C. H., "Aggregating deep convolutional neural network scans of broad-area high-resolution remote sensing imagery," in [*IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*], 665–668 (2018).

[14] Scott, G. J., England, M. R., Starms, W. A., Marcum, R. A., and Davis, C. H., "Training deep convolutional neural networks for land–cover classification of high-resolution imagery," *IEEE Geoscience and Remote Sensing Letters* **14**(4), 549–553 (2017).

[15] Narukawa, Y. and Murofushi, T., [*Choquet integral and Sugeno integral as aggregation functions*], 27–39, Springer Berlin Heidelberg, Berlin, Heidelberg (2003).

[16] Islam, M. A., Anderson, D. T., Pinar, A. J., and Havens, T. C., "Data-driven compression and efficient learning of the choquet integral," *IEEE Transactions on Fuzzy Systems* **26**(4), 1908–1922 (2018).

[17] Islam, M. A., Anderson, D. T., Petry, F., and Elmore, P., "An efficient evolutionary algorithm to optimize the choquet integral," *International Journal of Intelligent Systems* **34**(3), 366–385 (2019).

[18] Krishnapuram, R. and Keller, J. M., "The possibilistic c-means algorithm: insights and recommendations," *IEEE Transactions on Fuzzy Systems* **4**(3), 385–393 (1996).

[19] Ross, T. J., [*Fuzzy C-Means Algorithm*], 358, Wiley (1995).

[20] Moshtaghi, M., Bezdek, J. C., Erfani, S. M., Leckie, C., and Bailey, J., "Online cluster validity indices for performance monitoring of streaming data clustering," *International Journal of Intelligent Systems* **34**(4), 541–563 (2019).

[21] Islam, M. A., Anderson, D., Petry, F., and Elmore, P., "An efficient evolutionary algorithm to optimize the choquet integral," *International Journal of Intelligent Systems* **34** (09 2018).

[22] Yager, R. R., "A measure based approach to the fusion of possibilistic and probabilistic uncertainty," *Fuzzy Optimization and Decision Making* **10**, 91–113 (June 2011).

[23] 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).

[24] van der Maaten, L. and Hinton, G., "Viualizing data using t-sne," *Journal of Machine Learning Research* **9**, 2579–2605 (11 2008).

[25] Murray, B. J., Anderson, D. T., Havens, T. C., Wilkin, T., and Wilbik, A., "Information fusion-2-text: Explainable aggregation via linguistic protoforms," in [*Information Processing and Management of Uncertainty in Knowledge-Based Systems*], Lesot, M.-J., Vieira, S., Reformat, M. Z., Carvalho, J. P., Wilbik, A., Bouchon-Meunier, B., and Yager, R. R., eds., 114–127, Springer International Publishing, Cham (2020).

Chapter 10

Title: Improving explosive hazard detection with simulated and augmented data for an unmanned aerial system

Venue: SPIE Defense + Commercial Sensing,

Date Published: Aril 12, 2021

Improving explosive hazard detection with simulated and augmented data for an unmanned aerial system

Brendan Alvey, Derek Anderson, James Keller, Andrew Buck, Grant Scott, et al.

Brendan Alvey, Derek T. Anderson, James M. Keller, Andrew Buck, Grant Scott, Dominic Ho, Clare Yang, Brad Libbey, "Improving explosive hazard detection with simulated and augmented data for an unmanned aerial system," Proc. SPIE 11750, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXVI, 117500B (12 April 2021); doi: 10.1117/12.2586342

SPIE.

Event: SPIE Defense + Commercial Sensing, 2021, Online Only

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

Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXVI,
edited by Steven S. Bishop, Jason C. Isaacs, Proc. of SPIE Vol. 11750, 117500B
 · doi: 10.1117/12.2586342

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

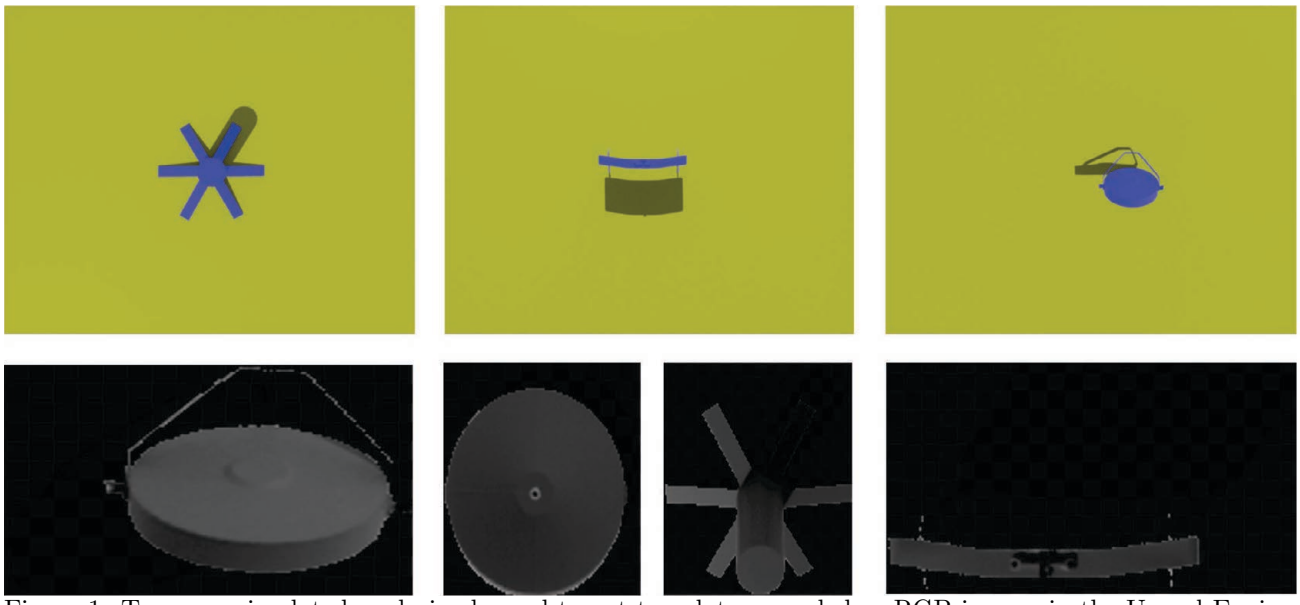


Figure 1: Top row: simulated explosive hazard target templates encoded as RGB images in the Unreal Engine. Bottom row: extracted grayscale target templates.

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

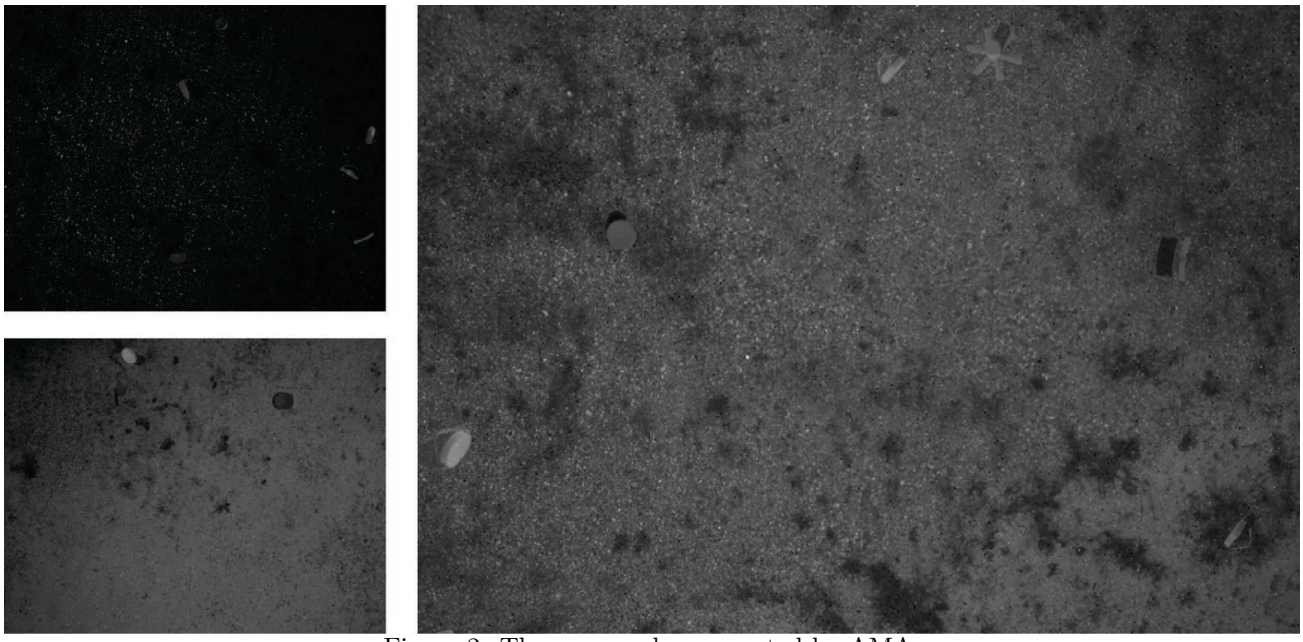


Figure 2: Three examples generated by AMA.

Algorithm 1 Altitude Modulated Augmentation (AMA)

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

Algorithm 2 Hot Cold Modification

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

Algorithm 3 Compression Modification

Given an image, I:
Determine $img\_max = max(I)$.
Determine $img\_min = min(I)$.
Set $max\_compression = 25$.
if $max\_compression * 2 > img\_max - img\_min$ **then**
    $max\_compression = 0.5 * (img\_max - img\_min)$
end if
Generate a random number, r between 0 and 1.
$compression\_level = r * max\_compression$
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

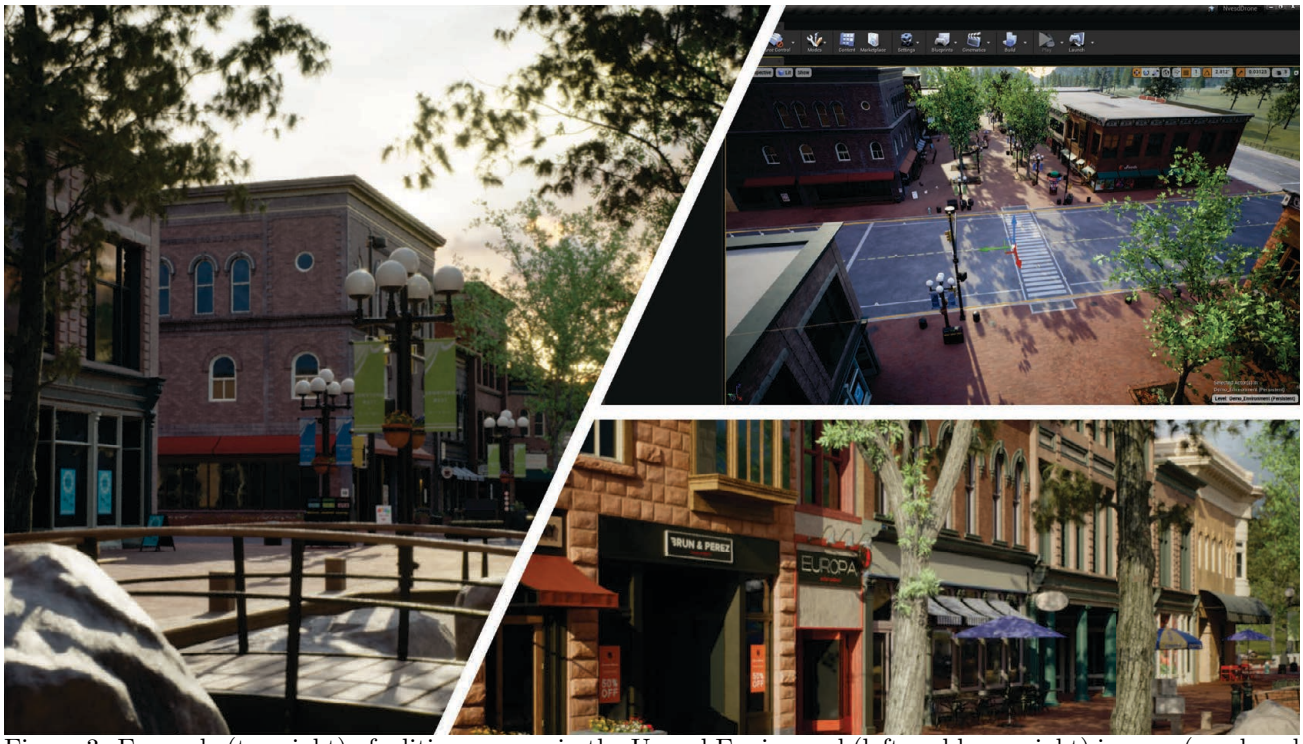


Figure 3: Example (top right) of editing a scene in the Unreal Engine and (left and lower right) images (produced using ray tracing and the Movie Render Queue).

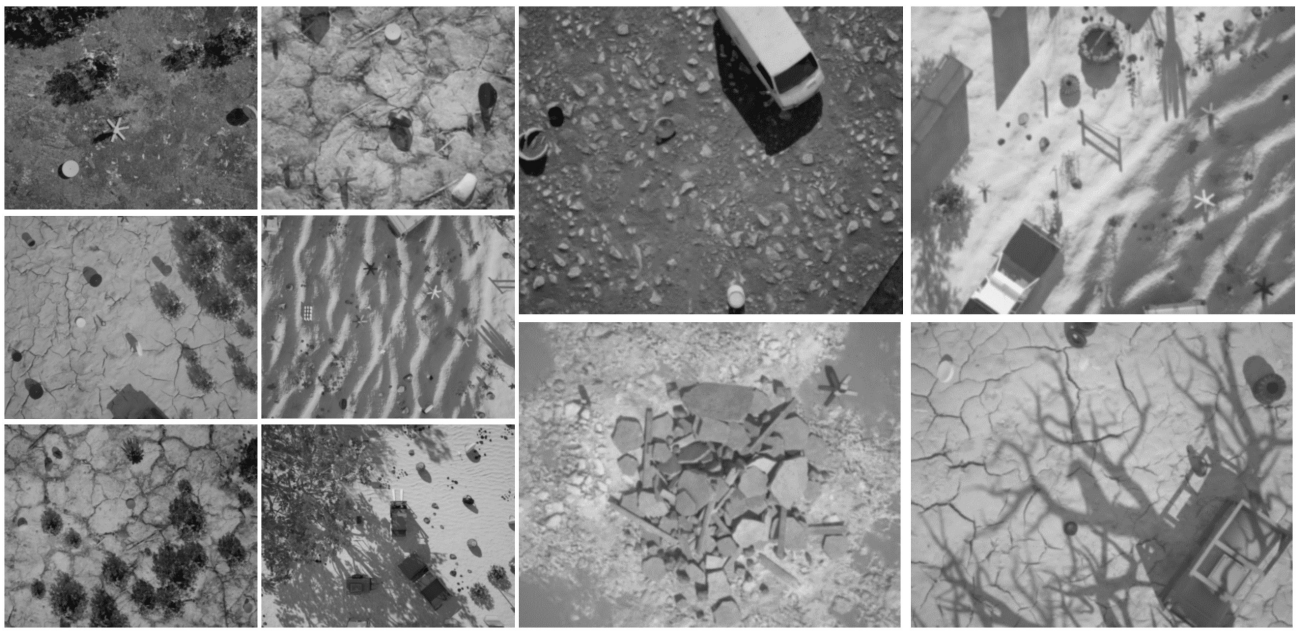


Figure 4: Arbitrarily selected Unreal Engine imagery for some of our modeled environments, whose content (3D models and textures) were authored by others. The imagery is generated from a simulated UAS with a specified RGB camera (e.g., particular f-stop, FOV, etc.) at different altitudes, sensor look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

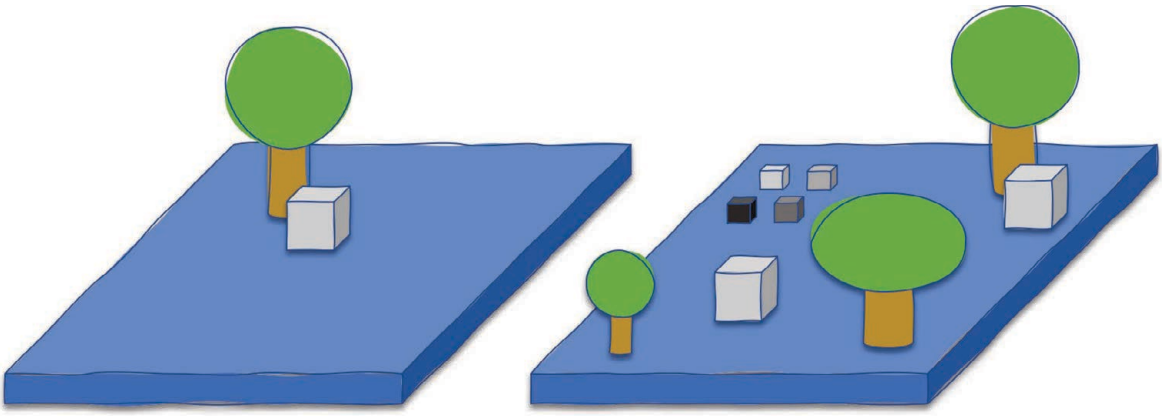


Figure 7: Illustration (left) of a scene with a target (the cube) and environment object (tree). Example (right) with randomly placed and scaled trees and targets.

