Assuring Learning-Enabled Components in Small Unmanned Aircraft Systems

Veera V.R.M.K.R. Muvva*, Justin Bradley†, and Marilyn Wolf‡
University of Nebraska-Lincoln, Lincoln, NE 68588, USA

Taylor T. Johnson§
Vanderbilt University, Nashville, TN, 37240, USA

Aviation has a remarkable safety record ensured by strict processes, rules, certifications, and regulations, in which formal methods have played a role in large companies developing commercial aerospace vehicles and related cyber-physical systems (CPS). This has not been the case for small Unmanned Aircraft Systems (UAS) that are still largely unregulated, uncertified, and not fully integrated into the national airspace. However, emerging UAS missions interact closely with the environment and utilize learning-enabled components (LECs), such as neural networks (NNs) for many tasks. Applying formal methods in this context will enable improved safety and ease the immersion of UASs into the national airspace.

We develop UAS that interact closely with the environment, interact with human users, and require precise plans, navigation, and controllers. They also generally leverage LECs for perception and data collection. However, the impact of ML-based LECs on UAS performance is still an area of research. We have developed an advanced simulator incorporating ML-based perception in highly dynamic situations requiring advanced control strategies to study the impacts of ML-based perception on holistic UAS performance.

In other work, we have developed a WebGME-based software framework called the Assurance-based Learning-enabled CPS (ALC) toolchain for designing CPS that incorporate LECs, including the Neural Network Verification (NNV) formal verification tool. In this paper, we present two key developments: 1) a quantification of the impact of ML-based perception on holistic (physical and cyber) UAS performance, and 2) a discussion of challenges in applying these methods in this environment to guarantee UAS performance under various Neural Net (NN) strategies, executed at various computational rates, and with vehicles moving at various speeds. We demonstrate that vehicle dynamics, rate of perception execution, the design of the controller, and the design of the NN all contributed to total vehicle performance.

I. Introduction

Aviation has a remarkable safety record ensured by strict processes, rules, certifications, and regulations. In the development of large, commercial aircraft/spacecraft, formal methods have rightly been added to development, testing, and certification processes [1–3]. However, small Unmanned Aircraft Systems (UAS) or "drones" have largely evolved out of the hobbyist community*, and small UAS research labs at many universities. Each of these vehicles is a safety-critical cyber-physical system (CPS) *most likely* operating an open-source, unverified, autopilot† not beholden to Federal Aviation Administration (FAA) certification, verification, or much regulation.

Increasingly, however, small companies, researchers, and military projects are building drones capable of advanced missions in which they interact closely with the environment under complex scenarios, possibly using machine learning enabled components and complicated controllers [4–6]. These learning enabled components (LEC) can be part of data collection algorithms, perception, classifiers, navigation, or even in low-level control strategies. This puts LECs directly in safety-critical systems where errors can result in catastrophes. Understanding how LECs in safety-critical systems impact system performance, safety, and assurances is still ongoing research.

*Graduate student, Department of Computer Science and Engineering
†Assistant Professor, Department of Computer Science and Engineering, AIAA Senior Member
‡Professor, Department of Computer Science and Engineering, AIAA Senior Member
§Assistant Professor, Electrical Engineering and Computer Science
*Represented by the Academy of Model Aeronautics `https://www.modelaircraft.org/`
†For example, ArduPilot `https://ardupilot.org/`

In the first of potentially many more rules, the FAA recently proposed the "Remote Identification of Unmanned Aircraft Systems" rule[‡] to more tightly, and safely integrate UAS into the national airspace. A consequence of the rule is to force drone manufacturers to include remote ID equipment and algorithms into their products. As machine learning components spread to UAS, more certification and verification required of manufacturers, and more advanced, dangerous missions for UAS are proposed, the need for rigorous formal methods in this space will increase dramatically. To-date, however, *very few* UAS companies or researchers utilize formal methods in any way in their development. A common reason given is the cost of building such processes into the development cycle especially without federal regulations requiring it. This, unfortunately, misses an opportunity to allow formal methods to help provide a safer product or system as well as save money by lowering overall costs of development [7].

Through the DARPA Assured Autonomy program, we have developed a WebGME-based[§] software framework called the *Assurance-based Learning-enabled CPS (ALC) toolchain* for designing CPS that incorporate learning-enabled components (LECs) [8, 9]. The Neural Network Verification (NNV) software tool[¶] is incorporated within ALC, which has been applied to verify safety properties for several CPS that incorporate LECs such as neural networks [10–17]. Figure 2 shows an overview of the ALC toolchain, where activities encompass traditional model-based design (MBD) of CPS, but with a focus toward LECs and providing guarantees of their behavior at design-time and during system execution (runtime).

Fig. 1 Prescribed burn ignited by Ignis from Drone Amplified (image courtesy of Drone Amplified).

