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.

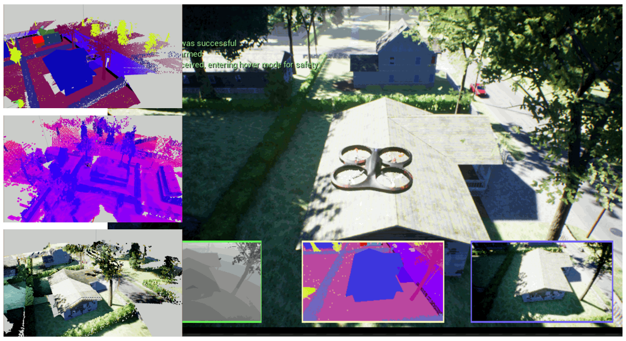


Figure 1. Example of a simulated aerial dataset for a rural neighborhood in AirSim and the Unreal Engine, automatically generated ground truth, voxel data, and 3D point cloud.

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.

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].

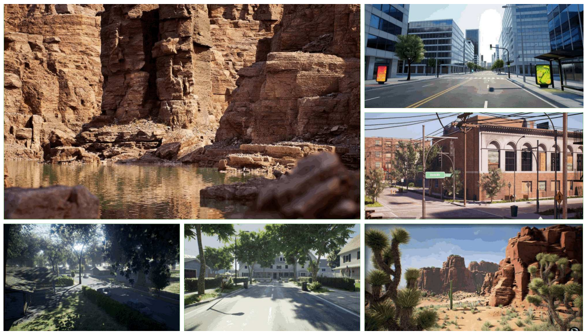


Figure 2. Example of urban and rural scenes–prices range from free to a few hundred dollars–on the UE Marketplace [18].

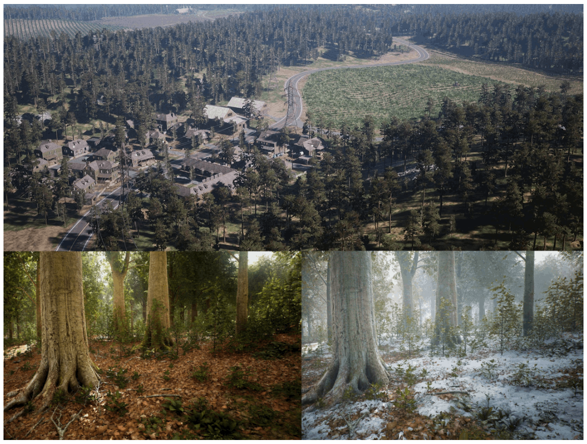


Figure 3. Example of large outdoor area from the UE Marketplace [18] and seasonal variation for train/test data variation.

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.

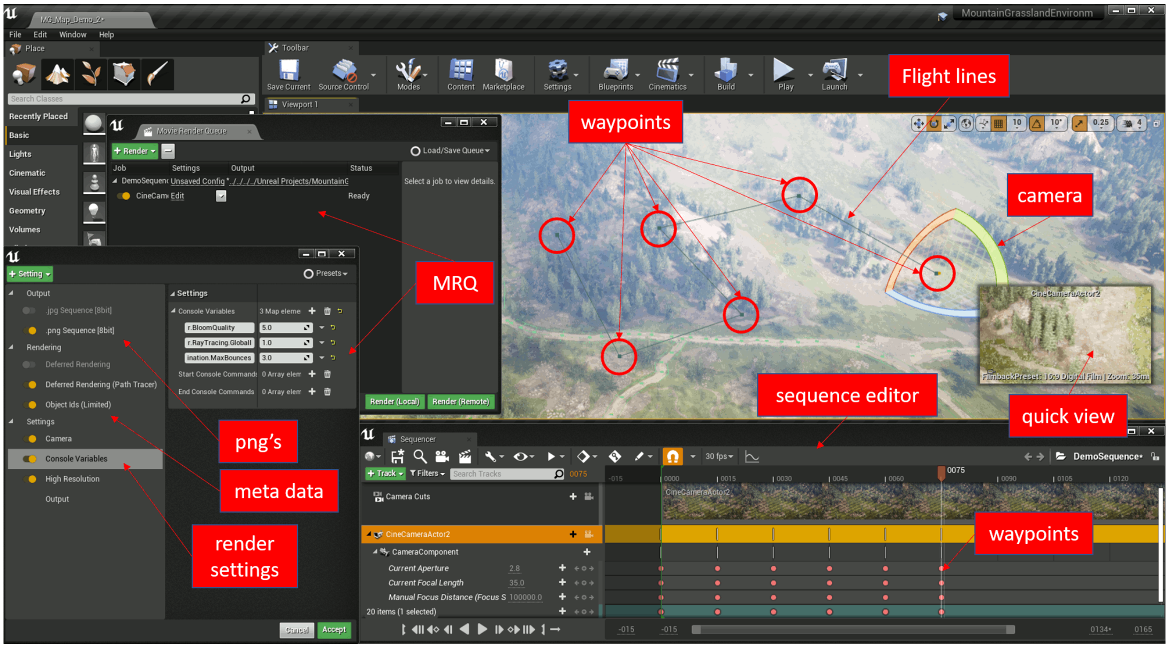
While at first AirSim appears to be the winner for all DL and UAV research, there are drawbacks. To start, the cost of entry is arguably higher, backed by significantly less documentation and tutorials. Second, AirSim only has access to a subset of UE functionality. This can be prohibitive if the reader 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