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.

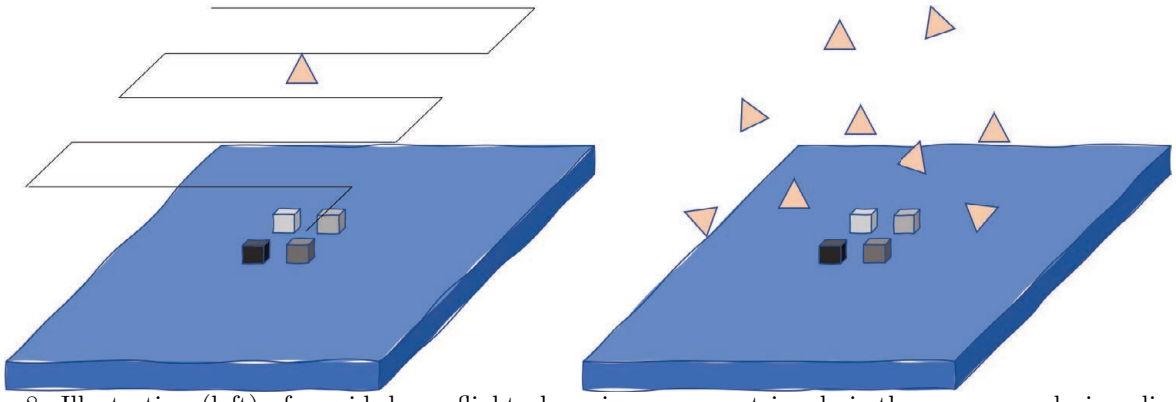


Figure 8: Illustration (left) of a grid drone flight observing a scene; triangle is the camera and view direction and cubes are targets. Example (right) showing randomly selected viewing positions.

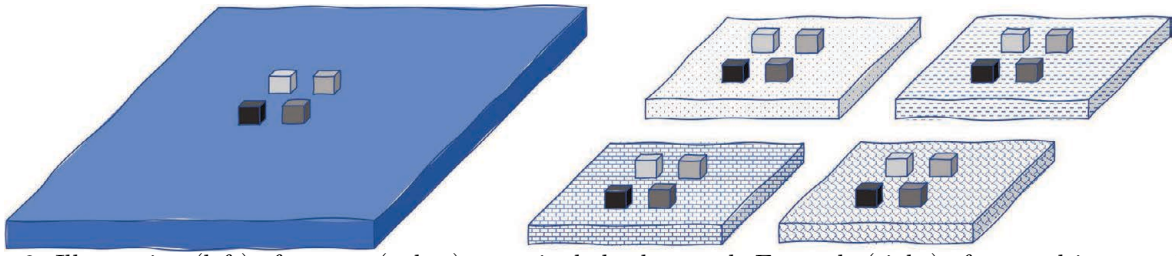


Figure 9: Illustration (left) of targets (cubes) on a single background. Example (right) of same objects on "mini islands" with different textures.

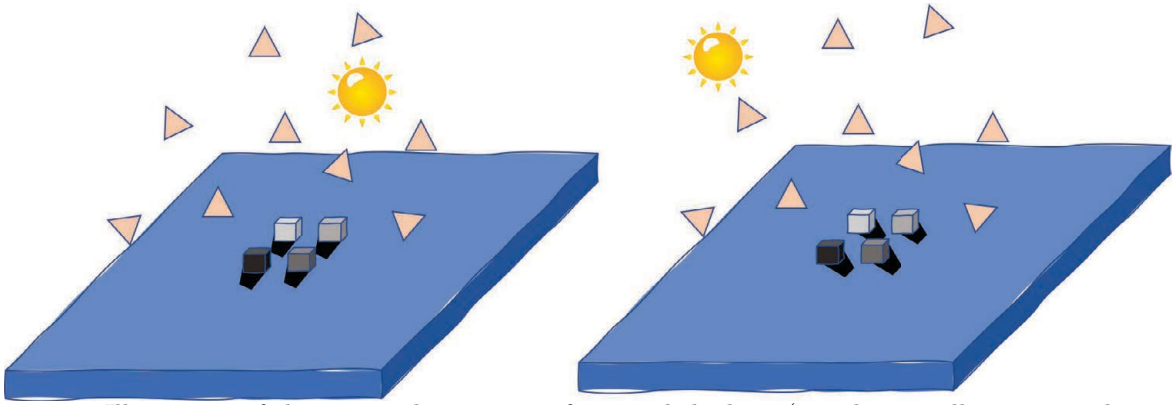


Figure 10: Illustration of changes in the position of sun and shadows (not showing illumination changes).

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

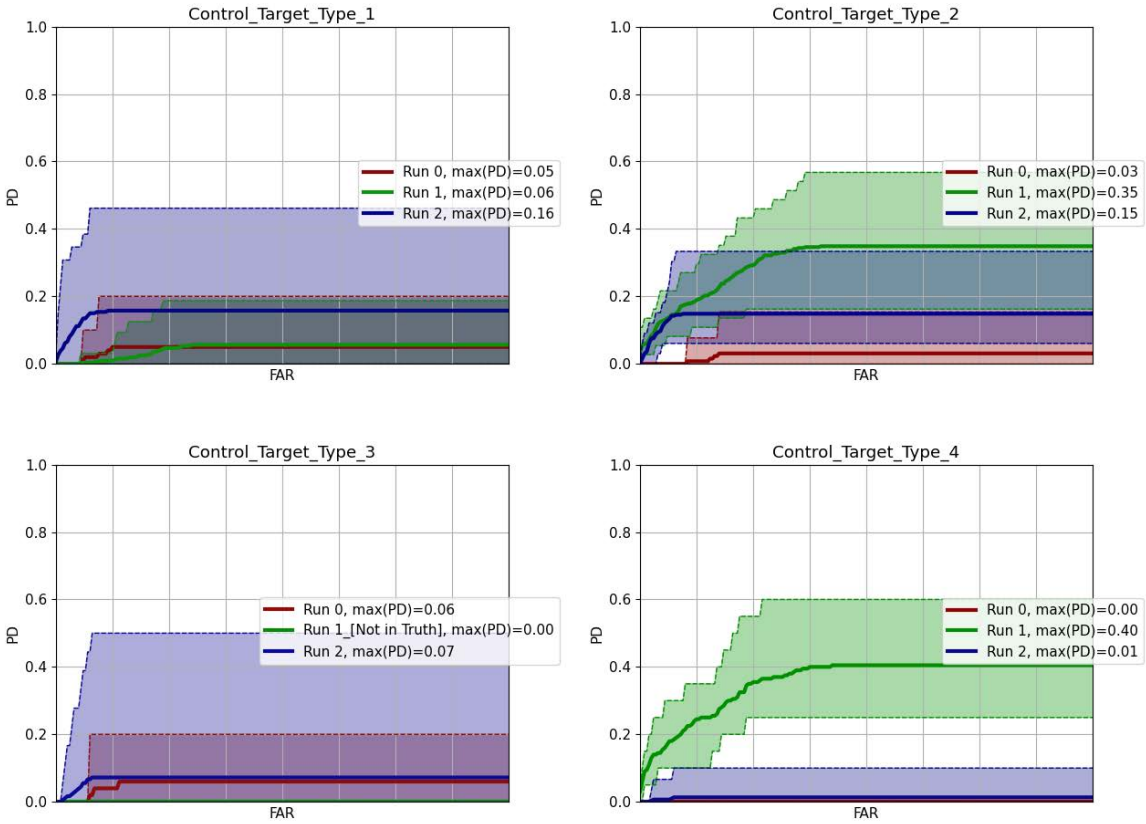


Figure 12: Receiver operator curve for the model trained only on real data.

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-

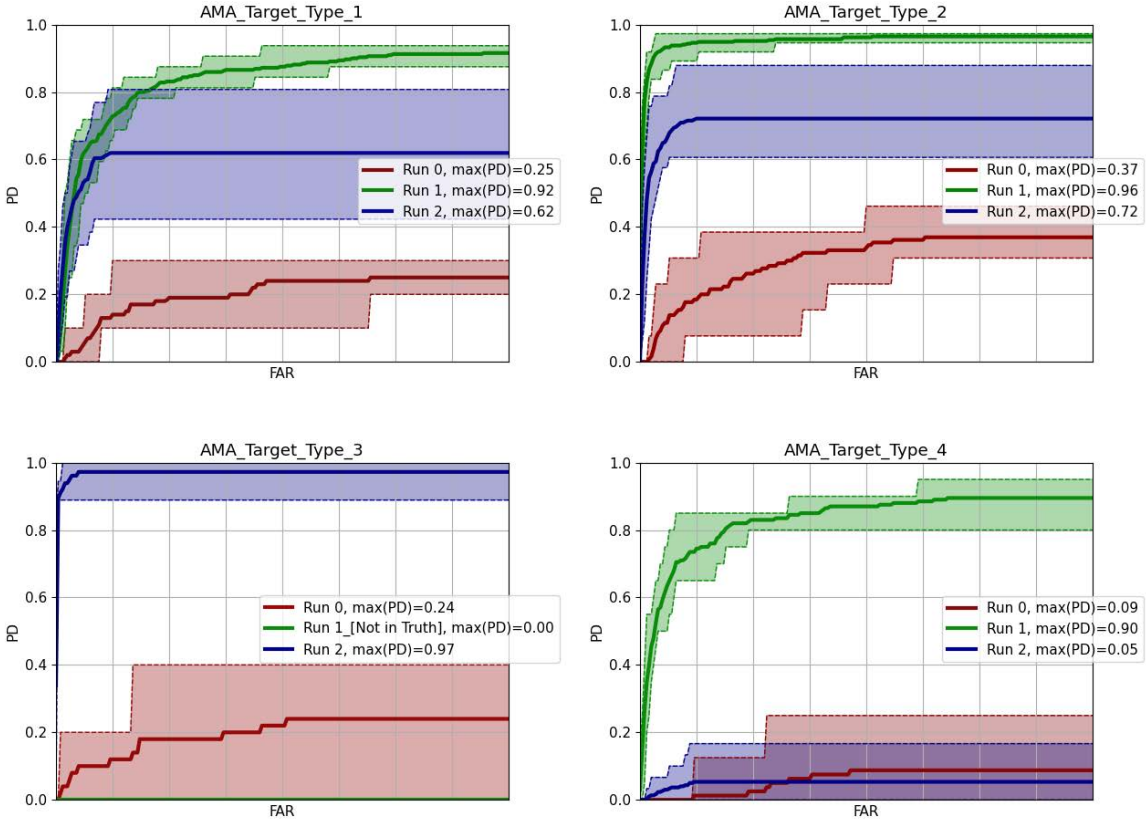


Figure 13: Receiver operator curve for the model trained only on AMA generated images.

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

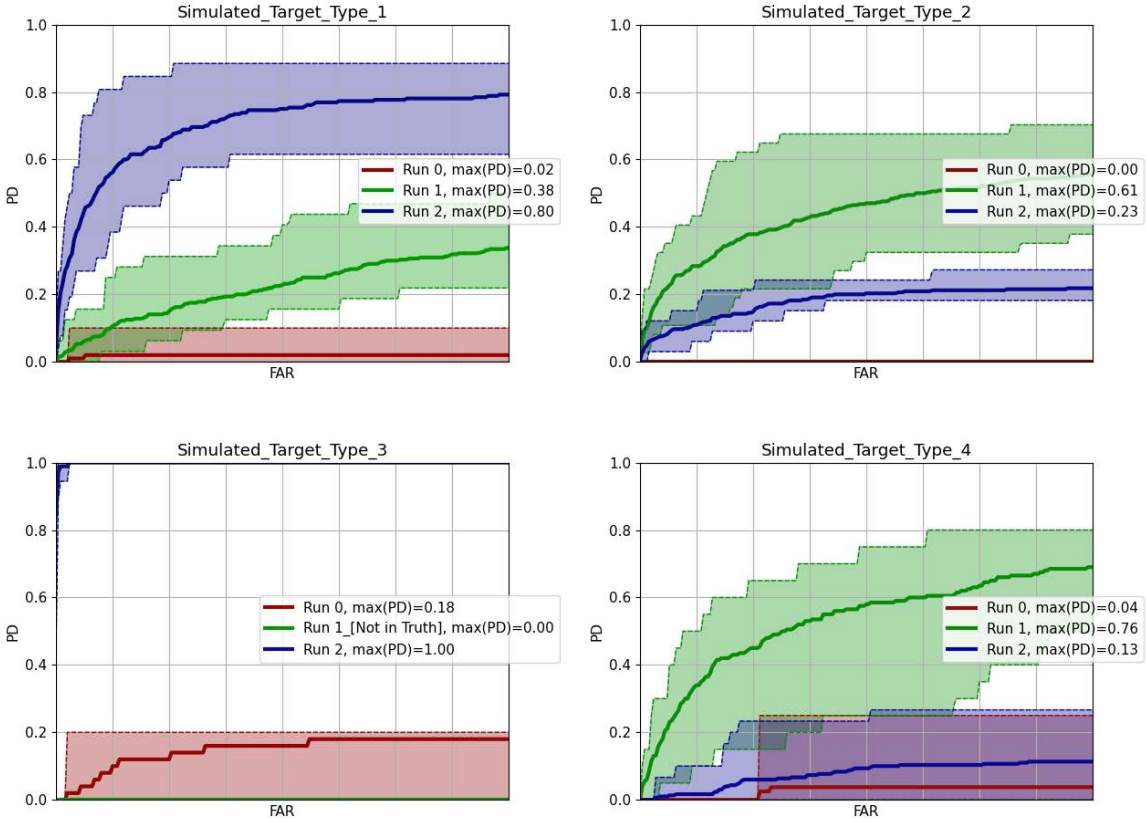


Figure 14: Receiver operator curve for the model trained only on simulated images.