The ALC toolchain incorporates methods for training, testing, validation, verification, and related activities for LECs and their closed-loop interaction with CPS. Existing representative case studies and applications include verifying collision avoidance of an unmanned underwater vehicle (UUV), where a SegNet-based encoder-decoder performs semantic segmentation, and verifying collision avoidance in autonomous emergency braking systems (AEBS) between motor vehicles, where a feedforward neural network trained using reinforcement learning acts as a feedback controller [13].

[‡] https://www.federalregister.gov/documents/2019/12/31/2019-28100/remote-identification-of-unmanned-aircraft-systems

[§] https://webgme.org/

[¶] https://github.com/verivital/nnv

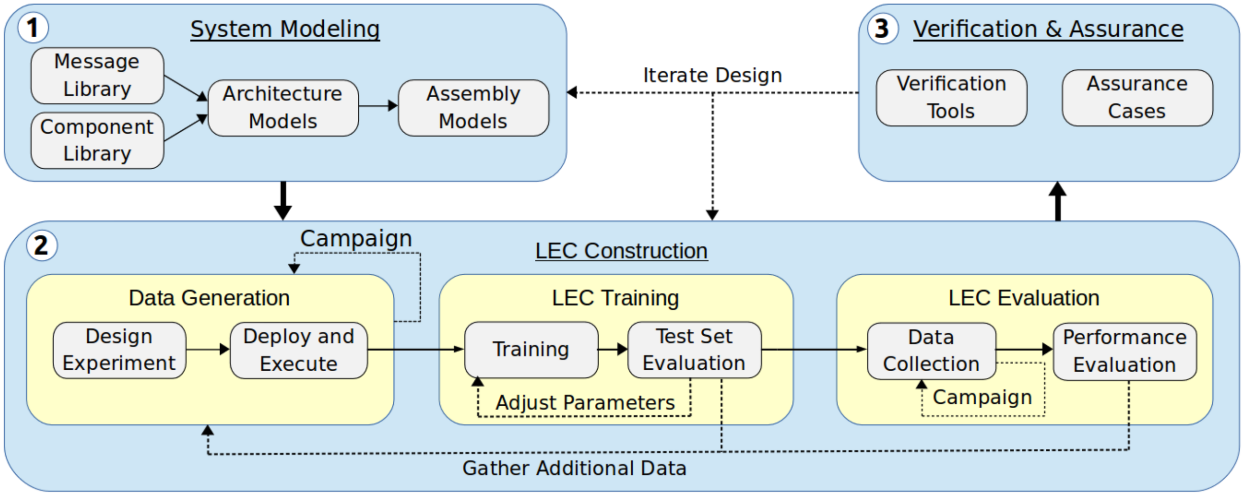

Fig. 2 ALC toolchain architecture [9].

In this paper we make two primary contributions. First, using a novel simulator we precisely quantify the impact of NN design, execution rate, vehicle dynamics, and controller design on UAS performance in UAS-to-UAS tracking scenarios – a dynamically complex scenario requiring advanced perception and control. Although work in advancing NNs is important, we demonstrate that to achieve reasonably good performance, the NN only need be designed to be "good enough" for the target application/scenario and controller. More advanced designs potentially waste critical resources at best, and at worst may require too many resources to be effective resulting in premature failure under taxing conditions. Second, we discuss the application of the ALC and NNV toolchain to a ML-based perception/control strategy, and show how this can improve and guide design while also providing assurances about performance.

II. ML-based Perception/Control UAS Simulator

We seek to directly control and measure the impact of NN design and execution, and controller design on holistic UAS performance. As a result, we seek to carefully control computation, dynamics, control, and time passage in an advanced simulator. Here we describe the simulator, experimental setup, and approach we use to quantify holistic perception/control performance.

A. Experimental Setup

The experimental setup is created by integrating ROS[‖] and AirSim [18]. The detailed information about these two is described here.

1. ROS

The Robot Operating System aka ROS is a collection of various software packages, which are intended for the software development of robots. For this experiment, we utilize ROS Melodic.

2. AirSim

AirSim is an open-source simulator for drones and cars[**]. It was designed and developed by Microsoft, built on the Unreal Engine, with the goal of assisting the community in researching deep learning, especially for computer vision and reinforcement learning domains. AirSim is capable of integrating into ROS through the AirSim ROS Wrapper. As a result, we use ROS to control the drones within AirSim for these experiments.

[‖] `http://wiki.ros.org/`
[**] `https://github.com/Microsoft/AirSim`

B. Approach

1. Experiment Scenario

The experiment scenario contains two drones, a target drone that can follow an arbitrary trajectory, and a chaser drone trying to follow its path. The chaser drone leverages an onboard camera to sense, detect, and ultimately control the chaser drone.