Output
- .jpg Sequence [8bit]
- .png Sequence [8bit]

Rendering
- Deferred Rendering
- Deferred Rendering (Path Tracer)
- Object Ids (Limited)

Settings
- Camera
- Console Variables
- High Resolution
- Output

Console Variables: 3 Map eleme
- r.BloomQuality 5.0
- r.RayTracing.GlobalI 1.0
- ination.MaxBounces 3.0

Start Console Commands: 0 Array elem
End Console Commands: 0 Array elem

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

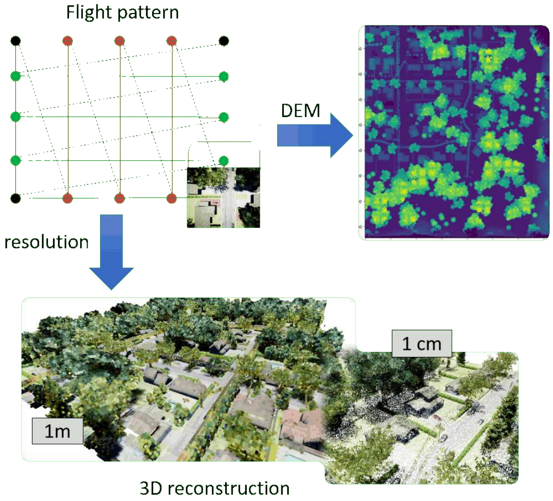


Figure 7. Example of a 3D point cloud and digital elevation model generated for a waffle flight UAV pattern in AirSim and UE.

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.

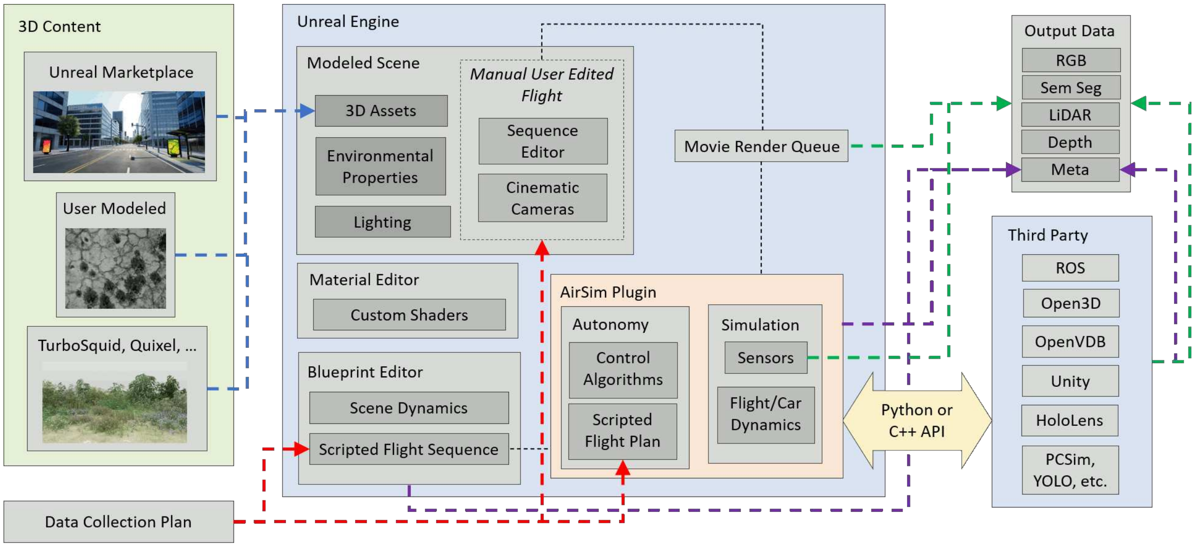


Figure 8. Example of overall predominantly open source photorealistic simulation framework discussed herein. Blue lines show scene modeling, red shows ways to carry out controlled data collection, green shows data generation, and purple shows corresponding metadata.