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 autmentation), 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

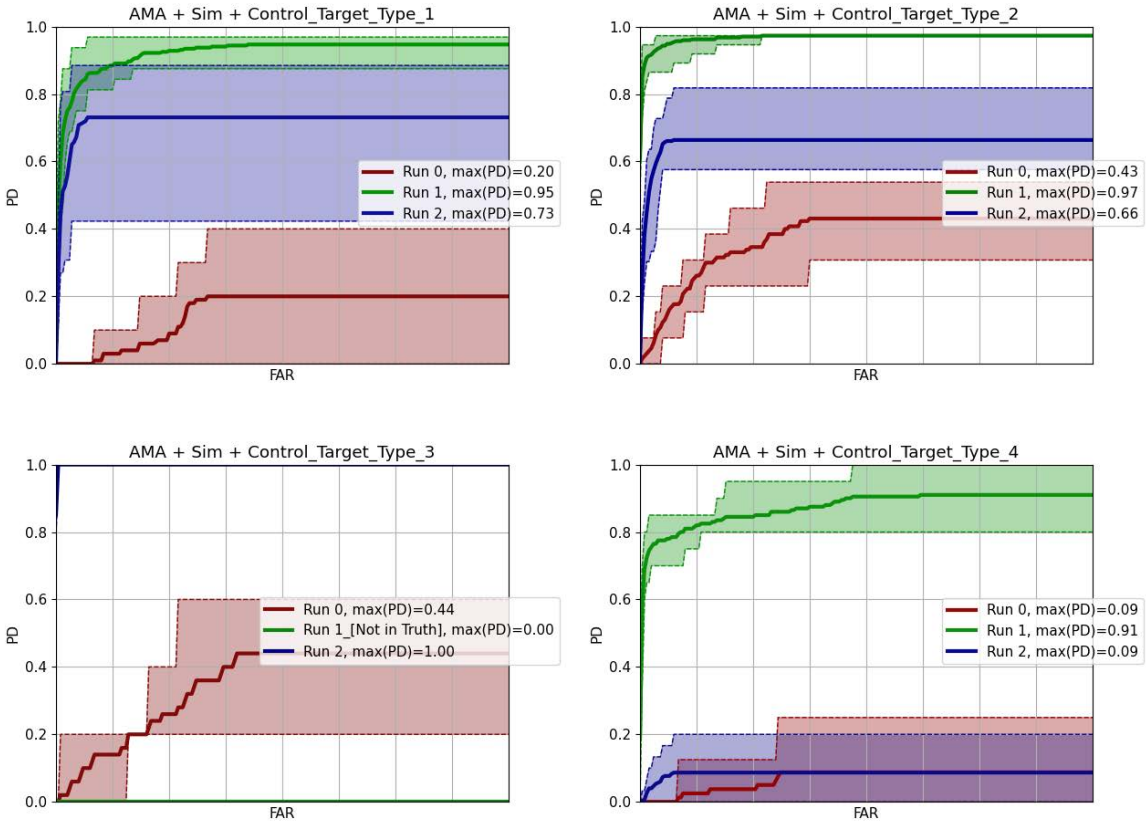


Figure 15: Receiver operator curve for the model trained on a combination of real, simulated and AMA generated images.

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 , L. Y., "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).

Chapter 11

Title: Earth Mover's Distance as a Similarity Measure for Linear Order Statistics and Fuzzy Integrals

Venue: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)

Date Published: July 11, 2021

Earth Mover's Distance as a Similarity Measure for Linear Order Statistics and Fuzzy Integrals

Matthew Deardorff, Derek T. Anderson, Timothy C. Havens, Bryce Murray, Siva K. Kakula, Timothy Wilkin

2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) | 978-1-6654-4407-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/FUZZ45933.2021.9494431

***Abstract*—This paper focuses on a powerful nonlinear aggregation function, the Choquet integral (ChI). Specifically, we focus on situations where the parameters of the ChI are learned from data. For N inputs, the ChI breaks down into $N!$ underlying linear convex sums (LCSs) with 2^N shared variables. Typically, these LCSs are reducible into a drastically smaller number of linear order statistics (LOSs). In the spirit of explainable AI (XAI), our goal is to discover the minimal underlying operator structure of a learned ChI to be conveyed to its users. The challenge is, there does not appear to be widespread research or agreement regarding how to compute similarity within and between measures or integrals. In this paper, we explore the earth mover's distance (EMD), a parametric cross-bin measure, to capture semantic relatedness between LOSs. EMD is used to measure dissimilarity between integrals. In the case of a single ChI, underlying aggregation operator structure is discovered via EMD and clustering. A combination of synthetic and real-world experiments are provided to demonstrate interpretability and reduction of complexity.**

I. Introduction

Aggregation operators, like the *linear order statistic* (LOS)[1], *the ordered weighted average* (OWA)[2], *and the fuzzy integral* (FI), are widely used for tasks like regression and classification in contexts such as multi-criteria decision making and image processing. These operators combine data (aka inputs) relative to knowledge about the utility of the individuals and their interactions. An aspect of using these is, in our modern era of data-driven artificial intelligence (AI) and machine learning (ML), algorithm users are reluctant to deploy a technique if it is opaque and cannot be explained or trusted. Whereas a great deal of effort has gone into understanding the above operators at a fundamental level (e.g., [1]), there is more to be understood relative to their derivation from data. In the current paper, we confront data-driven XAI for fusion by exploring how to measure and use similarity within and between linear convex sums (LCSs), LOSs, and FIs.

The reader can refer to [2] for a recent survey on explainable AI (XAI) and the kinds of questions it allows us to answer, such as why did the algorithm learn what it did, or *how do we explain* to the user what an algorithm is doing. In the past few years, our group has explored XAI for information fusion. Specifically, we have proposed statistical, visual, and linguistic XAI fusion explanations relative to tasks like classification and regression [3–10].

The XAI challenge confronted in the current article is, how do we unearth and communicate a minimal set of underling logic behind a learned instance of the *Choquet integral* (ChI)? This XAI task fits into the category of generating local explanations [2]. For example, consider N sources of information, $X = \{x_1, x_2, ..., x_N\}$, which provide the input $\mathbf{h} = (h_1, h_2, ..., h_N)$. Herein, we let $h_i = h(x_i)$ for sake of equation simplification. It can be shown that the ChI can be decomposed into $N!$ underlying LCSs (one for each input sort) with 2^N variables (number of fuzzy measure (FM) variables). Without loss of generality, consider $N = 3$ inputs, where these inputs have the relation $h_2 > h_1 > h_3$. The output of the ChI for this input sort is the linear function $w_1h_2 + w_2h_1 + w_3h_3$. An important detail here is that the decomposition and resulting equation can be interpreted and explained. Interpretability of the entire process hinges on if the inputs are interpretable. The aspect that is unique to this paper is that in practice most applications do not make use of the full representational capability of the ChI, i.e., use all $N!$ unique LCSs. Usually, the ChI breaks down into a single, or handful of, simple, context-dependent operators like the max (t-conorm), min (t-norm), mean, soft and trimmed variants, or a more unique LOS. The question we aim to answer is, for a given set of data, which operator(s) did the ChI learn? Our goal is to demystify the ChI, helping inform users in their specific application domains about what their ChI is doing. Our current paper addresses this by proposing a similarity measure designed for the ChI, and then by using this to provide discovery and communication of the underlying hidden structure/logic in ChIs that are learned from data.

While we are interested in measuring similarity between any LOS and ChI, it is helpful to restrict our discussion, and at moments only consider, the canonical aggregation operators **max**, **median**, **mean**, and **min**. These operators are different points on the LOS and ChI "spectrum," from the smallest of the inputs to the largest. They represent a tractable set of important commonly encountered operators that we believe outline a set of relationships that help us establish the semantics of similarity in this context. If we cannot satisfy these relationships then it is unlikely that a proposed similarity measure will work on more unique LOSs/ChIs. To this end, we consider the following three touchstones:

(T1) The **max** and **min** operators are least similar to each other and should have a similarity of 0.

Matthew, Bryce, and Derek are with the Electrical Engineering and Computer Science department, University of Missouri, MO, USA, e-mail: msdrm8@mail.missouri.edu. Siva and Timothy Havens are with the College of Computing and the Department of Electrical and Computer Engineering, Michigan Technological University, MI, USA. Tim Wilkin is with the School of Information Technology, Deakin University, Victoria, AU.

[1]The LOS is also referred to commonly as linear functions of order statistics and linear combinations of order statistics.

[2]When the input and weights are real-valued numbers and not fuzzy sets, the OWA is an LOS.

(T2) Similarity between equivalent operators should be 1.
(T3) Similarity between (**max**, **median**) and (**min**, **median**) should be greater than 0 but less than 1, as the **median** is somewhat similar to the other two operators.

These touchstones are based on our intuitive understanding of what a partial ordering of LOSs and ChIs should look like, as we could not find any literature proposing an ordering structure across all LOSs or ChIs.

The remainder of this article is organized as follows. First, we succinctly review the LOS, Euclidean distance, and the EMD. Next, we review the ChI, the EMD is extended to measure similarity between ChIs, and clustering is proposed to discover underlying similarity structure within an integral that can be easily communicated to people. Lastly, synthetic and real-world experiments are provided to show the effectiveness of the proposed ideas.

II. Linear Order Statistic

Let X be a set of N sources of information, e.g., from humans, algorithms, or sensors. Furthermore, let h_i be the input from source i, e.g., a subjective belief, objective sensor measurement, classifier output, etc. An LCS is defined as

$$f_{LCS}(\mathbf{w}, \mathbf{h}) = \sum_{i=1}^{N} w_i h_i, \tag{1}$$

where $\mathbf{w} = (w_1, ..., w_N)^t \geq \mathbf{0}^N$ and $\left(\sum_{i=1}^{N} w_i\right) = 1$.

The LOS is

$$f_{LOS}(\mathbf{w}, \mathbf{h}) = \sum_{i=1}^{N} w_i h_{\pi(i)}, \tag{2}$$

where π is a sorting such that $h_{\pi(1)} \geq h_{\pi(2)} ... \geq h_{\pi(N)}$. Thus, a LOS is a LCS with a "pre-sort" where $\mathbf{w} = (w_1, ..., w_N)^t \geq \mathbf{0}^N$ and $\left(\sum_{i=1}^{N} w_i\right) = 1$.

A. ℓ_p-Norm as an LOS Proximity Measure

The frequently used ℓ_p-norm is a rational place to start when it comes to capturing similarity between pairs of LCSs or LOSs. In this section, we show that while this norm can be useful as a baseline, it has semantic issues that leave us desiring a superior measure.

The ℓ_p-norm d between two vectors (which herein represent LOS coefficients) $\mathbf{w}_j$ and $\mathbf{w}_k$, where $\mathbf{w}_j, \mathbf{w}_k \in \mathcal{R}^N$, is

$$d(\mathbf{w}_j, \mathbf{w}_k) = \left(\sum_{i=1}^{N} (w_{ji} - w_{ki})^p\right)^{\frac{1}{p}}, \tag{3}$$

which can be converted into a similarity[3] (if desired) via

$$s(\mathbf{w}_j, \mathbf{w}_k) = \frac{\rho - d(\mathbf{w}_j, \mathbf{w}_k)}{\rho}, \tag{4}$$

where ρ is the maximum allowable distance.[4] Second, the ℓ_p-norm operates on a per-bin basis, meaning only $\mathbf{w}_i^1$ and $\mathbf{w}_i^2$ are compared, with no interaction considered between $\mathbf{w}_i^1$ and $\mathbf{w}_j^2$, where $i \neq j$. This lack of cross-bin interaction is often acceptable for comparing LCSs if there is no implicit relation between bins. However, this is not the case for LOSs due to their sort, meaning information is lost in this comparison.

Consider our touchstones mentioned in Section I (T1-T3); the question is, which of these do the ℓ_p-norm satisfy? Without loss of generality, we use $N = 3$ in the following to demonstrate compactly our points.

(T1) The max and min operators **max** $= (1, 0, 0)$ and **min** $= (0, 0, 1)$, result in $s($**max**, **min**$) = 0$ meaning these are as dissimilar as possible. However, note that we get a maximal distance in many cases where there is no overlap in the non-zero coefficients.
(T2) The similarity s(**max**,**max**) is 1, meaning that the ℓ_p proximity measure has a satisfying upper bound.
(T3) The similarity s(**max**,**median**) and s(**min**,**median**) are both 0, which is not what we would expect. This means that the ℓ_p based similarity does not enforce the same ordering we relate to the max, median, and min. This is the core issue that leads us to finding a superior measure.

In summation, the ℓ_p-norm behaves favorably when the LOS weight vectors overlap. However, it falls short of many semantic expectations due to the fact it ignores the interactions across bins, a scenario we remedy in the next section.

B. Earth Mover's Distance on LOSs

In this section, we evaluate the EMD to remedy shortcomings identified in Section II. The EMD is a measure of divergence between two distributions.[5] It is based on a solution to the well-known transportation problem, i.e., the Monge-Kantorovich problem. In [11], Rubner introduced the EMD in the context of *content based image retrieval* (CBIR) for unequal mass distributions. In [12], Levina and Bickel proved that the EMD, a parametric measure, is equivalent to the Mallows and Wasserstein distance for the case of two probability distributions and it is different when applied to unnormalized distributions of different masses (e.g., *signatures* in CBIR). We use the EMD herein as it enables cross-bin interactions during dissimilarity, a concept not possible with an ℓ_p-norm, Jaccard or Dice measures, etc.

Let $\mathbf{h}$ be a (one-dimensional) histogram of length L_1, $h_i \in \Re+$, $1 \leq i \leq L_1$; and let $\mathbf{b}$ be a second histogram of length L_2, where $b_i \in \Re^+$. $EMD(\mathbf{h}, \mathbf{b})$ is the EMD between $\mathbf{h}$ and $\mathbf{b}$. The goal is to find a flow $F = [f_{ij}]$, where f_{ij} is the *flow* between h_i and b_j, which minimizes

$$WORK(\mathbf{h}, \mathbf{b}, F) = \sum_{i=1}^{L_1} \sum_{j=1}^{L_2} d_{ij} f_{ij}, \tag{5}$$

[3]It is important to note that not all dissimilarity (or similarities) can be converted to their dual on such a simple premise.

[4]For example, $\rho = \sqrt{2}$ for the ℓ_2-norm, $w_i \geq 0$, and $(\sum_{i=1}^{N} w_i) = 1$.

[5]We refer to these entities as histograms hereafter versus distributions or signatures or etc. Typically, the nomenclature depends on the application and/or properties, e.g., probabilistic for positivity and sum to one.

subject to

$$f_{ij}, \qquad 1 \leq i \leq L_1, 1 \leq j \leq L_2, \tag{6a}$$

$$\sum_{j=1}^{L_2} f_{ij}, \leq h_i \qquad 1 \leq i \leq L_1, \tag{6b}$$

$$\sum_{i=1}^{L_1} f_{ij}, \leq b_j \qquad 1 \leq i \leq L_1, \tag{6c}$$

$$\sum_{i=1}^{L_1}\sum_{j=1}^{L_2} f_{ij} = \min\left(\sum_{i=1}^{L_2} h_i, \sum_{j=1}^{L_1} b_j\right), \tag{6d}$$

where $D = [d_{ij}]$ is called the *ground distance*. Once the transportation problem is solved—i.e., the optimal F^* is found—the resulting EMD is

$$EMD(h, g) = \frac{\sum_{i=1}^{L_1}\sum_{j=1}^{L_2} f_{ij}^* d_{ij}}{\sum_{i=1}^{L_1}\sum_{j=1}^{L_2} f_{ij}^*}. \tag{7}$$

Herein, d_{ij} is the distance between "bins" (indices), e.g., $d_{11} = 0$, $d_{12} = 1$, $d_{13} = 2$, $d_{1L} = L-1$, etc. There is no cost to stay in the same bin, which increases by one per bin thereafter. The EMD can be converted into a similarity via

$$s(\mathbf{w}_j, \mathbf{w}_k) = \frac{\rho - EMD(\mathbf{w}_j, \mathbf{w}_k)}{\rho}, \tag{8}$$

where the maximal allowable distance $\rho = L - 1$.