Target Drone – The target drone is the "target" or point of interest and in this experiment moves in an S-shaped trajectory; the significance of the S shape trajectory is that it has varying velocity and acceleration directions.

Chasing Drone – The chasing drone aims to sense, detect, and track the target drone, maintaining the same distance from it throughout. The target drone's commanded S-shape trajectory is unknown to the chaser drone. To accomplish this task, convolutional neural networks are used to detect the target drone, and PID (Proportional, Integral, Derivative) controllers are used to control the chasing drone based on the detected location of the target drone.

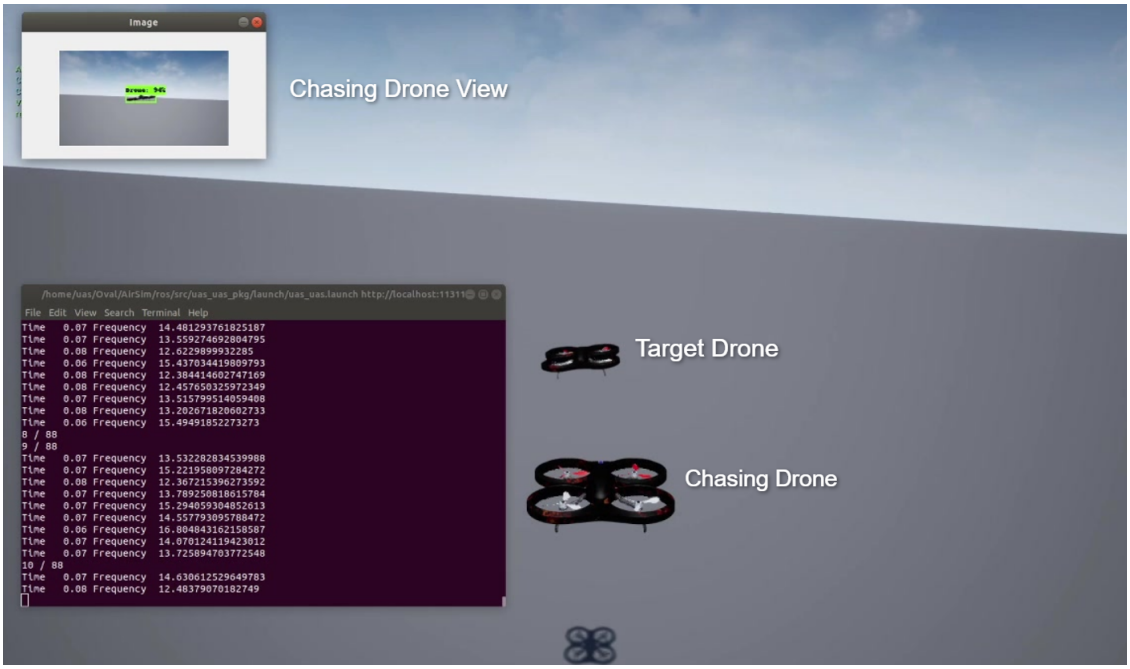

Fig. 3 Sample scene from simulation experiment.

A sample picture of the environment can be seen in Figure 3. The aim of the experiment is that the chasing drone should track and follow the target drone. This scenario represents taxing conditions where velocity and the environment can change rapidly, requiring a tight feedback loop between NN-based detection and PID-based controllers. The overall architecture of the system is shown in Figure 4.

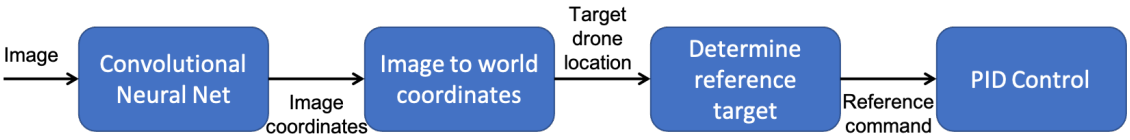

Fig. 4 Block diagram for chasing drone in UAS-UAS tracking scenario.

2. Drone Detection

Computational complexity vs. accuracy is a common trade-off in convolutional neural networks (CNNs). Some networks are very accurate but are computationally complex and hence may take longer to execute, while others execute faster but accuracy is sacrificed. To design our drone detection network we choose two different networks exemplifying

this tradeoff: Faster RCNN (Regions-based Convolutional Neural Networks) [19], and SSD (Single Shot multi-box Detector). Faster RCNN is highly accurate but is more computationally complex, and hence executes slower on our hardware resulting in a lower rate of target detection, thereby impacting the controller. In contrast, the SSD has lower detection accuracy, but execute much more quickly, matching the sampling rate of the controller.

