AEGIS: Autonomous Entity Global Intelligence System for Urban Air Mobility

Aditya Das[1], Kristina Marotta[2] and Husni Idris[3]
NASA Ames Research Center, Moffett Field, CA, 94035, USA

***Abstract* –This paper presents a global intelligence system that synthesizes aerial vehicles' real-time physical data, planned actions, and historical behavior into engineered data frames representing the collective state of the airspace and suitable for efficient machine learning consumption. These data frames are then learnt by a deep neural net to build a prediction model that estimates the expected evolution path of the current state, thereby identifying potential future conflicts. This approach lends itself to an automated early warning system that the aerial vehicles can implement onboard with a suitable edge computing module more efficiently and effectively than non-AI methods, and eventually take preventive or corrective measures towards self/collaborative resolution of the issues. Contrary to a centralized early warning system where all vehicles' task-space eventually converges to a global optimum state, the presented distributed global intelligence system brings in a balance between local utility functions of each vehicle and the global operating framework. This contributes to effectively handle the potential massive scaling in urban air mobility in the near future.**

I. Introduction

Commercial air mobility for transportation of people and packages in urban environments is both strategically as well as tactically challenging. Heterogeneity of aerial vehicles coupled with their expected high density in highly fragmented airspace poses a significant challenge in implementing a coordinated urban air mobility strategy. Furthermore, varying individual business objectives and priorities of different aerial vehicle operators, make the tactical management of Urban Air Mobility (UAM) operations in tight and shared airspace difficult, specifically in a centralized manner as it can be quickly overwhelmed by the demand for decision making. On the other hand, a distributed management approach for the airspace usage provides scalability of services whereby these diverse aerial vehicles participate in providing services; however, it adds to the computational needs for vehicle decision making. Leveraging onboard artificial intelligence (AI) capabilities, promises a faster and commercially viable self/collaborative airspace management, as long as it does not compromise airspace safety.

With over seven million unmanned aircraft systems (UAS) sales projected over the next five years, the national air space is destined to get more crowded and more challenging to manage. As it is almost impractical to foresee this fully scaled and widely heterogeneous air traffic being entirely human controlled or centrally managed, alternative approaches involving some level of autonomy and distributiveness are warranted. This paper presents such an approach for future air mobility, wherein a distributed network of vehicle-based artificial intelligence units are employed to assure tactical separation among cooperative and uncooperative air traffic.

The work presented in this paper is inspired from National Aeronautics and Space Administration (NASA)'s rich aviation technology foundation, developed over the years in collaboration with the Federal Aviation Agency (FAA), to make utilization of the national airspace system (NAS) safer, structured and efficient. Among the many decision-making tasks in the conventional air traffic management (ATM), separation assurance and runway access are probably the two most critical and frequent tasks managed by the air traffic controllers at both strategic as well as tactical levels. This has led to several research and development efforts, including the Traffic Alert and Collision Avoidance System (TCAS) [1], which is an FAA implementation of International Civil Aviation Organization (ICAO)'s airborne collision avoidance system (ACAS) for short range systems (often mentioned in conjunction with the airborne

[1] Research Aerospace Engineer, Flight Dynamics, Trajectory and Controls Branch, Aviation Systems Division.
[2] Pathway Intern, Flight Dynamics, Trajectory and Controls Branch, Aviation Systems Division.
[3] Research Aerospace Engineer, Aviation Systems Division.

separation assurance system (ASAS) for long range systems). ACAS serves as a non-centralized & automated conflict detection and avoidance system, operating independently of ground-based equipment. While older ACAS systems relied exclusively on an interrogation mechanism using on-vehicle transponders and beacons followed by issuance of potential threat advisory based on fixed set of rules, the newer ACAS X [2] system implements its alerting logic based on a numeric lookup table optimized with respect to a probabilistic model of the airspace and a set of safety and operational considerations. Furthermore, ACAS X uses multiple surveillance sources such as sensors and global positioning system (GPS) measurements, while maintaining full compatibility with newer generations of TCAS. Kuchar & Yang have reported a survey of multiple contemporary conflict detection & resolution (CDR) methods [3].