The primary benefit of using the EMD is that it allows us to define a ground distance matrix in a way that mirrors the sort induced by LOSs. The ground distance matrix asserts that bins that are adjacent to each other are "closer" than bins that are non-adjacent, the same way values that are similar are sorted to be near each other. As a result, this ground distance matrix changes the distance topology in a way that makes the EMD satisfy each of our touchstones, which we now describe.

(T1) $EMD(\textbf{max}, \textbf{min})$ is the scenario in which the entire "mass" has to be moved across the largest possible number of bins. With the constraint that the sum of weights is 1, we are moving the largest possible mass the farthest possible distance; hence, the EMD is maximized and the similarity is 0.

(T2) $EMD(\textbf{max}, \textbf{max}) = 0$ as there is no change in the coefficients, no work is required. The corresponding similarity is 1, resulting in the upper bound.

(T3) $EMD(\textbf{max}, \textbf{median}) = 0.5, EMD(\textbf{min}, \textbf{median}) = 0.5$, as these cases result in the entire mass of their respective histograms being shifted half the maximum possible distance.

Based on the above criteria (extreme bounds), which could clearly be expanded to include other desirable properties like monotonicity, idempotency, etc., the EMD is a more suitable distance measure for LOSs than the ℓ_p-norm, or other bin-to-bin measures at that. What is the disadvantage of using it? First, the EMD is undefined for distributions with negative values. While LOSs are usually constrained to have non-negative values, they do pop up in cases of regression such as in [13]. Additionally, the calculation of the EMD is computationally more expensive than an ℓ_p-norm, with [14] running in $O(n^2)$, though approximation algorithms such as [15] can run in linear time.

The difference between the EMD and an ℓ_p-norm can be visualized by calculating the distance between a point and the set of possible LOSs that exist in 3-space, shown in Figure 1. If we treat each axis as a different weight, we obtain a triangular plane that describes the set of possible LOSs with weights that sum to one. If we color-map this plane based on the distance from specific points, we can see how the topology changes based on the distance measure. The plot in (b) shows how under an ℓ_p each extreme operator is equidistant from the **mean**. This is counter-intuitive as we more closely associate the **median** with the **mean** rather than the **max** or **min**. In the EMD plot, we can see that this is correctly modeled as the triangle vertex corresponding to the **median** is closer than the other two vertices. We can see something similar in plots (c) and (d) where the **max** operator appear equidistant from the other two vertices under an ℓ_p norm, but not the EMD.

III. EMD Between ChIs

In this section, the EMD is extended to integrals via the idea of decomposing the ChI into its underlying set of LCS and corresponding LOSs. To the best of our knowledge, the only prior work on distances between FMs is based on the Hellinger distance [16, 17]. In 1909 Ernst Hellinger introduced a distance measure for two probability distributions (additive measures). In 2013, [16] Torra et al. explored its extension to FMs. While the authors introduce a variation of the Hellinger distance for non-additive measures, which relies on a Radon-Nikodym-type derivative for the FM, ultimately their distance measure works on a bin-to-bin basis and therefore it does not satisfy the semantic touchstones considered herein.

A. Choquet Integral

The FM, $g : 2^X \to R^+$, is a function with two properties[6]: (i) (boundary condition) $g(\emptyset) = 0$, and (ii) (monotonicity) if $A, B \subseteq X$, and $A \subseteq B$, then $g(A) \leq g(B)$. The ChI is

$$\int \mathbf{h} \circ g = C_g(\mathbf{h}) = \sum_{j=1}^{N} h_{\pi(j)}(g(A_{\pi(j)}) - g(A_{\pi(j-1)})), \tag{9}$$

for $A_{\pi(j)} = \{x_{\pi(1)}, \ldots, x_{\pi(j)}\}$, $g(A_{\pi(0)}) = 0$, and π (sort).

B. EMD on a Decomposed ChI

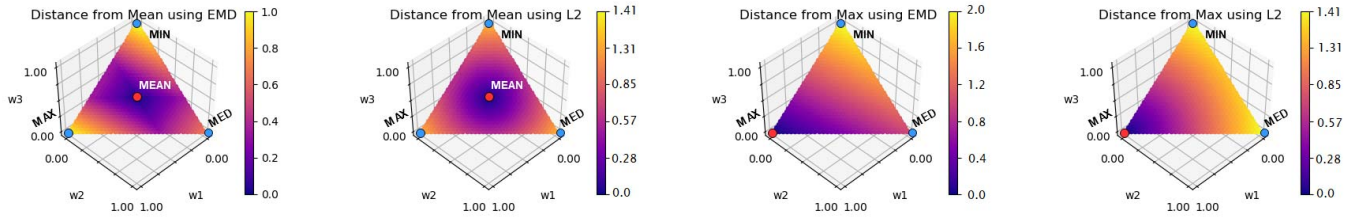
As already stated, a ChI on N variables has $N!$ underlying unique[7] LCSs (one for each possible sort, π). It is important to note that this sort is consistent across integrals. As such, distance can be measured and aggregated across the partitioned input space,

$$EMD_c^1(g_1, g_2) = \frac{1}{N!}\sum_{i=1}^{N!} EMD((g_1)_{\pi_i}, (g_2)_{\pi_i}), \tag{10}$$

where g_1, and g_2 are FMs, and $(g_k)_{\pi_i}$ is the ith sort of g (aka LCS). (10) is superscripted with a 1 to differentiate it from the

[6]For finite X, there is a third condition for continuous domains.

[7]We say unique with respect to the $N!$ sorts. LCSs are often duplicated (aka have the same weights) across sorts.



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

Fig. 1: Visualizing the set of possible LOSs as a plane can help us understand what impact the EMD has on distance topology. In these graphs, each axis represents a variable and the color of a pixel on the plane represents the distance from a specified point to that location. In (b) and (d) we see that an ℓ_2 norm considers each axis to be equally distant from the others, where as in (a) and (c) it is more expensive to move from weight 1 to weight 3 than weight 1 to weight 2.

data-derived variant detailed in (12). We elected to normalize the distance between all LCSs by the number of walks so the resulting value can be interpreted as an average distance between any two analogous walks. In summary, the idea is to produce an exhaustive and non-intersecting partitioning of the input (and thus operator) space and to measure the average distance across all decomposed LCSs, with respect to already noted the EMD benefits.

Equation (10) is expressed in exhaustive—aka all $N!$ sorts—form. When both integrals have the same *underlying LOS structure*[8], (10) can be expressed as

$$EMD_c^1(g_1, g_2) = \sum_{i=1}^{M} \alpha_i EMD(\mathbf{w}_i^1, \mathbf{w}_i^2), \tag{11}$$

where M is the number of LOSs, $\mathbf{w}_i^k$ are the LOS weights for the kth FM, and α_i is the number of sorts in LOS i divided by $N!$. Otherwise, the EMD can be expressed as a combination of (10) and (11), where the $[0, 1]$ weights are $\frac{1}{N!}$ for individual LCS terms and the relative frequency terms outlined above for LOSs. The point is, the EMD between two ChIs can be expressed as an aggregation of the individual LCSs or its respective underlying LOSs.

C. EMD on Data-Derived Choquet Integrals

The ChI is often learned from data, e.g., [5, 6, 18–22], meaning we have additional information to aid in comparison. In [4], we showed a way to measure the frequency of observations per walk/context. A limitation of (10) is that it does not take data observably into account, which can lead to comparing walks that were not learned or have little data support. The following remedies this by weighting each walk,

$$EMD_c^2(g_1, g_2) = \sum_{i=1}^{|C|} \beta_i EMD((g_1)_{\pi_i}, (g_2)_{\pi_i}), \tag{12}$$

where $\beta_i = \frac{1}{2}\left(f([g^1]_{\pi_i}) + f([g^2]_{\pi_i})\right)$, C is the intersecting set of walks between the two datasets used to learn FMs g^1 and g^2, and f is the relative frequency of a walk with respect to our reduced scope. We choose to limit our region of interest to only walks that are observable in both sets of data. There are two reasons for this. First, as was shown in [3], variables in unobserved walks are imputed by whatever solving method was utilized, often defaulting to the initialization value or floor or ceiling imposed upon the variable by the monotonicity restraints. These values are often close enough for application uses, but we argue there is nothing truly to be learned about the ChI itself from these data-unsupported variables. Second, portions of ChIs are incomparable if one or both of them are not intended to operate on those specific sorts or partitions.

These frequency values act as a normalizing mechanism, where partitions of the data space that are more common in data—and thus, we argue, more important to the operator—are proportionately weighted. We treat these coefficients additively and rationalize that with the following three examples. If the f values for both $[g^1]_{\pi_i}$ and $[g^2]_{\pi_i}$ are near zero, the sort is very infrequent in either dataset, and therefore it matters less in the overall comparison of FMs. If both f values are high, then the sort in question is very common for both sets of data, meaning the difference between those LOSs is relatively more important. Finally, if one f value is high and the other is low, that sort is important in at least one of the data sets, meaning it should have some influence on the resultant distance. The inclusion of relative frequencies also takes care of the normalization, meaning we do not need to divide by *N!* as in Eq. (10).

IV. EMD on a Single ChI: Structure Discovery

In this section, a second payoff of researching aggregation operator similarity is outlined. We show how measuring proximity between LCSs and LOSs can help us discover *underlying structure* in a single integral. This is useful as it relates to answering the XAI question, "what *logic* was learned?" As N gets larger, reporting $N!$ LCSs is intractable. Instead, it is more effective to summarize the $N!$ individual logics. Specifically, we are interested in a minimal, but still comprehensive, set of operators that capture the majority of aggregation information. See Fig. 2 for a flow diagram of the proposed idea and Algorithm 8 provides a formal description.

Herein, we use the *improved visual assessment of cluster tendency* (iVAT) algorithm [23] to highlight and recommend potential cluster structure in the LCSs. In the iVAT algorithm,

[8]Same structure here means that both integrals have the same number of underlying LOSs bound to the same sort sets.

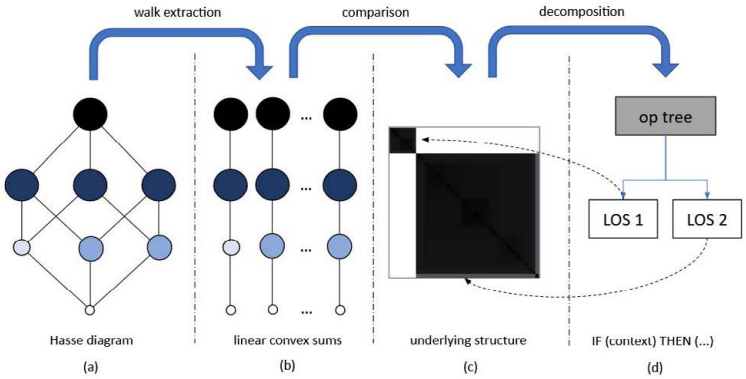


Fig. 2: The proposed decomposition process starts with (a) a FM, which is (b) 'unrolled' into a set of LCSs (b). Next, these LCSs are compared against each other to (c) produce an iVAT image for structure discovery. Lastly, (d) clusters can be manually or automatically extracted into a smaller set of aggregation operators with additional context, e.g., LOSs.

Algorithm 1: Discovery of underlying LOSs in a ChI

Data: g - input FM
for i *to* $N!$ **do**
  for j *to* $N!$ **do**
    $D(i,j)$ = EMD(g_i,g_j)
D = IVAT(D)
if *automatic clustering* == *True* **then**
  return CLODD(D)
else
  return D

similar data patterns generally appear as "dark rectangular blocks" along the Image (matrix) diagonal of the iVAT image. Once an iVAT image is obtained, clusters can automatically be extracted using an algorithm like CLODD [24], if desired. We leave it to the reader to determine if a *human is in the loop* (HITL) or if an automatic procedure is needed.

V. Color-coded XAI Visualizations

To further our goal of interpretability of the ChI, we devised a method to color the Hasse diagram in a way that helps the reader understand which subsets and walks up the diagram correspond to which iVAT/CLODD clusters. To this end, the walks which belong to each cluster are enumerated and all nodes that are a part of that walk receive a tally for contributing to that cluster — and by extension LOS. Each node in the diagram is then drawn as a pie chart, where the previous tallies can be used to show the proportions of how often each node is used by each LOS.

In summary, our iVAT and CLODD color-coded visualizations inform us about operator substructure and percentage of LCSs in each cluster. As a companion to the iVAT visualization, the color-coded Hasse diagram informs us about how each of the identified LCSs are associated with FM nodes and the interplay of the FM and ChI across all possible sorts.

VI. Experiments

In this section, we demonstrate the proposed techniques on two synthetic examples and a real-world example from the benchmark AID remote sensing dataset [25]. The synthetic cases are designed to test controlled scenarios where we know the answer and wish to demonstrate and validate the proposed approaches. The real-world experiment is a case study where we do not have the answer and no analytical solution exists.

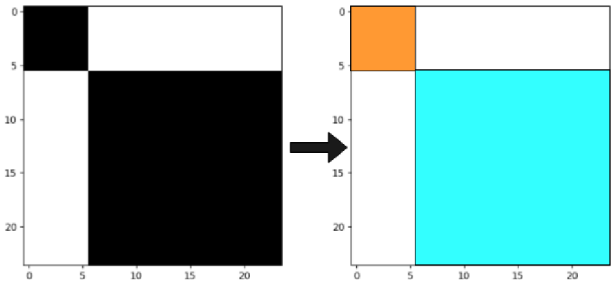
A. Synthetic Experiments

Experiments 1 and 2 are based on binary FMs, meaning $g(A) \in \{0,1\}, \forall A \in 2^X$.[9] Though synthetic, many real-world data fusion problems can be best solved, or closely approximated via binary FMs/ChIs; see [27–29] for multi-sensor and multi-algorithm data fusion examples in hyperspectral image processing and remote sensing.

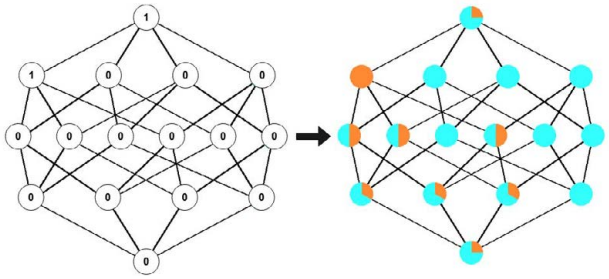
In Experiment 1, shown in Figure 3, we show the decomposition and discovery process on an FM such that $g(A) = 0, \forall A \in 2^X$, except $g(\{x_1, x_2, x_3\}) = 1$. Therefore, there are only two underlying LOSs, $\mathbf{w}^1 = (0,0,0,1)^t$ and $\mathbf{w}^2 = (0,0,1,0)^t$. In Experiment 1 we focus on setting a value at the top of the Hasse diagram to 1. In Experiment 2 (see Figure 4) we set a single value low in the Hasse to 1 (namely $g(\{x_2\}) = 1$). Due to the monotonicity constraints on the FM, the FM value for all subsets that contain source $\{x_2\}$ is therefore 1, and 0 otherwise. This results in 4 clearly separable LOSs; $\mathbf{w}^1 = (0,0,1,0)^t$, $\mathbf{w}^2 = (0,1,0,0)^t$, $\mathbf{w}^3 = (1,0,0,0)^t$, and $\mathbf{w}^4 = (0,0,0,1)^t$. It is interesting to note that this 'max-like' operator appears to be more complex than the above 'min-like' operator, despite their constructions being similar.