Using the AirSim environment, an artificial dataset containing three hundred images is created. A drone is placed in various environments within the simulator; several images of it are collected during its flight. The dataset contains several variations of images such as various poses of drone, scenarios with occlusion, among others. All of these images are labeled manually by using a tool called LabelImg. These three hundred images are then used to train the neural networks.

The two networks Faster RCNN, and SSD are trained with the dataset mentioned above. These networks are trained on a machine that has 32 GB RAM, and 2 Nvidia RTX Graphics cards. It took thirty-two thousand epochs for Faster RCNN to converge and Sixty-three thousand epochs for SSD to converge. Faster RCNN is designed to execute at a rate of 3 Hz – providing 3 detections of the target drone every second. In contrast, SSD can execute at 30 Hz. Sample results of detection can be seen in Figure 5.

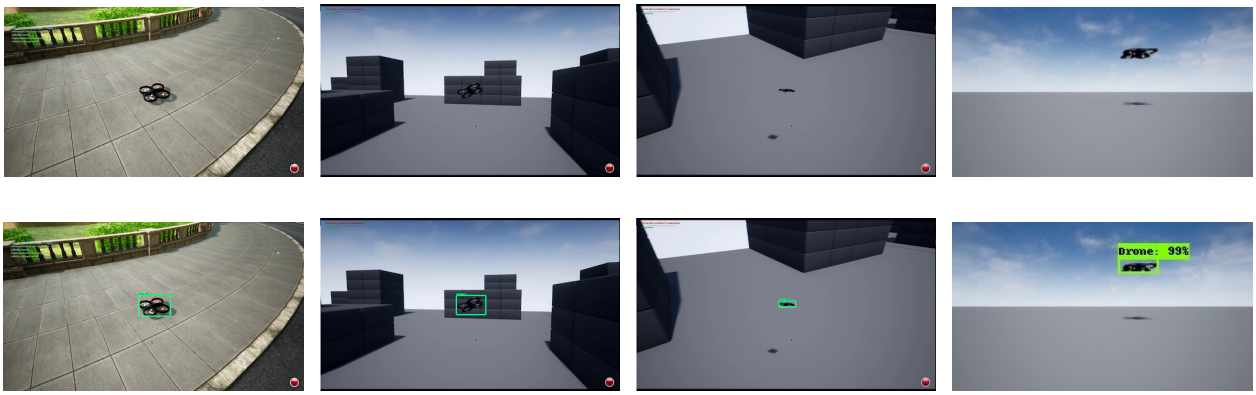

Fig. 5 Sample drone detections

3. Control

The NNs detect the target drone in the images which are taken from the chasing drone's camera producing image coordinates. The image coordinates are then converted to world coordinates, and the world coordinates are handed off to the PID controller of the chasing drone. First, the detection algorithm writes the area of the bounding box and image coordinates as (a, w, h), where

- a – Area of the detected bounding box
- w – Detected drone center along the width of the image
- h – Detected drone center along with the height of the image

In the AirSim world coordinate system, the coordinates are X (Front – Back axis), Y (Left-Right axis), and Z (Up – Down axis). The image coordinates (w, h) will be used to get the position of the target drone using

$$\begin{pmatrix} X_{TD} \\ Y_{TD} \\ Z_{TD} \end{pmatrix} = \begin{pmatrix} X_{CD} \\ Y_{CD} \\ Z_{CD} \end{pmatrix} + \begin{pmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & \alpha_z \end{pmatrix} \begin{pmatrix} a_c - a \\ w_c - a \\ h_c - a \end{pmatrix}. \tag{1}$$

The center of the image frame is w_c along the width and h_c along with the height. The center of the detection bounding box along the width is w, and along the height is h. When the target drone stays in front of the chasing drone without any left-right or top-down offsets, then w values tend towards w_c and h value tends towards h_c. The difference between w and w_c is proportional to the target drone's displacement changes along the Y-axis. Likewise, the difference of h and h_c is proportional to the displacement changes of the target drone along the Z-axis. Multiplying these by constant values leads to the actual displacement along the Y and Z axes.

Unlike the Y, Z axes, the X (Front-Back) axis is the lost dimension (i.e., depth) in the image. To address this an we leverage the known area of the bounding box of the drone in an image a known distance away. By observing the

area variation of the detected box, the algorithm estimates the distance (X) away of the target drone. For this approach, the initial area of the bounding box was detected and saved as a_c. Every time the detection happens, the area of the bounding will be compared to obtain the X-axis values of the drone. The difference between a_c and a is proportional to X-axis displacement.

The coordinate conversion is shown in the Equation 2. $\alpha_x, \alpha_y, \alpha_z$ are the camera constants along the X, Y, and Z axes. These values are calculated by running several experiments with the target and chasing drones with different trajectories and velocities and are 0.05, -6, -3.5, respectively. The coordinate conversion is done relative to the chasing drone. Hence adding the chasing drone's coordinates gives the absolute coordinates of the target drone.