[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.

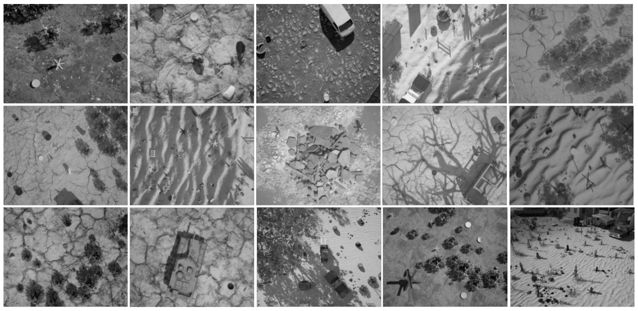


Figure 9. Example non-photorealistic UE scenes and imagery we built by mixing existing content to train an EHD DL detector [33].

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

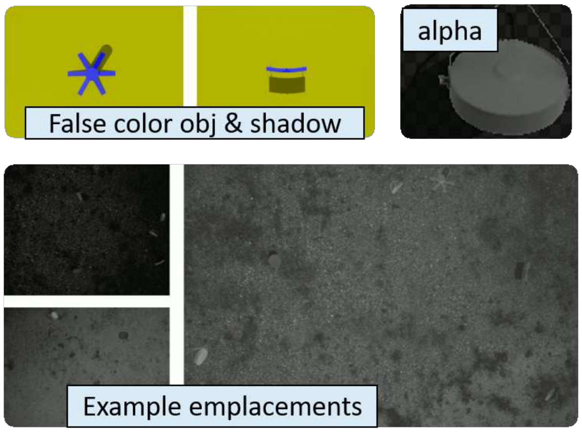


Figure 11. Example false color imagery and resultant alpha transparency EHD target templates from UE placed into real UAV drone imagery. Our insertion technique [33] combines UE and real UAV metadata to intelligently insert EHD templates.

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, 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.

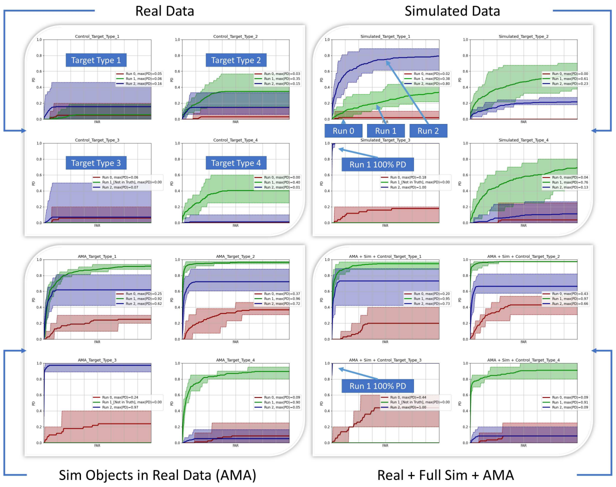


Figure 12. ROC results for YOLOv5 trained using real data, UE full simulated data only, real data with UE templates, and all combinations [33]. ROCs are divided into performance on four EH targets across three runs, and shading shows max, min, and average (the bold color lines) performance across a set 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.

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

[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.

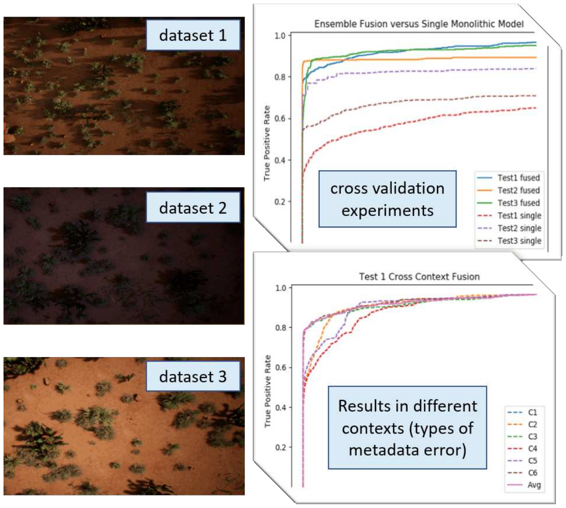


Figure 13. UE images and ROCs showing simulation predicted multi object detection algorithm fusion gain and degradation with respect to different types and amounts of metadata error. As in Figure 12, x-axis units are not reported due to sensitivity of EHD. The reader can still see and gauge relative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