Figs. 3 and 4 show that despite there being 24 (4!) walks up the lattice for $N = 4$, iVAT and CLODD clearly highlight that there are two and four unique underlying LOSs respectively. In these figures, view (a) shows the iVAT result of the 24 underlying LCSs and its color-coded CLODD clustering result. The images in (b) are a new visualization technique proposed herein. The challenge is that the images shown in (a) help us understand underlying cluster structure. However, context is

[9] In [26] we proved that a binary ChI is a Sugeno integral.

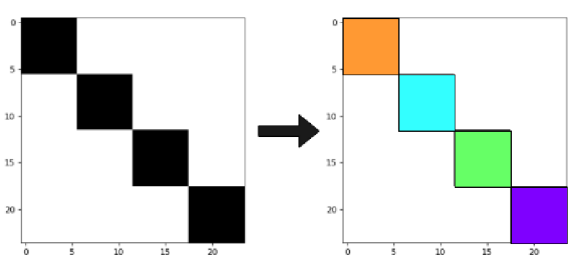


(a) (left) iVAT image and (right) color coded CLODD results.

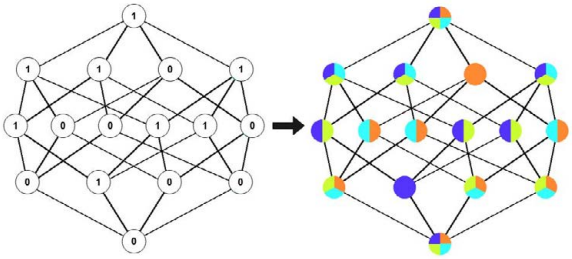


(b) (left) Hasse diagram with measure values and corresponding (right) color coded pie chart diagram.

Fig. 3: iVAT, CLODD, and color coded Hasse diagram for the synthetic Experiment 1.



(a) (left) iVAT image and (right) color coded CLODD results.



(b) (left) Hasse diagram with measure values and corresponding (right) color coded pie chart diagram.

Fig. 4: iVAT, CLODD, and color coded Hasse diagram for the synthetic Experiment 2.

not preserved. In view (b), the left image is the Hasse diagram, where $g(\emptyset = 0)$ is on the bottom of the diagram, and $g(X) = 1$ set is on the top, and the monotonicity constraints are shown as edges. Each layer corresponds to a k-tuple, e.g., the layer above the empty set are the singletons, $\{g_1, g_2, g_3, g_4\}$.

The color-coded Hasse diagram corresponding to these FMs allow the reader to see at a glance which walks make up which LOS. For example, the top and bottom layers of the color-coded Hasse diagram in 3 reveal that approximately a quarter of the walks in this FM result in the orange LOS (0,0,1,0), while the rest result in the teal LOS (0,0,0,1). Additionally, due to the fact that the node corresponding to the singleton $\{g_4\}$ is solid teal, if that node is visited during a walk the resultant LOS is guaranteed to be the teal LOS.

B. Real-World Experiment

Unlike Section VI-A, the FMs in Experiment 3 are learned from real-world remote sensing data. In [4, 6, 30–32], we used the ChI to fuse a set of heterogeneous architecture deep convolutional neural networks (DCNNs) for object detection and land classification in remote sensing. The motivation of that work was that no single deep learning architecture nor trained model had been shown to be superior across all data sets and classification tasks. As such, the goal was to exploit the individual advantages of the networks and use the set to overcome their individual weaknesses to obtain a more accurate and reliable solution.[10]

[10]We showed in [6] that the ChI can be represented and trained as a neural net. Thus, the fusion of a set of heterogeneous nets is, therefore, merely a larger net with the benefit of explainability versus opaqueness [5].

While our previous publications focused on algorithm development and establishing the quantitative performance benefits of our ChI for fusing deep networks, we also explored different benchmark datasets, types, and numbers of architectures (see [4, 6, 30–32]). Herein, we restrict our analysis to the single case of fusing four DCNNs on the AID remote sensing dataset [25]. The goal here is to qualitatively explore the proposed XAI tools. The AID dataset contains $10,000$ images of 30 different aerial scene types. As outlined in [4, 6, 30–32], we trained both a "shared ChI", i.e., one FM shared across all 30 classes, and also one ChI per class. The benefit of the prior is that more data are available for training each class, whereas the latter has the benefit of learning class-specific fusion, generally at the cost of training data sparsity. We performed 5-fold cross validation with respect to neural learning and 2-fold CV for fusion. Rather than show all of the resulting 300 FMs, we summarize the findings next.

Overall, using the methods discussed herein, we discovered that nearly all of the learned ChIs were associated with a single operator, the minimum or a trimmed-minimum. This led us to further study our network outputs and we discovered that the deep networks are strong learners.[11] That is, the DCNNs were almost always certain—i.e., output values near 0 or 1—and they were frequently in agreement. The networks have not learned, nor where they informed to during learning, how to express uncertainty. They were trained to either output a 0 (not that class) or 1 (is a class). As such, it is logical to expect that the ChI learned to take a pessimistic stance, aka listen to the

[11]A scenario not the particularly ideal for fusion

lowest confidence across the networks. In return, knowing this informs us that we should revisit the learning paradigm and, perhaps, take an ensemble of weak learners approach.

Fig. 5 shows the result of a typical experiment where the fusion of classifiers led to an increase in classification accuracy and algorithm robustness [4, 6, 30–32]. In general, much like iVAT on data for clustering, non-binary FMs result in less clearly separated and trivial LOS groupings. The distances shown in Figure 5(a) are normalized with respect to the min and max observed EMD distances. If the matrix was instead normalized with respect to the theoretical maximum possible EMD distance then we might be led to believe that there is instead a single cluster, a (trimmed) minimum. However, when normalized between min and max observed EMD value we see somewhere between two to five or perhaps seven clusters. The point is, it is up to the human visualizing these results, the set of CLODD parameters in the case of automatic cluster extraction, or some user-specified threshold governing approximation error between using the entire ChI and the iVAT/CLODD discovered number of LOSs. The focus of this paper is to introduce the tools and raise awareness of such questions. In future work, we will explore if there are answers to this question or if there are different truths for different contexts and applications.

VII. Conclusion and Future Work

In summary, the aim of this paper is to explore the role of the EMD as a measure to aid similarity analysis between and within FMs and ChI. We discovered that the bin-to-bin and ground matrix benefits of EMD improved our ability to recover semantic expectations of aggregation operator ranking. Furthermore, we showed how to compute distance between ChIs and considered how to take sampling frequency into account. We also presented a way to apply the measure within a single ChI, facilitating underlying operator discovery. These methods were illustrated via a combination of synthetic examples and a real-world dataset.

In future work, we will explore how to better express and study ground matrix selection and EMD in general relative to semantic ranking of a wider set of operators beyond the extreme cases of the max, median, mean, and min. We also plan to use the proposed measure on a set of learned ChIs across cross validations to understand if our remote sensing deep learner fusion is learning different logics. Other planned work involves looking at our most recent articles on visualization of the ChI [8, 9]. Namely, we have other ways to show data- and ChI-relevant information and information theoretic indices that likely would be beneficial to fold into the illustrations shown here. As already discussed, it is not clear, much like iVAT, CLODD, and clustering in general, what the underlying structure answer should be. We will further explore this topic, most likely in the context of a specific goal or application to aid the specification process. Lastly, one of the major thrusts of the propsed article is XAI. We will look to combine the low-level methods proposed here into more useful high-level explanations for an end user.

References

[1] Y. Narukawa and T. Murofushi, *Choquet integral and Sugeno integral as aggregation functions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 27–39. [Online]. Available: https://doi.org/10.1007/978-3-540-36519-8-3

[2] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," 2019.

[3] B. Murray, M. A. Islam, A. Pinar, D. Anderson, G. Scott, T. Havens, F. Petry, and P. Elmore, "Transfer learning for the choquet integral," 06 2019, pp. 1–6.

[4] B. Murray, M. A. Islam, A. J. Pinar, T. C. Havens, D. T. Anderson, and G. Scott, "Explainable ai for understanding decisions and data-driven optimization of the choquet integral," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2018, pp. 1–8.

[5] B. J. Murray, M. A. Islam, A. J. Pinar, D. T. Anderson, G. J. Scott, T. C. Havens, and J. M. Keller, "Explainable ai for the choquet integral," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–10, 2020.

[6] M. Islam, D. T. Anderson, A. J. Pinar, T. C. Havens, G. Scott, and J. M. Keller, "Enabling explainable fusion in deep learning with fuzzy integral neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 7, pp. 1291–1300, 2020.

[7] B. J. Murray, D. T. Anderson, T. C. Havens, T. Wilkin, and A. Wilbik, "Information fusion-2-text: Explainable aggregation via linguistic protoforms," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, M.-J. Lesot, S. Vieira, M. Z. Reformat, J. P. Carvalho, A. Wilbik, B. Bouchon-Meunier, and R. R. Yager, Eds. Cham: Springer International Publishing, 2020, pp. 114–127.

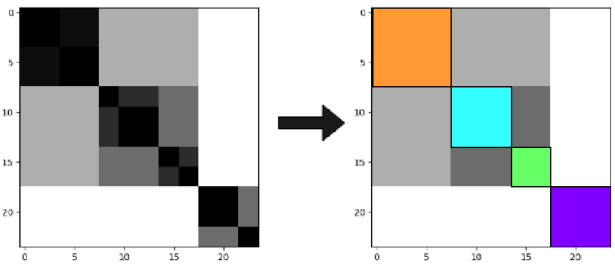
[8] S. K. Kakula, A. Pinar, T. Havens, and D. Anderson, "Visualization and analysis tools for explainable choquet integral regression," 2020.

[9] A. R. Buck, D. T. Anderson, J. M. Keller, T. Wilkin, and M. A. Islam, "A weighted matrix visualization for fuzzy measures and integrals," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2020, pp. 1–8.

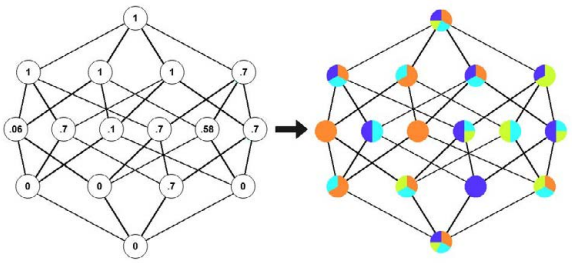
[10] A. J. Pinar, T. C. Havens, M. A. Islam, and D. T. Anderson, "Visualization and learning of the choquet integral with limited training data," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–6.

[11] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99–121, 01 2000.

[12] E. Levina and P. Bickel, "The earth mover's distance is the mallows distance: some insights from statistics," in *Proceedings Eighth IEEE International Conference on*



(a) (left) iVAT image and (right) color-coded CLODD results.



(b) (left) Hasse diagram with measure values and corresponding (right) color-coded pie chart diagram

Fig. 5: iVAT, CLODD, and color-coded Hasse diagram for the real-world Experiment 3.

Computer Vision. ICCV 2001, vol. 2, 2001, pp. 251–256 vol.2.

[13] S. K. Kakula, A. Pinar, T. Havens, and D. Anderson, "Choquet integral ridge regression," 07 2020.

[14] R. Yossi, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, 11 2000.

[15] S. Shirdhonkar and D. W. Jacobs, "Approximate earth movers distance in linear time," *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[16] V. Torra, Y. Narukawa, M. Sugeno, and M. Carlson, "Hellinger distance for fuzzy measures," *8th conference of the European Society for Fuzzy Logic and Technology*, 08 2013.

[17] H. Agahi, "A generalized hellinger distance for choquet integral," *Fuzzy Sets and Systems*, vol. 396, pp. 42 – 50, 2020, generalized Measures and Integrals.

[18] S. K. Kakula, A. Pinar, M. A. Islam, D. Anderson, and T. Havens, "Novel regularization for learning the fuzzy choquet integral with limited training data," *IEEE Transactions on Fuzzy Systems*, pp. 1–1, 2020.

[19] G. Beliakov, "Construction of aggregation functions from data using linear programming," *Fuzzy Sets and Systems*, vol. 160, no. 1, pp. 65–75, 2009.

[20] M. Grabisch, H. T. Nguyen, and E. A. Walker, *Fundamentals of uncertainty calculi with applications to fuzzy inference*. Springer Science & Business Media, 2013, vol. 30.

[21] M. A. Islam, D. Anderson, F. Petry, and P. Elmore, "An efficient evolutionary algorithm to optimize the choquet integral," *International Journal of Intelligent Systems*, 09 2018.

[22] A. Mendez-Vazquez and P. Gader, "Sparsity promotion models for the choquet integral," in *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on*. IEEE, 2007, pp. 454–459.

[23] L. Wang, U. Nguyen, J. Bezdek, C. Leckie, and K. Ramamohanarao, "ivat and avat: Enhanced visual analysis for cluster tendency assessment," vol. 6118, 06 2010, pp. 16–27.

[24] T. C. Havens, J. C. Bezdek, J. M. Keller, and M. Popescu, "Clustering in ordered dissimilarity data," in *International Journal of Intelligent Systems*, vol. 24, 2009, pp. 504–528.

[25] G. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, and X. Lu, "Aid: A benchmark data set for performance evaluation of aerial scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3965–3981, July 2017.

[26] D. T. Anderson, M. A. Islam, R. King, N. H. Younan, J. R. Fairley, S. Howington, F. Petry, P. Elmore, and A. Zare, "Binary fuzzy measures and choquet integration for multi-source fusion," in *2017 International Conference on Military Technologies (ICMT)*, 2017, pp. 676–681.

[27] X. Du, A. Zare, and D. T. Anderson, "Multiple instance choquet integral with binary fuzzy measures for remote sensing classifier fusion with imprecise labels," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019, pp. 1154–1162.

[28] X. Du, A. Zare, J. M. Keller, and D. T. Anderson, "Multiple instance choquet integral for classifier fusion," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 1054–1061.

[29] X. Du and A. Zare, "Multiple instance choquet integral classifier fusion and regression for remote sensing applications," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 5, pp. 2741–2753, 2019.

[30] G. J. Scott, K. C. Hagan, R. A. Marcum, J. A. Hurt, D. T. Anderson, and C. H. Davis, "Enhanced fusion of deep neural networks for classification of benchmark high-resolution image data sets," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 9, pp. 1451–1455, Sep. 2018.

[31] J. A. Hurt, G. J. Scott, D. T. Anderson, and C. H. Davis, "Benchmark meta-dataset of high-resolution remote sensing imagery for training robust deep learning models in machine-assisted visual analytics," in *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2018, pp. 1–9.

[32] G. J. Scott, J. A. Hurt, R. A. Marcum, D. T. Anderson, and C. H. Davis, "Aggregating deep convolutional neural network scans of broad-area high-resolution remote sensing imagery," in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018, pp. 665–668.

Chapter 12

Title: Doing more with less: similarity neural nets and metrics for small class imbalanced data sets

Venue: SPIE Defense + Commercial Sensing

Date Published: April 24, 2020

PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Doing more with less: similarity neural nets and metrics for small class imbalanced data sets

Charlie Veal, Jeffrey Schulz, Andrew Buck, Derek Anderson, James Keller, et al.

Charlie Veal, Jeffrey Schulz, Andrew Buck, Derek Anderson, James Keller, Mihail Popescu, Grant Scott, Dominic Ho, Timothy Wilkin, "Doing more with less: similarity neural nets and metrics for small class imbalanced data sets," Proc. SPIE 11418, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV, 1141802 (24 April 2020); doi: 10.1117/12.2558092

SPIE.