A PID controller is designed for the chasing drone with the objective of following the target drone. Let $[X_{CD}, Y_{CD}, Z_{CD}]^T$ be the pose of the chasing drone, and $[X_{TD}, Y_{TD}, Z_{TD}]^T$ be the pose of the target drone. The target drone's pose is used as a reference point to calculate the errors and generate the required velocities using PID controllers. These error calculations and generating the velocities are

$$\begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix} = \begin{bmatrix} X_{CD} \\ Y_{CD} \\ Z_{CD} \end{bmatrix} - \begin{bmatrix} X_{TD} \\ Y_{TD} \\ Z_{TD} \end{bmatrix}, \tag{2}$$

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} (K_{px} * E_x) + (K_{ix} * \int E_x\, dt) + (K_{dx} * \frac{dE_x}{dt}) \\ (K_{py} * E_y) + (K_{iy} * \int E_y\, dt) + (K_{dy} * \frac{dE_y}{dt}) \\ (K_{pz} * E_z) + (K_{iz} * \int E_z\, dt) + (K_{dz} * \frac{dE_z}{dt}) \end{bmatrix}. \tag{3}$$

The generated velocities are used to give the control input to the chasing drone. The controller operates at a rate of 30 Hz. The performance of the chasing drone following the target drone can be seen in Figure 3.

C. Preliminary Results

We study the impacts of 3 key parameters: 1) NN design, where RCNN and SSD are placeholders for highly accurate, but complex NNs, and less accurate, but simple NNs respectively; 2) execution rate of the NNs; and 3) velocity of the UASs. We vary each of these parameters to examine the impact of each and report results.

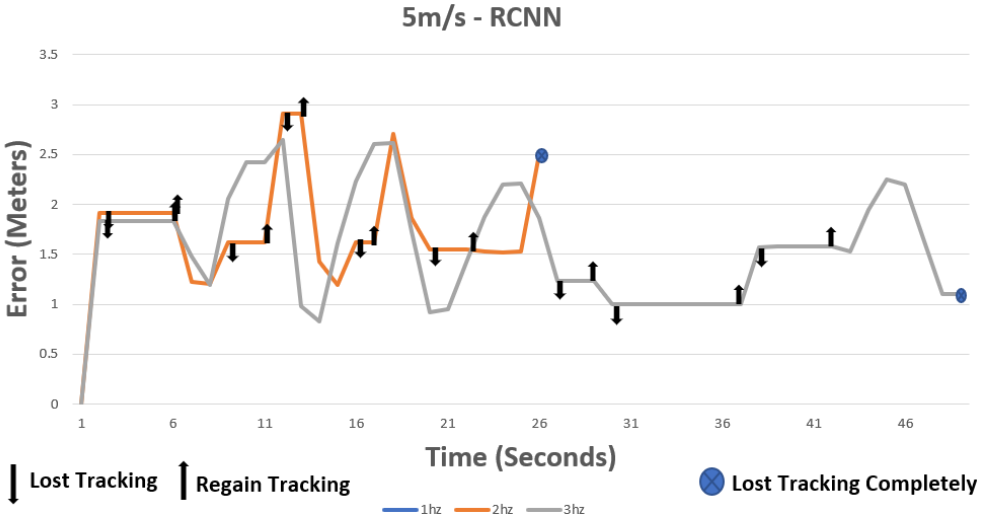

Fig. 6 RCNN Performance

Avg. Velocity	Frequency	Avg. Error	Basic Tracking
1 m/s	1 Hz	0.74 m	Yes
	2 Hz	0.37 m	Yes
	3 Hz	0.37 m	Yes
2 m/s	1 Hz	1.17 m	Yes
	2 Hz	0.71 m	Yes
	3 Hz	0.66 m	Yes
5 m/s	1 Hz	NA	No
	2 Hz	NA	No
	3 Hz	NA	No
10 m/s	1 Hz	NA	No
	2 Hz	NA	No
	3 Hz	NA	No

Table 1 RCNN Neural Net Performance

Figure 6 and Table 1 show results for the RCNN, and Figure 7 and Table 2 show results for the SSD NN. Figures 6 and 7 illustrate the control performance (based on tracking error) of the chasing UAS as the target UAS moves at an average of 5 m/s. Each line in the plots represents a different execution rate of the perception algorithm (NN). Although we ran this simulation at numerous velocities for the target UAS, this particular one (5 m/s) illustrates the limitations of the controller if the NN is not executed sufficiently quickly. In Figure 6 we see that when executing the NN at 1 Hz, there is no tracking, that is, the perception cannot provide information quickly enough for any tracking. Similarly, at 2 Hz, the chasing UAS loses track completely of the target at time 26 s. In contrast, when executed at 3 Hz, the NN is capable of providing sufficient detection of the target drone to allow the controller to track it. The black up and down arrows show when the NN loses track (down arrow) and then regains tracking (up arrow). This raises a significant point that an NN does not need to have perfect detection for a controller to track a target.