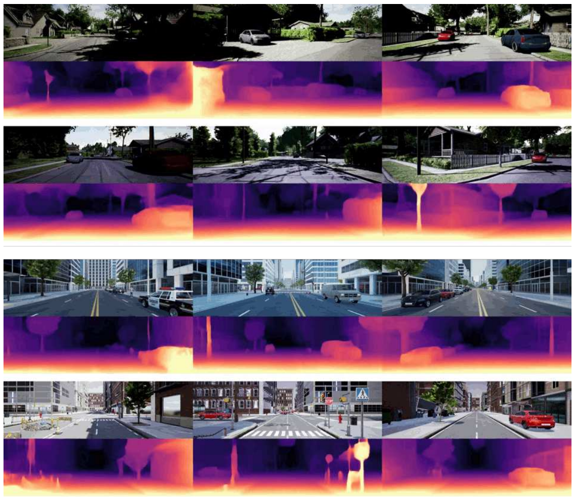


Figure 14. Monodepth2 output on UE data.

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 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

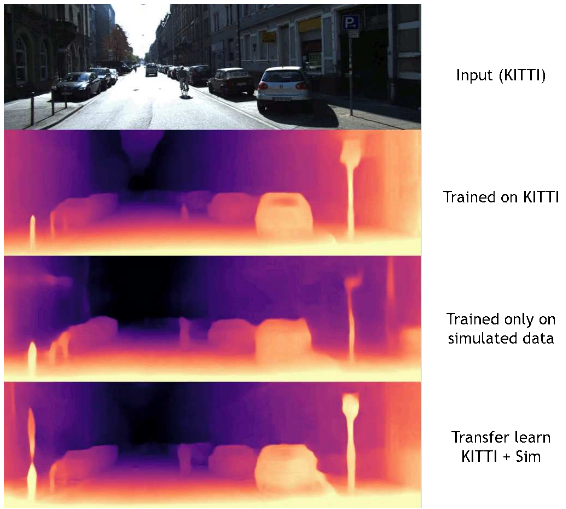


Figure 15. Monodepth2 on real, simulated, and combined data.

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 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.

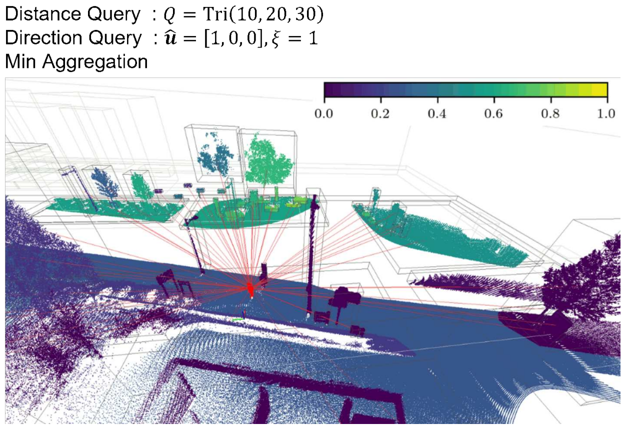

Distance Query : $Q = \mathrm{Tri}(10, 20, 30)$
Direction Query : $\hat{\mathbf{u}} = [1, 0, 0], \xi = 1$
Min Aggregation
0.0
0.2
0.4
0.6
0.8
1.0

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."

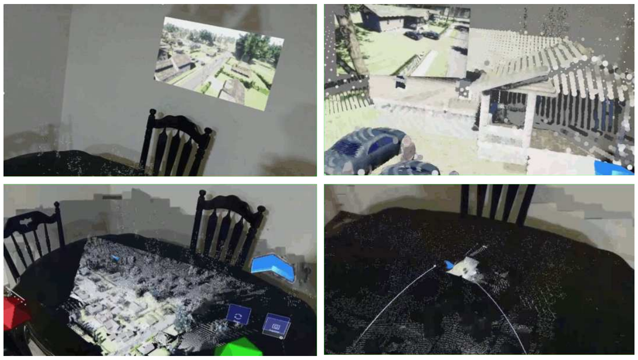


Figure 17. Example UE data in the Microsoft AR HoloLens. Real time video feed on a virtual AR monitor, real time point cloud streaming, and dynamic interrogation UAV data.

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, "VirtualWorlds 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.