Event: SPIE Defense + Commercial Sensing, 2020, Online Only

Invited Paper

Doing More With Less: Similarity Neural Nets and Metrics for Small Class Imbalanced Data Sets

Charlie Veal[a], Jeffrey Schulz[a], Andrew Buck[a], Derek Anderson[a], James Keller[a], Mihail Popescu[a], Grant Scott[a], Dominic Ho[a], and Timothy Wilkin[b]

[a]Department of Electrical Engineering and Computer Science, University of Missouri, Columbia MO, USA
[b]School of Information Technology, Deakin University, Geelong, Victoria, AU

ABSTRACT

The focus of this article is deep learning on small, class imbalanced data sets in support of explosive hazard detection (EHD) and automatic target recognition (ATR). To this end, we explore artificial neural networks that are driven by similarity versus classification or regression. Similarity can be emphasized via network design, e.g., siamese networks, and/or underlying metric, e.g., contrastive or triple loss. The general goal of a similarity neural network (SNN) is discriminative training via focusing on similarity between tuples of like (and unlike) inputs. As such, SNNs have the potential to learn improved solutions on small data; aka "do more with less". Herein, we explore different avenues and we show that SNNs are essentially neural feature extractors followed by k-nearest neighbor classification. Instead of experimenting on a government data set that cannot be shared, we instead focus on benchmark community data sets for sake of reproducible research. Preliminary findings are encouraging as they suggest that SNNs have great potential for tasks like EHD and ATR.

Keywords: artificial neural network; siamese network; triplet loss; deep learning; explosive hazard detection; automatic target recognition; similarity network

1. INTRODUCTION

In our current golden age of data driven learning, deep feature extraction has become one of the largest contributions of modern neural networks. While this process can lead to state-of-the-art performance, it typically comes at an *expense*; the need for large datasets that embody both class balance and diversity. The neural network community is no stranger to this requirement, and it has accordingly embraced a variety of countermeasures. In[1,2] generative models were explored in combination with traditional data augmentations to improve the size and quality of the training data. In[3,4] regularization techniques were utilized to help weights generalize to smaller datasets. Another strategy is a re-emerging methodology of neural networks that focus on variations of discriminatory training. Examples of these networks are siamese neural networks,[5] and triplet neural networks.[6] Siamese networks are best known for one-shot learning[7,8] and object tracking.[9–11] Triplet networks are best known for one shot learning[12] and face detection.[13,14] The goal of these SNNs is to find an intelligent way to exploit the similarity and dissimilarity between inter-class datasets to learn better feature embeddings, which ultimately aid in classification tasks such as target detection, target localization, etc. However, in practice, how do these SNN models differ from mainstream convolutional neural networks (CNNs)? Herein, we focus on the following questions. First, we know that siamese and triplet networks embrace novel forms of discriminatory training.[5,6] When data is limited, does this form of training allow these models to perform better than traditional CNNs? Second, what do these training strategies mean with respect to analyzing feature space?

We make the following contributions. First, we investigate the utility in using SNNs vs CNNs for object detection, with respect to limited data, through benchmark analysis and custom datasets. Second, we provide we provide

Send correspondence to Charlie Veal
Charlie Veal: E-mail: ctvqfq@mail.missouri.edu

Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV, edited by Steven S. Bishop, Jason C. Isaacs, Proc. of SPIE Vol. 11418, 1141802 · © 2020 SPIE
CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2558092

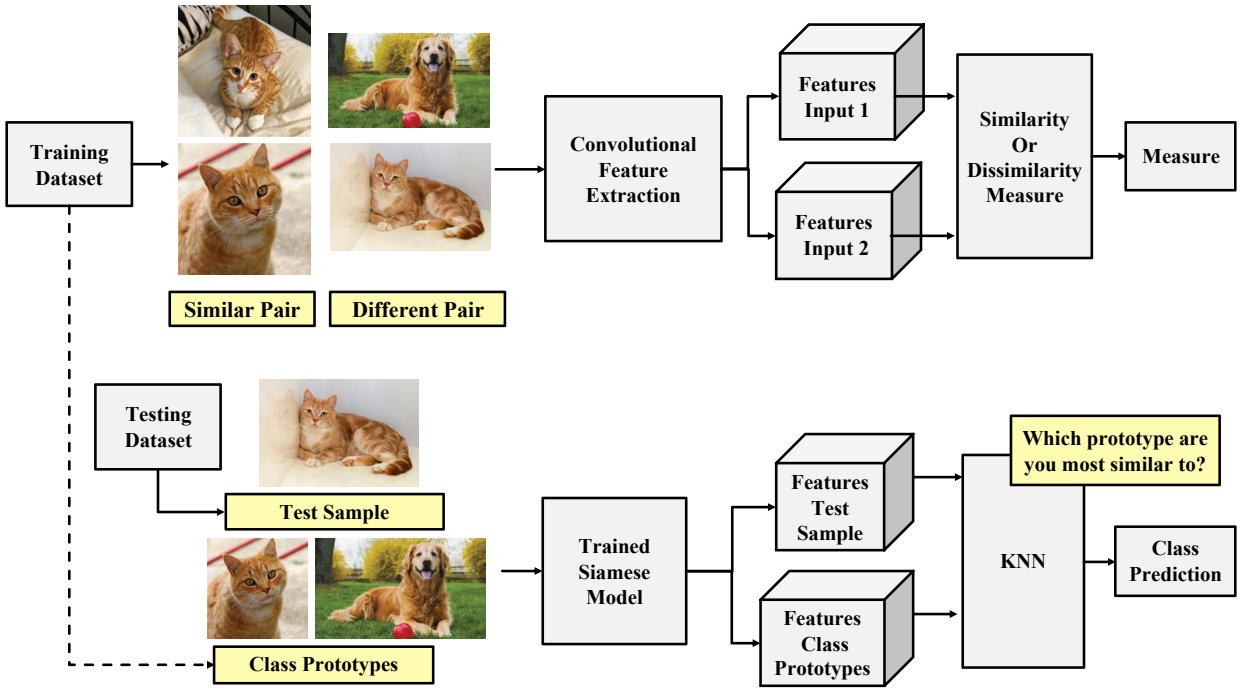


Figure 1: Visualization of the training and testing phases in a siamese net. The top diagram illustrates training and the bottom diagram is testing. Section 2 explains each process in detail.

qualitative visualizations of feature embeddings, highlighting the effects of the different SNN training paradigms. The remainder of this article is organized as follows. First, siamese networks are introduced in Section 2 and triplet networks in Section 3. Afterwards, the implementation details and experiments follow in Section 4. Tables 1, 2, 3 summarize the multi-class benchmark analysis and Tables 4, 5, 6 summarize the custom dataset assessment. Lastly, conclusions and future work are found in Section 5.

2. SIAMESE NEURAL NETWORKS

Siamese networks, originally introduced by LeCun and Bromley for signature verification,[5] have been shown to be capable of maximizing performance with respect to limited data volume and variety.[7–9] Simply put, this net is a structure that focuses on learning feature similarity rather than traditional pattern classification or regression. For a conceptual visualization of the model, one can reference Figure 1.

2.1 Siamese Training

Before training can begin, siamese networks require data preparation in the form of a tuplewise assortment. Let $\boldsymbol{x}$ represent an image from training dataset, such that $\{\boldsymbol{x_0}, \boldsymbol{x_1}, ..., \boldsymbol{x_n}\} \in X$. For data preparation, X is broken down into a set of c pairs $\{s_0^l, s_1^l, ..., s_c^l\}$, where s_c^l represents the c-th pair of inputs, e.g., $\{\boldsymbol{x_1}, \boldsymbol{x_2}\}_c^l$, with training label l. The training label l is assigned 0 if elements of the pair s_c^l belong to the same class. Otherwise, the the elements of the pair are from different classes and l is assigned 1.

After data preparation, these pairs are fed into the siamese network for model training. Each pairwise input s_c^l undergoes feature extraction through various convolutional layers. This results in a corresponding set of activation vectors $f(s_c^l)$, which represents the pair of dense activations corresponding to the pair of inputs, i.e $\{f(\boldsymbol{x_1}), f(\boldsymbol{x_2})\}_c^l$. Unlike most modern CNNs, a soft max activation layer—which represents the input belonging to one of N classes—is not present. Instead, the activation vectors $f(s_c^l)$ are passed to a dissimilarity layer. This produces the measure d, which represents how dissimilar the feature spaces are between the original pairwise inputs. For an illustration, reference the top diagram in Figure 1. Lastly, using the gradients with respect to this loss function, the siamese network is trained with traditional backpropagation with mini-batch gradient descent.

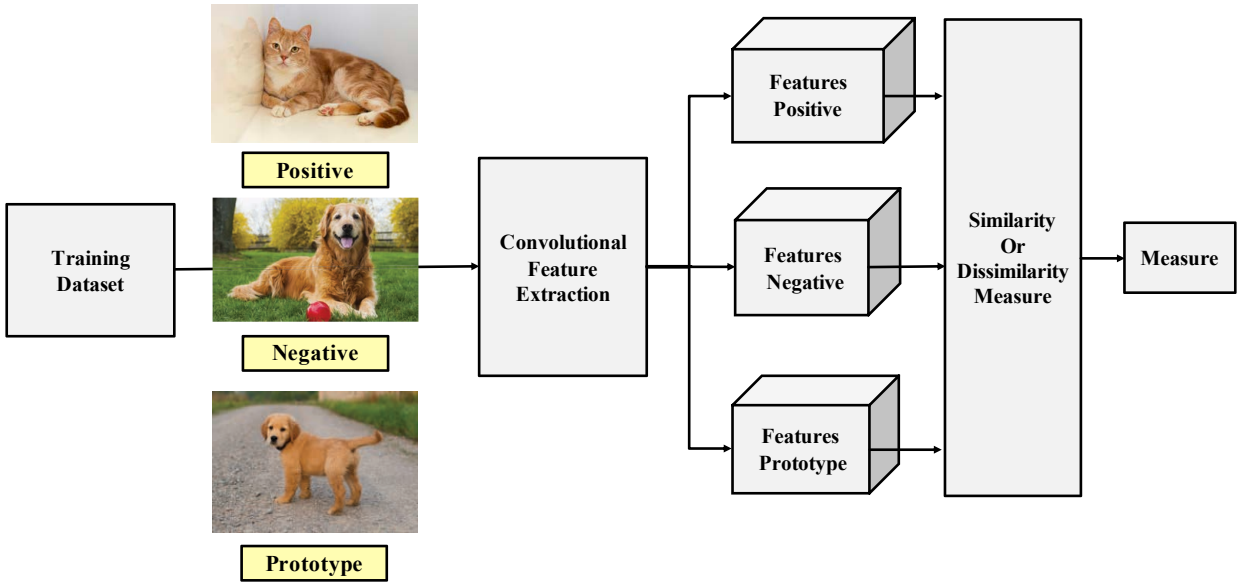


Figure 2: Visualization of training in a triplet network. Section 3.1 explains triplet training in detail.

2.2 Siamese Evaluation

.

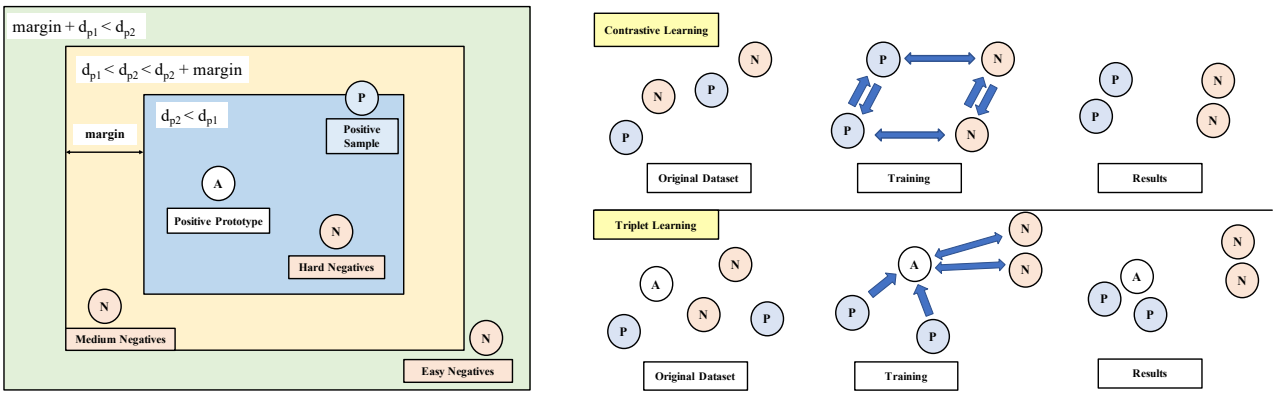
Model evaluation for a trained siamese network is similar to the training paradigm. However, there are a few important differences. First, the labels for siamese evaluation are traditional supervised labels, i.e., truth labels that correspond with each test sample. Second, model evaluation requires both a test dataset and a selection of class prototypes, where each prototypes serves as a representative of its corresponding class from the train dataset. Let $\boldsymbol{z}$ represent an image from the testing dataset, such that $\{\boldsymbol{z_0}, \boldsymbol{z_1}, ..., \boldsymbol{z_n}\} \in T$. Let $\boldsymbol{p}$ represent a class prototype image originating from training dataset, such that $\{\boldsymbol{p_0}, \boldsymbol{p_1}, ..., \boldsymbol{p_n}\} \in X$. For each $\boldsymbol{z}$, the trained siamese network uses $\{\boldsymbol{p_0}, \boldsymbol{p_1}, ..., \boldsymbol{p_n}\}$ and produces K $\{d_0, d_1, ..., d_k\}$, i.e, one dissimilarity score for each class prototype. Afterwards, the KNN algorithm makes a class prediction from the most similar prototype(s) to the test sample. The distance metric in KNN is the metric used for siamese network training. For an illustration, reference the bottom diagram in Figure 1.

2.3 Contrastive Loss

Contrastive loss is an error metric used in settings like siamese networks that focuses on balancing similarity with dissimilarity across pairs of inputs. Specifically, this loss function is used for discriminatory training where the traditional notion of classes is removed (e.g., dog, cat, car, etc). Instead, this loss function requires only two class labels (similar, dissimilar) and because of this, inputs to the model are either a similar pair (e.g., cat, cat) or dissimilar pair (e.g., cat, dog). The constrastive margin loss function, for one siamese pair, is defined as the following:

$$J(d, l, m) = (1 - l)\frac{1}{2}(d)^2 + (l)\frac{1}{2}\{\max(0, m - d)\}^2. \quad (1)$$

In Equation 1, l represents the training label and d represents the dissimilarity measure between the pairwise inputs to the model, i.e., $d(f(\boldsymbol{x_1}), f(\boldsymbol{x_2}))$. The margin, m, is a threshold such that dissimilarity values that are greater than this value do not contribute to the loss function.[15] The goal of this equation is to minimize the distance between features produced by the same classes, while maximizing the distance between features produced by different classes.[15] An example illustration of contrastive loss is shown in figure 3b. Looking at this figure, one can see that after training, the instances of similar classes are brought together and the instances of different classes are separated.



(a) (b)

Figure 3: Illustrations of triplet learning. (a) Example of online negative mining for triplet loss. Details on this procedure is be found in section 3.2. (b) Example of contrastive loss vs triplet loss. Details on contrastive loss is found in section 2.3. Details on triplet loss is found in section 3.2.

3. TRIPLET NEURAL NETWORKS

Triplet networks, inspired by the discriminatory learning of siamese networks,[6] have been shown to be capable of achieving state of art performance for face detection.[13,14] Similar to siamese networks, this net also focuses on learning a similarity and dissimilarity between classes of data. However, instead of just using pairwise inputs for training, a triplet net requires three inputs: positive samples, negative samples, and prototypical examples.