A slightly different story unfolds in Figure 7 which uses the SSD NN. The NN executed at 5 Hz does detect the target drone sufficiently often (despite losing and regaining tracking regularly) to allow the controller to track. However, performance is not great. In contrast, when executed at 10 Hz or above, performance is approximately the same (illustrating the limits of the controller). This illustrates a tradeoff between controller design, NN design, and execution rates and suggests a more optimal co-design of NN and controller would improve holistic performance.

Table 1 and 2 quantify the performance of each NN under the varying parameters. Of note, in Table 2 does not significantly improve its performance with more frequent execution beyond 5 Hz.

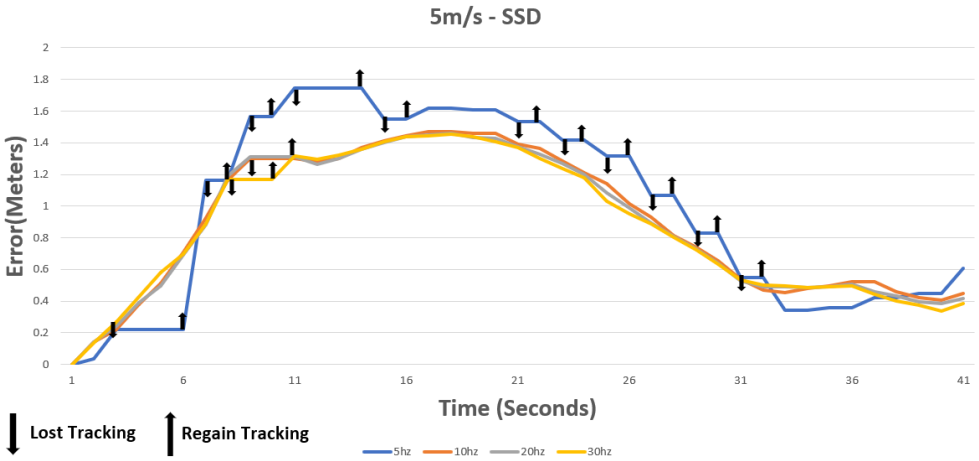

Fig. 7 SSD Performance

Ave. Velocity	Frequency	Avg. Error	Basic Tracking
1 m/s	5 Hz	0.269 m	Yes
	10 Hz	0.168 m	Yes
	20 Hz	0.155 m	Yes
	30 Hz	0.143 m	Yes
2 m/s	5 Hz	0.295 m	Yes
	10 Hz	0.227 m	Yes
	20 Hz	0.207 m	Yes
	30 Hz	0.202 m	Yes
5 m/s	5 Hz	0.978 m	Yes
	10 Hz	0.909 m	Yes
	20 Hz	0.897 m	Yes
	30 Hz	0.886 m	Yes
8 m/s	5 Hz	NA	No
	10 Hz	NA	No
	20 Hz	1.746 m	Yes
	30 Hz	1.674 m	Yes
10 m/s	5 Hz	NA	No
	10 Hz	NA	No
	20 Hz	NA	No
	30 Hz	1.803 m	Yes

Table 2 SSD Neural Net Performance

III. Challenges in Applying NNV and the ALC Toolchain

The overall ALC toolchain is illustrated in Figure 2. The toolchain incorporates methods for modeling closed-loop systems, performing data-driven training and learning for LEC construction and evaluation (e.g., testing and validation),

as well as verification with NNV [16]. In this work, we attempted to verify aspects of the UAS LECs, but experienced several major challenges, that we discuss next and are representative of the issues arising in the application particularly of formal verification for LECs.

First, most existing work in neural network verification has focused on image classification and related classification problems (such as determining the control advisory in ACAS-Xu). In our UAS architecture, instead of classification, object detection and localization are used, which produce bounding boxes in the output space. To the best of our knowledge, no existing neural network verification methods have been applied to this problem, and defining robustness specifications in these scenarios is an avenue for future work.