3.1 Triplet Training

An example of training a triplet network is shown in Figure 2. One can see that the positive and negative samples are simply two instances of different classes. The prototype, however, is an anchor reference that is used to facilitate a basis for similarity and dissimilarity. For data preparation, let X be broken down into a set of c triplets $\{t_0, t_1, ..., t_c\}$, where t_c represents the c-th triplet of inputs (e.g., $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{p}\}_c$). Unlike training siamese networks, triplet learning does not require a training label for similarity. Instead, an example prototype $\boldsymbol{p}$, which belongs to the positive or negative class is required and serves as reference point. After data preparation, these triplets are fed into the triplet net for model training. Each el ement of the triplet t_c undergoes convolutional feature extraction, resulting in feature embeddings for the positive class, the negative class, and the prototype example, i.e., $\{f(\boldsymbol{x_1}), f(\boldsymbol{x_2}), f(\boldsymbol{p})\}_c$. Next, positive and negative features are compared to the features of the prototype example in a dissimilarity layer. This produces two measures ,d_{p1} and d_{p2}. The measure d_{p1} represents dissimilarity between the positive sample and the prototype example. The measure d_{p2} represents dissimilarity between the negative sample and the prototype example. Using these dissimilarities, triplet networks are trained with traditional backpropagation and mini-batch gradient descent. They also follow the siamese net evaluation paradigm shown in Section 2.2.

3.2 Triplet Loss

Triplet loss is the error metric of triplet networks that allows the model to learn similarity and dissimilarity.[13,14,16] However, one big difference between itself and constrastive loss, covered in Section 2.3, is that triplet loss relies on an prototypical example $\boldsymbol{p}$ as the truth-label. The triplet loss function is defined as the following:

$$J(d_{p1}, d_{p2}, m) = max(d_{p1} - d_{p2} + m, 0). \tag{2}$$

In Equation 2, d_{p1} represents the dissimilarity measure between the prototype and positive class sample while d_{p2} represents the dissimilarity measure between the prototype and negative class sample, i.e., $d(f(\boldsymbol{p}), f(\boldsymbol{x_1}))$ and $d(f(\boldsymbol{p}), f(\boldsymbol{x_2}))$ respectively. The margin term m holds the same meaning from the contrastive loss Equation 1.

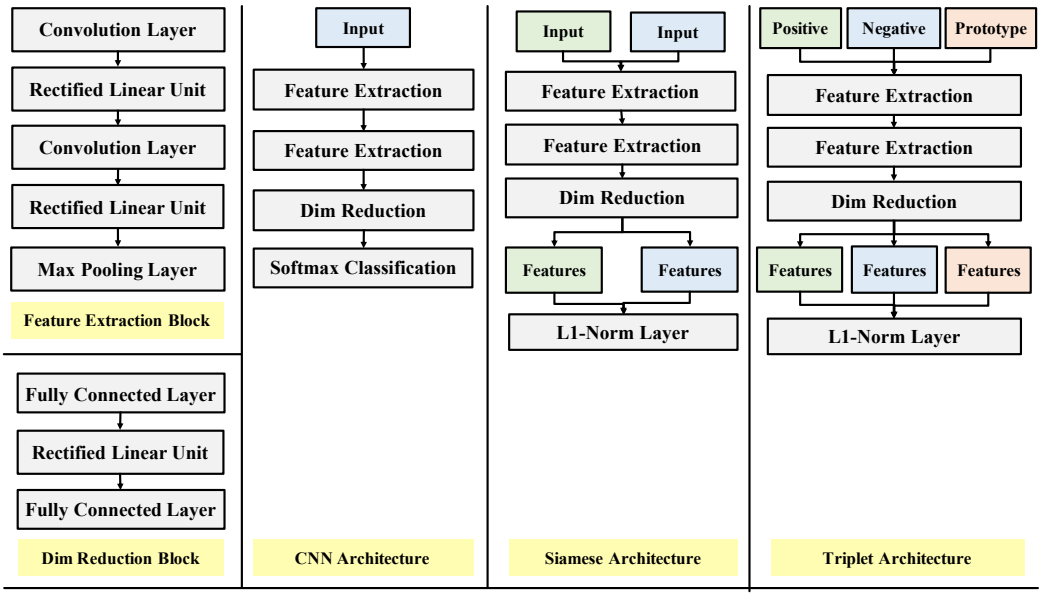


Figure 4: Illustration of network architectures for the CNN, siamese, triplet models.

Using triplet loss, one will notice the possibility to return zero error. This by design is attributed to having non-optimal triplets, and because of this, triplet selection is a very sensitive procedure. While there have been many different methodologies proposed for triplet selection,[13,16] the basic approach focuses on constructing triplets *online* during the training process. Specifically, this means that for each mini-batch of a training epoch, ideal triplets must be intelligently constructed to facilitate non-stagnant learning. For the remainder of this section, the explained triplet selection strategy will be with respect to online negative mining.[13] This process involves finding negative samples that constrain the loss function to produce non-zero returns. With this in mind, each triplet will consist of positive prototypes, positive samples and negative samples.

One can reference Figure 3a for an illustration of triplet selection with respect to online negative mining. Looking at this figure, the negative samples of interest can be grouped into three categories: easy negatives, medium negatives, and hard negatives. Easy negative samples are the least wanted members of the negative batch. This is because these negative samples are already too far away from the positive prototypical example, i.e, $d_{p1} - d_{p2} + m < 0$. Using these samples forces the triplet loss function to produce zero error and therefore contribute the least to the learning process of the model. Hard negatives are more ideal to have than easy negatives, but can also be detrimental to triplet learning. This is because these samples are very close to the positive positive prototypical example, i.e., $d_{p2} < d_{p1}$, and therefore produce the largest response from the loss function. Specifically, too many of these hard negatives could result in a collapsed model. Medium negatives are the most ideal negative members to have for online triplet selection. This is because they are not closer to the positive prototype than the positive sample and also are not too far away. This results in the medium negative samples being $d_{p1} < d_{p2} < d_{p1} + m$, and because of this, produces the most consistent error for triplet training.[13]

4. EXPERIMENTS & RESULTS

The goal of this section is to explore the utility of using SNNs (e.g., siamese contrastive loss and triplet loss-based models) vs a traditional CNN with respect to object detection. Specifically, we want to address two questions. First, when data is limited, does using contrastive or triplet loss provide any classification benefit over just using a CNN? Second, do these discriminatory training strategies provide any benefit with respect to feature space analysis? To answer these questions, we perform two experiments focused on object detection with limited data. First, we run the models through basic benchmark datasets (e.g., MNIST,[17] Fashion MNIST,[18] and CIFAR-10[19]) to compare their multi-class detection accuracy. Second, we explore three binary class problems

	N = 5000	N = 1000	N = 500	N = 100	N = 20
CNN	99.4%	97.5%	97.3%	91.1%	54.7%
Siamese	99.2%	98.2%	97.7%	94.2%	86.8%
Triplet	97.1%	95.4%	94.6%	90.5%	79.1%

Table 1: Benchmark results of MNIST dataset across different class sample sizes

	N = 5000	N = 1000	N = 500	N = 100	N = 20
CNN	92.0%	86.7%	82.7%	69.7%	56.5%
Siamese	90.0%	86.1%	84.4%	78.7%	73.8%
Triplet	85.8%	81.0%	81.0%	73.9%	68.3%

Table 2: Benchmark results of F-MNIST dataset across different class sample sizes

from the food-101 dataset[20] to evaluate the learning capabilities of these models on higher resolution imagery. However, for each dataset used, the number of samples per class will be incrementally decreased to also evaluate each models detection capabilities under the constraints of limited data. Throughout these experiments, feature visualizations strategies will also be explored in order to assess any additional utility of the SNNs outside of classification results.

4.1 Community Benchmark Datasets

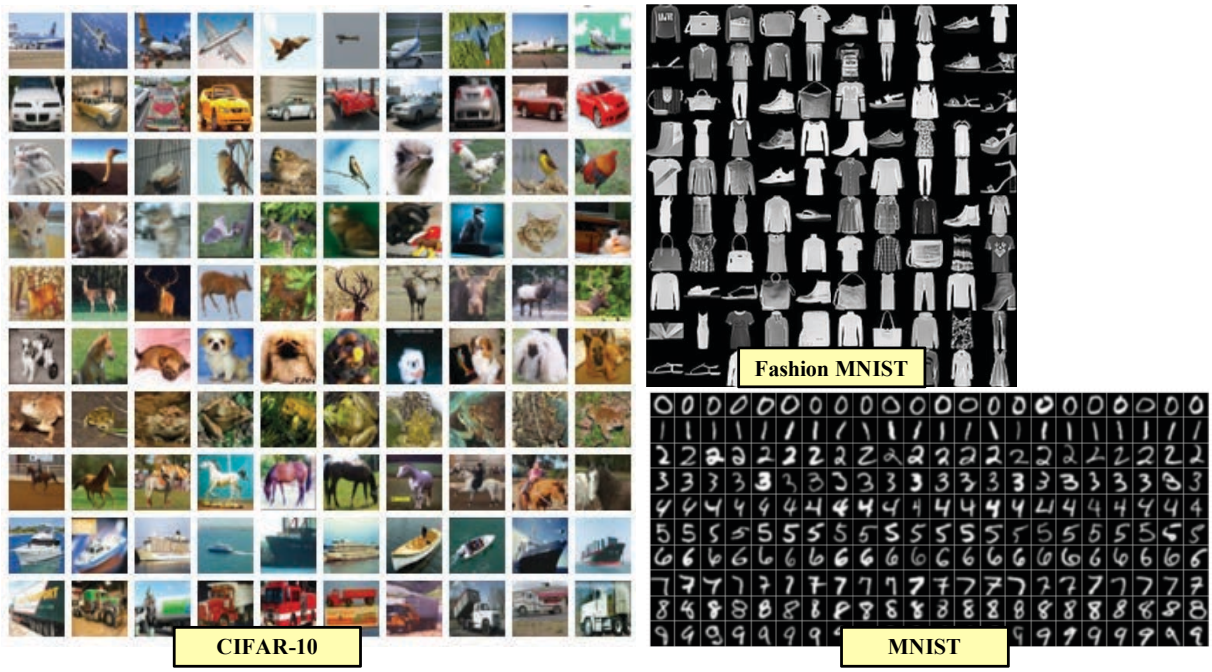
We chose the proposed datasets for the following reasons. First, the MNIST[17] dataset was selected for validating the basics of both the siamese and triplet learning paradigms. Next, we chose Fashion MNIST (F-MNIST)[18] to illustrate each models ability to distinguish between similar class shape and structure. Next, CIFAR-10[19] was selected to test each models multi-class capabilities on targets with more complex features and varying backgrounds. Lastly, the binary datasets, originating from food-101,[20] were chosen due to their high resolution, small initial data sizes, and extremely similar features. Examples from all datasets can be seen in Figure 5.

4.2 Evaluating Multi-Class Detection

The purpose of this experiment is to analyze the multi-class performance capabilities between the SNNs and a traditional CNN. To facilitate reproducible research, the following guidelines were used to create this experiment. First, each dataset underwent basic normalization. This process involves assuring each sample to a (28, 28) spatial size and then standardizing their channel outputs between (-1, 1), i.e., zero-mean and divide by standard deviation per channel. Next, a shallow feature extraction network (similar style as a VGG network[21]) was created to produce feature embeddings across all models. The CNN architecture was then made attaching a multi-layer-perceptron (MLP) followed by softmax normalization. The siamese and triplet architectures were made by instead attaching a dissimilarity layer. For a visualization of each training architecture, see Figure 4. Each of these models are trained with ADAM optimization[22] with initial learning rate of 0.001 for 50 epochs. For the CNN, multi-class cross entropy was the loss function. For the siamese network, contrastive margin loss function was used with a margin value of 1 and an L1-norm dissimilarity measure. For the triplet network, online negative mining, the variation discussed in Section 3.2, was used as well as an L1-norm dissimilarity measure. Lastly, for evaluating the test samples with the siamese and triplet models, KNN was utilized with $K = 3$. The class prototypes for KNN were 10 random training samples per class.

	N = 5000	N = 1000	N = 500	N = 100	N = 20
CNN	70.4%	59.8%	54.6%	39.9%	23.5%
Siamese	63.3%	52.3%	48.8%	38.1%	25.5%
Triplet	60.4%	54.9%	46.0%	38.2%	28.2%

Table 3: Benchmark results of CIFAR-10 dataset across different class sample sizes



(a)

(b)

Figure 5: Illustration of datasets. (a) Visualizations of MNIST,[17] F-MNIST,[18] and CIFAR-10[19] datasets. (b) Visualizations of the custom binary datasets, originating from food-101.[20] These datasets are ramen vs bulgogi (dataset 1), donuts vs beignets (dataset 2), and pancakes vs waffles (dataset 3).

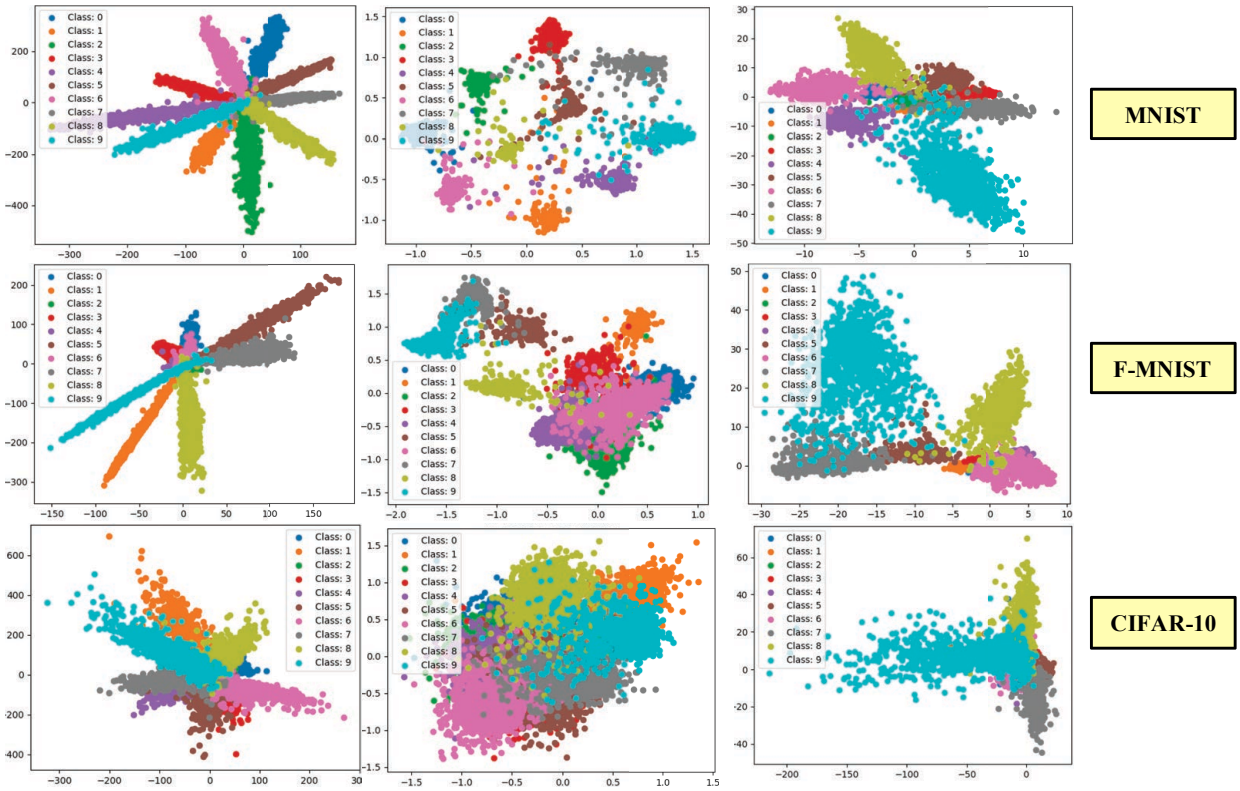


Figure 6: Implicit model feature space visualizations of the following multi-class test datasets: MNIST (top row), F-MNIST (middle row), and CIFAR-10 (bottom row). Each column shows feature embeddings with the following model order: CNN, siamese, triplet.

4.2.1 Data Visualization

For data visualization, we constrain each network to learn an implicit mapping to two dimensional space. This was achieved by having each model project features down to two dimensional space before their final output layer. Using this simple strategy, one can visualize feature separation while still allowing each model to focus on the classification task. Visualizations for each dataset can found in Figure 6. Looking at this figure, one can notice the following trends. First, the traditional CNN training paradigm (first column) produces feature embeddings that look mostly separable. However, unlike the siamese and triplet learning strategies, the CNN does not prioritize feature separation in learning. Thus the CNNs feature space partitions are unintentional boon of the training procedure. Second, the siamese training paradigm (second column) produces more spread out clusters of different classes due to the constrastive loss ideology. Using this inherit projection capability is valuable to algorithms where spatial knowledge assessment is needed (e.g. clustering strategies, nearest neighbor strategies, etc.). Third, the triplet training paradigm (third column) produces clusters in between the embeddings shown by the siamese and the CNN learning strategies. This means that depending on the triplet selection strategy, i.e, the quality of triplets used, one has the potential to produce similar or better cluster separation than the siamese learning strategy or the simple separation of the CNN learning strategy.

4.2.2 Benchmark Analysis

For the benchmark analysis, results for MNIST are found in Table 1, results for F-MNIST are found in Table 2, and results for CIFAR-10 are found in Table 3. Observing these tables, each model was run across a variety of class sample sizes (N) which corresponds to the number of instances available for training inside of each class. So for this analysis, we are not just testing multi-classification accuracy, but also models accuracy with respect to a diminishing size of data. From these tables, we made the following observations. First, the siamese and

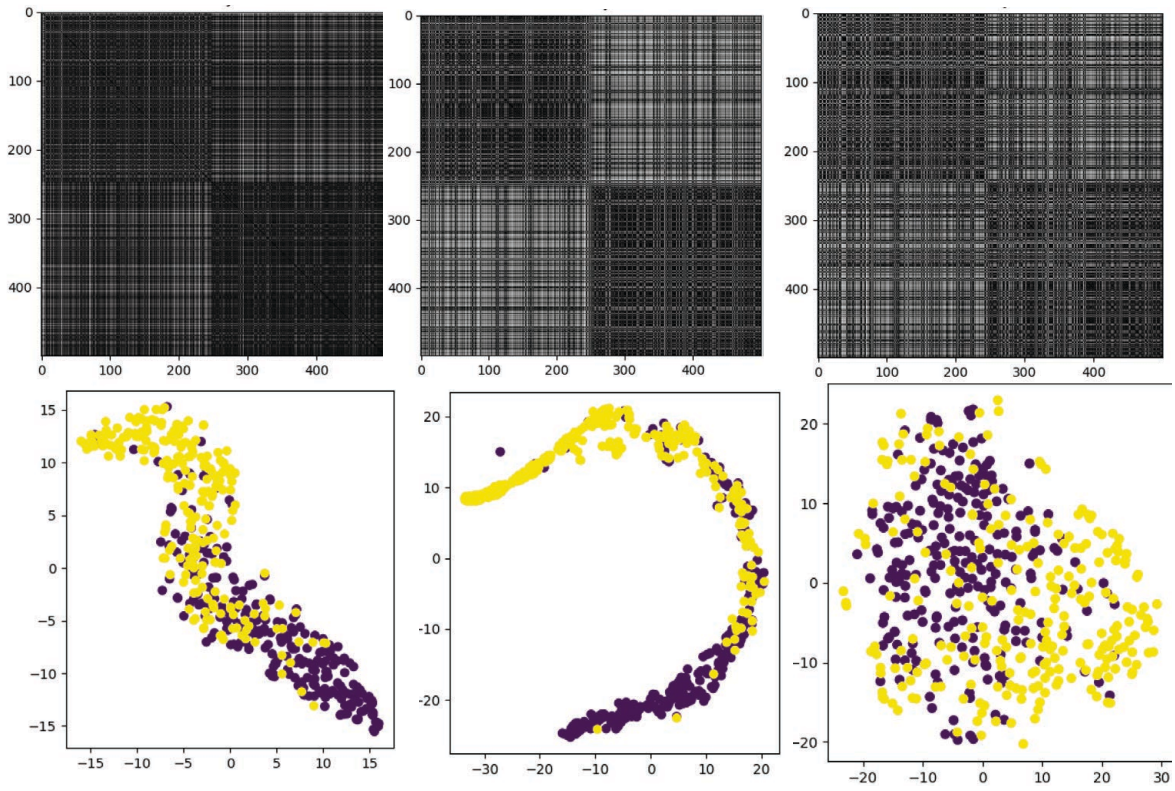


Figure 7: Visualizations of high dimensional feature space using TSNE, with a perplexity value of 30, and their corresponding class sorted dissimilarity matrix for each dataset. The top row shows the dissimilarity matrices, and the bottom row is TSNE plots. Each column shows visualizations with the following model order: CNN, siamese, triplet.

triplet models were able to perform well on MNIST and F-MNIST dataset in comparison to the CNN. As the data became limited, the CNN models classification accuracy lowered as expected. However, the siamese and triplet models were able to maintain much higher performance. This shows potential such that when data is limited, using a SNN training paradigm can be advantageous, even for a multi-class problem. Second, the SNN models can follow the classification accuracy decline of the CNN. This is because when dealing with a dataset like CIFAR-10, classification becomes increasingly more difficult due to the combination of similar features and high variety of backgrounds. On top of this, the learning strategies of these discriminatory models is much more constrained than the basic approach of the CNN. Lastly, when observing all the results, we believe when a multi-class dataset is large and ideal, the CNN should be the algorithm of choice. However, when data size is limited or when further metric analysis is valuable, the discriminatory training paradigms of the SNNs should be considered as an alternative over the CNN.

4.3 Evaluating Binary High Resolution

The purpose of this experiment is to analyze the learning capabilities of the SNNs on higher resolution imagery consisting of limited size and binary classes. To facilitate reproducible research, the guidelines mentioned in Section 4.2.2 were used as a basis, but with a few modifications. First, since we are using higher resolution imagery, each sample was normalized to (128, 128) spatial size and underwent channel standardization. Next, the number of training epochs was extended to 100 instead of the original proposed 50. Lastly, referencing the same model architecture guidelines shown in figure 4, each model was made deeper by adding more feature extraction blocks.

	N = 750	N = 500	N = 250	N = 100	N = 20
CNN	79.4%	77.8%	68.6%	64.8%	58.7%
Siamese	81.2%	73.6%	69.0%	67.2%	64.6%
Triplet	80.0%	77.4%	65.2%	61.0%	61.8%

Table 4: Evaluation results of ramen vs bulgogi across different class sample sizes

	N = 750	N = 500	N = 250	N = 100	N = 20
CNN	75.6%	70.2%	65.8%	58.8%	61.4%
Siamese	71.2%	68.6%	64.6%	65.4%	64.8%
Triplet	75.0%	73.2%	68.4%	62.6%	59.9%

Table 5: Evaluation results of donuts vs beignets across different class sample sizes

4.3.1 Data Visualization

Before exploring the binary classification results, we first wanted qualitative assurance that each trained model provided a feature space which minimized class overlap. To validate this, we used T-distributed Stochastic Neighbor Embedding (TSNE) and a class sorted dissimilarity matrix for data visualization of high dimensional spaces. Using these methods, we produced the example illustrations found in Figure 7 for dataset 1 (ramen vs bulgogi). Looking at this figure, the top row are VAT imagery, and the bottom row are TSNE imagery. Each column holds the visualizations with respect to the following model order: CNN, siamese, triplet. Each dissimilarity image is scaled from min to max distance for display purposes. Each axis represents all samples, where class 1 comes first, followed by class 2. Thus, matrix/image element (i, j) is the L1-norm between sample i and j. Thus, darker areas show similarity and brighter is dissimilarity. Thus, it is ideal to have dark blocks along the matrix diagonal and white on the off diagonal. TSNE is a nonlinear, and non-deterministic, projection technique, that uses local relationships between points to create a low-dimensional mapping. However, while it observes general topology, the distances between points in high-dimensional space is not well preserved, and will sometimes become distorted, even when tuning the perplexity parameter. Nevertheless, for our visualizations, TSNE was explored across a range of perplexities (10, 30, 50, 100) and served as an alternative strategy to illustrate that the siamese training was able to separate two opposing classes. More details on TSNE can be found here.[23]

4.3.2 Classification Performance

For the binary dataset analysis, the results are found in the following tables: 4, 5, 6. Looking at the tables, we made the following observations. First, when trained on low sample and high resolution imagery, the siamese and triplet networks have shown the capabilities to train better than the CNN. Unlike the previous experiment in section 4.2.2, where multi-class assessment was the primary interest, the discriminatory networks now have a much narrower problem scope. We believe this allows for more focus on feature separation, and consequently, better classification accuracy. We also believe this reinforces the idea that the complex learning paradigms of siamese and triplet models transition better to smaller class problems. Second, when comparing the classification results to the visualization techniques found in figure 7, we noticed the dissimilarity matrices were a good representation of the recorded table results. This is due to the dark blocks along the matrix diagonal which signifies similarity amongst samples in the same classes. The siamese network in particular had the lightest blocks in the off diagonal. This illustrates how the network was able to learn better feature separation in comparison to the

	N = 750	N = 500	N = 250	N = 100	N = 20
CNN	83.2%	79.0%	82.0%	75.4%	70.8%
Siamese	87.2%	82.0%	72.4%	75.6%	71.4%
Triplet	72.2%	73.4%	70.2%	66.1%	65.3%

Table 6: Evaluation results of pancakes vs waffles across different class sample sizes

CNN. The TSNE plots illustrate another perspective of the learned feature embeddings. For each of these plots, there is overlap between the two classes. We believe this is due to the how TSNE uses the local relationships between the points it observes to create its low-dimensional mappings. Because of this, we can infer for that TSNE to produce highly separable feature embeddings, the points would already have to be very separable in higher dimensional feature space. This could be achieved from having an easier dataset or a more powerful trained model. However, in the case of this experiment, the points were separable enough for each trained model to score the results found in the tables.

5. CONCLUSIONS

Herein, this paper explored the utility in using SNNs vs a traditional CNN for object detection with respect to limited data. For each model, this paper illustrated the trade offs between classification performance and the data availability. This paper also gave qualitative insight to the learning paradigms of siamese and triplet networks. The experiments in this paper showed the following results. First, when dealing with multi-class object detection and data is not limited, the CNN is the better algorithm for task. However, as data becomes very limited in class samples, the performance of the CNN drastically declines and the siamese and triplet models become valid alternative strategies. Second, when dealing with high resolution imagery of limited class and sample size, the siamese and triplet models have potential to outperform the CNN. In future work, we will explore the following methods. First, we want to run more thorough benchmarks analyzing the capabilities of these discriminatory networks. For example, we would focus on using better sampling strategies for siamese pair or triplet selection, as well as any highlighting any advantages with respect to transfer learning an initial visual vocabulary. Furthermore, we also plan to improve the siamese network. For this we will focus on designing better learning metrics that have more flexibility with prioritizing feature separation while while maintaining the simplistic training style of a CNN. We will also investigate alternatives to the similarity and dissimilarity metrics that govern the networks ability to discriminate between features in higher dimensional spaces. Lastly, by exploring the proposed methods, we want to continue our pursuit to find alternative deep learning strategies that accommodate the reality of limited real world data.

ACKNOWLEDGMENTS

This work is partially funded by the Army Research Office (ARO) grants numbered W911NF-18-1-0153 and W911NF-19-1-0181 to support the U.S. Army RDECOM CERDEC NVESD.

REFERENCES

[1] Shorten, C. and Khoshgoftaar, T. M., "A survey on image data augmentation for deep learning," *Journal of Big Data* **6**, 60 (Jul 2019).

[2] Mikołajczyk, A. and Grochowski, M., "Data augmentation for improving deep learning in image classification problem," in [*2018 International Interdisciplinary PhD Workshop (IIPhDW)*], 117–122 (May 2018).

[3] Krogh, A. and Hertz, J. A., "A simple weight decay can improve generalization," in [*Advances in Neural Information Processing Systems 4*], Moody, J. E., Hanson, S. J., and Lippmann, R. P., eds., 950–957, San Francisco, CA: Morgan Kaufmann (1992).

[4] Zhao, H., Tsai, Y. H., Salakhutdinov, R., and Gordon, G. J., "Learning neural networks with adaptive regularization," *CoRR* **abs/1907.06288** (2019).

[5] Jane Bromley, Isabelle Guyon, Y. L. E. S. and Shah, R., "Signature verification using a "siamese" time delay neural network," *Proceeding NIPS'93 Proceedings of the 6th International Conference on Neural Information Processing Systems* , 737–744 (1993).

[6] Hoffer, E. and Ailon, N., "Deep metric learning using triplet network," (2014).

[7] Koch, G. R., "Siamese neural networks for one-shot image recognition," (2015).

[8] Shaban, A., Bansal, S., Liu, Z., Essa, I., and Boots, B., "One-shot learning for semantic segmentation," (09 2017).

[9] Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. S., "Fully-convolutional siamese networks for object tracking," in [*Computer Vision – ECCV 2016 Workshops*], Hua, G. and Jégou, H., eds., 850–865, Springer International Publishing, Cham (2016).

[10] Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., and Wang, S., "Learning dynamic siamese network for visual object tracking," in [*The IEEE International Conference on Computer Vision (ICCV)*], (Oct 2017).

[11] He, A., Luo, C., Tian, X., and Zeng, W., "A twofold siamese network for real-time object tracking," in [*The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], (June 2018).

[12] Ye, M. and Guo, Y., "Deep triplet ranking networks for one-shot recognition," (2018).

[13] Schroff, F., Kalenichenko, D., and Philbin, J., "Facenet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2015).

[14] Parkhi, O. M., Vedaldi, A., and Zisserman, A., "Deep face recognition," in [*Proceedings of the British Machine Vision Conference (BMVC)*], Xianghua Xie, M. W. J. and Tam, G. K. L., eds., 41.1–41.12, BMVA Press (September 2015).

[15] Hadsell, R., Chopra, S., and LeCun, Y., "Dimensionality reduction by learning an invariant mapping," in [*Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*], *CVPR '06*, 1735–1742, IEEE Computer Society, Washington, DC, USA (2006).

[16] Hermans, A., Beyer, L., and Leibe, B., "In defense of the triplet loss for person re-identification," (2017).

[17] LeCun, Y. and Cortes, C., "MNIST handwritten digit database," (2010).

[18] Xiao, H., Rasul, K., and Vollgraf, R., "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," (2017).

[19] Krizhevsky, A., Nair, V., and Hinton, G., "Cifar-10 (canadian institute for advanced research),"

[20] Bossard, L., Guillaumin, M., and Van Gool, L., "Food-101 – mining discriminative components with random forests," in [*European Conference on Computer Vision*], (2014).

[21]

[22] Kingma, D. and Ba, J., "Adam: A method for stochastic optimization," *International Conference on Learning Representations* (12 2014).

[23] van der Maaten, L. and Hinton, G., "Visualizing data using t-SNE," *Journal of Machine Learning Research* **9**, 2579–2605 (2008).