Second, specifically for the chosen neural network architectures (Faster RCNN and SSD), the networks as created use pre-trained weights that make them quite large, larger likely than the VGG16/VGG19 ImageNet classifiers previously analyzed with NNV [15]. This aspect motivates coordination in the creation of LECs between LEC designers and verification engineers, such as that the LECs have been "designed for verification," as commonly arises in other domains (digital logic, software, etc.). Primary design-for-verification concerns with LECs currently include (a) the input dimensionality, (b) the specifications, (c) the types of layers used, and (d) the goal of the verification activities. The input dimensionality is essential, because if a smaller input space may be used, such as lower resolution images, the overall network architecture will be smaller, yielding fewer parameters, numbers of challenging activation functions and layer types (e.g., ReLUs), and related complexities. In this instance, as object detection and localization have not significantly been considered for robustness, defining the specifications would involve determining what is important for robustness in this scenario. Several layer types, especially ReLUs and max-pooling layers, are computationally demanding to analyze (NP-complete). For object detection and localization, support specifically for the layers that lift from the latent space to the output space would also need to be supported for verification. Lastly, a critical design-for-verification concern is what the goal of the verification activities would be: in this case, one could analyze robustness to perturbations, but perhaps for this scenario, other closed-loop or system-level specifications are of greater practical benefit.

Third, considering the closed-loop coordination of perception LECs, like Faster RCNN and SSD, with controllers and a physical plant, has also not been considered from a formal verification standpoint. Primary applications thus far that consider LECs in closed-loop with plant models utilize LECs as controllers, such as the classes of systems considered in the ARCH-COMP AINNCS category [10, 11, 20]. In these contexts, the LEC inputs are typically low-dimensional, on the order of the size of the number of state variables in the plant model (or the observable states of the plant). Considering perception LECs in closed-loop is significantly more challenging. For instance, when considering low-dimensional LEC inputs that represent the plant state, one does not need to consider the impact of environmental configurations that arise in perception, such as obstacles, as the low-dimensional inputs have typically abstracted these. One avenue to address this is to consider an architecture where the LEC output is augmented so that it is directly used within the control strategy, instead of having various post-processing steps, moving closer to end-to-end style learning. Of course, this introduces a tradeoff, as this makes the LEC architecture more complex. Even if addressed, another limitation for closed-loop verification is that is must be done for specific scenarios, encompassing these environmental aspects.

We aim to address these issues in future work, by considering standard architectures used for object detection and localization, such as Faster RCNN and SSD, as used in this work, and adding support for these architectures, necessary layers, and formalization of robustness within NNV. Once these architectures are incorporated, another direction for future work is to consider these object detection and localization components within closed-loop scenarios. A last direction for future work is to utilize the full ALC toolchain to perform the training and validation, which currently were done directly with TensorFlow and AirSim, which would make the application of assurance methods incorporated within ALC easier. These methods go beyond the discussed verification methods like NNV, and would include runtime assurance monitoring methods that can be generated by the ALC toolchain, as well as methods for out-of-distribution detection, runtime verification [21, 22], and beyond.

IV. Conclusion

Machine Learning is being deployed in Learning-enabled components in a variety of safety-critical systems, but a characterization of their impacts on UAS, and the application of formal methods for guaranteed assurance in UAS is still lacking. In this paper, we report on a novel simulation that we use to characterize the impacts of LECs on UAS performance in a UAS-to-UAS tracking scenario. We then discuss challenges in applying our ALC toolchain to the ML-based perception/control system to provide formal guarantees of performance.

Much work is still needed to fully integrate formal method strategies into the small UAS community. Of primary

interest is the need to make toolchains easy to use, accessible, and worthwhile for small business, researchers, and startups to use. Emerging FAA rules and regulations may force this issue, and the research community can help aid the transition to the emerging rules by providing available toolsets and methods to meet demands.

References

[1] Graves, H., and Bijan, Y., "Using formal methods with SysML in aerospace design and engineering," *Annals of Mathematics and Artificial Intelligence*, Vol. 63, No. 1, 2011, pp. 53–102.

[2] Ameur, Y. A., Boniol, F., and Wiels, V., "Toward a wider use of formal methods for aerospace systems design and verification," *International Journal on Software Tools for Technology Transfer*, Vol. 12, No. 1, 2010, pp. 1–7.

[3] Feron, E., "Formal methods for aerospace applications," *2012 Formal Methods in Computer-Aided Design (FMCAD)*, IEEE, 2012, pp. 3–3.

[4] Plowcha, A., Sun, Y., Detweiler, C., and Bradley, J., "Predicting Digging Success for Unmanned Aircraft System Sensor Emplacement," *2018 International Symposium on Experimental Robotics*, 2018 International Symposium on Experimental Robotics, Buenos Aires, Argentina, 2018.

[5] Shankar, A., Elbaum, S., and Detweiler, C., "Towards Aerial Recovery of Parachute-Deployed Payloads," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 4700–4707.

[6] Luo, C., Nightingale, J., Asemota, E., and Grecos, C., "A UAV-cloud system for disaster sensing applications," *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, IEEE, 2015, pp. 1–5.

[7] Cofer, D., "Formal methods in the aerospace industry: follow the money," *International Conference on Formal Engineering Methods*, Springer, 2012, pp. 2–3.

[8] Hartsell, C., Mahadevan, N., Ramakrishna, S., Dubey, A., Bapty, T., Johnson, T., Koutsoukos, X., Sztipanovits, J., and Karsai, G., "Model-based Design for CPS with Learning-enabled Components," *Proceedings of the Workshop on Design Automation for CPS and IoT*, ACM, New York, NY, USA, 2019, pp. 1–9. https://doi.org/10.1145/3313151.3313166.

[9] Hartsell, C., Mahadevan, N., Ramakrishna, S., Dubey, A., Bapty, T., Johnson, T., Koutsoukos, X., Sztipanovits, J., and Karsai, G., "CPS Design with Learning-Enabled Components: A Case Study," *Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP'19)*, Association for Computing Machinery, New York, NY, USA, 2019, p. 57–63. https://doi.org/10.1145/3339985.3358491, URL https://doi.org/10.1145/3339985.3358491.

[10] Lopez, D. M., Musau, P., Tran, H.-D., Dutta, S., Carpenter, T. J., Ivanov, R., and Johnson, T. T., "ARCH-COMP19 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants," *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, EPiC Series in Computing, Vol. 61, edited by G. Frehse and M. Althoff, EasyChair, 2019, pp. 103–119. https://doi.org/10.29007/rgv8, URL https://easychair.org/publications/paper/BFKs.

[11] Johnson, T. T., Lopez, D. M., Musau, P., Tran, H.-D., Botoeva, E., Leofante, F., Maleki, A., Sidrane, C., Fan, J., and Huang, C., "ARCH-COMP20 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants," *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, EPiC Series in Computing, Vol. 74, edited by G. Frehse and M. Althoff, EasyChair, 2020, pp. 107–139. https://doi.org/10.29007/9xgv.

[12] Tran, H.-D., Musau, P., Lopez, D. M., Yang, X., Nguyen, L. V., Xiang, W., and Johnson, T. T., "Parallelizable Reachability Analysis Algorithms for Feed-forward Neural Networks," *Proceedings of the 7th International Workshop on Formal Methods in Software Engineering (FormaliSE'19)*, IEEE Press, Piscataway, NJ, USA, 2019, pp. 31–40. https://doi.org/10.1109/FormaliSE.2019.00012.

[13] Tran, H.-D., Cei, F., Lopez, D. M., Johnson, T. T., and Koutsoukos, X., "Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control," *ACM SIGBED International Conference on Embedded Software (EMSOFT'19)*, ACM, 2019.

[14] Tran, H.-D., Musau, P., Lopez, D. M., Yang, X., Nguyen, L. V., Xiang, W., and Johnson, T. T., "Star-Based Reachability Analysis for Deep Neural Networks," *23rd International Symposium on Formal Methods (FM'19)*, Springer International Publishing, 2019.

[15] Tran, H.-D., Bak, S., Xiang, W., and Johnson, T. T., "Verification of Deep Convolutional Neural Networks Using ImageStars," *Computer Aided Verification*, edited by S. K. Lahiri and C. Wang, Springer International Publishing, Cham, 2020, pp. 18–42.

[16] Tran, H.-D., Yang, X., Manzanas Lopez, D., Musau, P., Nguyen, L. V., Xiang, W., Bak, S., and Johnson, T. T., "NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems," *Computer Aided Verification*, edited by S. K. Lahiri and C. Wang, Springer International Publishing, Cham, 2020, pp. 3–17.

[17] Bak, S., Tran, H.-D., Hobbs, K., and Johnson, T. T., "Improved Geometric Path Enumeration for Verifying ReLU Neural Networks," *Computer Aided Verification*, edited by S. K. Lahiri and C. Wang, Springer International Publishing, Cham, 2020, pp. 66–96.

[18] Shah, S., Dey, D., Lovett, C., and Kapoor, A., "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and service robotics*, Springer, 2018, pp. 621–635.

[19] Ren, S., He, K., Girshick, R., and Sun, J., "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, 2015, pp. 91–99.

[20] Lopez, D. M., Musau, P., Tran, H.-D., and Johnson, T. T., "Verification of Closed-loop Systems with Neural Network Controllers," *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, EPiC Series in Computing, Vol. 61, edited by G. Frehse and M. Althoff, EasyChair, 2019, pp. 201–210. https://doi.org/10.29007/btv1, URL https://easychair.org/publications/paper/ZmnC.

[21] Bak, S., Johnson, T. T., Caccamo, M., and Sha, L., "Real-Time Reachability for Verified Simplex Design," *IEEE Real-Time Systems Symposium (RTSS)*, IEEE Computer Society, Rome, Italy, 2014.

[22] Johnson, T. T., Bak, S., Caccamo, M., and Sha, L., "Real-Time Reachability for Verified Simplex Design," *ACM Transactions on Embedded Computing Systems (TECS)*, 2016.