In ATM, which is essentially a centralized management model, ACAS X (or TCAS) is expected to serve primarily as an emergency event handling tool with short and clear directions for the pilots to take preventive course of action. In this regard, the distributed control advisories issued by the ACAS X system are purely for physical conflict avoidance and not for any other business utility function optimization. Furthermore, such advisories are only issued on a case by case basis to resolve specific conflicts and aircraft return to centralized control after the conflict resolution. A variation of ACAS X, called ACAS Xu, has been studied [4] for unmanned aircrafts. ACAS Xu brings in certain special consideration such as handling of non-cooperative sensors and use of horizontal resolutions, especially in view of electronic Vertical Take-Off and Landing (eVTOL) vehicles [5]. Specific to UAS, Skowron *et al.* have discussed recent advances in sense and avoid methods for small UAS [6]. While these technological advancements are designed to ensure UAM operational safety, the presented work offers a supplementary edge-mounted AI technique for UAM optimization based on UAS operators' business utility functions within the operational safety boundaries, thereby, paving the pathway for UAM democratization, especially benefitting small & new market entrants. This paper utilizes conflict detection as one of the instantiations of such an overarching vision of pushing the smartness to the edge.

The rest of the paper is organized as follows: Section II discusses the research focus followed by the proposed approach in section III that presents a graphical data aggregation and AI modelling method for collection behavior assessment of UASs. In section IV the concept of operation scenarios and corresponding metrics are introduced. Section V presents a custom simulator developed for UASs collective behavior simulation. Research findings are summarized in section VI. Finally, section VII concludes the paper with a discussion on the future directions.

II. Research Focus: Heterogeneous Multi-Entity Collaboration for UAM

Multiple market studies, concepts of operations (ConOps), and other research reports have collectively portrayed a UAM vision that is highly diverse in nature, comprising of a wide variety of automated flying entities (or agents), their operators with different utility functions driven by their business models, and a whole range of stakeholders such as operation support businesses, cities, emergency service providers, regulatory agencies, general public and so on. At present, there is no established infrastructure to enable and safely manage the widespread use of low-altitude airspace and UAS operations, regardless of the type and operation of UAS. Accordingly the FAA, NASA and several industry partners are collaborating on the conceptualization and development of a UTM system.

Ref. [7] reflects the UTM architecture that introduces a layered/hierarchical framework for urban air traffic management with entities such as UAS service suppliers (USS), supplemental data service providers (SDSP) and UAS operators, working in coordination with the UASs for operational decision making. The industry components, including the UAS operators, USSs, and SDSPs, are expected to coordinate with each other for UAM operations. A critical capability that ensures overall safety of the airspace shared by conventional air vehicles such as airplanes, and helicopters, and UAS agents is the Flight Information Management System (FIMS). FIMS is a central, cloud-based data exchange gateway among ATM, UTM, public service, and regulatory entities. In the UTM architecture, connections to the FIMS are made by the USSs authorized for functionality, quality of service, and reliability.

Logically, FIMS supports UTM functionalities such as registration and licensing, assurance of equitable use of airspace including information sharing about restrictions and notice to airmen (NOTAMs), identification and authentication of data/service providers, overall airspace health and status monitoring, and most importantly potential de-confliction of flight plans coming from different USSs servicing the same airspace. The deployment model for these functions, however, determines the efficacy of UTM. For example, taking tactical deconfliction into account in a centralized model may offer better control, but may also become a critical bottleneck as operation volume scales up. On the other hand, a federated model can become complex due to standardization requirements.

A more practical solution, as considered by the UAM community, is a blended architecture using elements of centralized and federated models. In such a case, certain critical operations can be centralized while leaving the others to a federated model where the USSs can prosper in a fair market competition to deliver added value to their customers, while working within the Government defined regulations. Additionally, the centralized model can always serve as the fall-back plan in situations where the federated model could not resolve a particular issue.

Assuming that contemporary and future UASs already come equipped with sophisticated onboard processing capability, or alternatively they can be augmented with auxiliary processing modules onboard, it is possible that they can take part in tactical decision making, thereby extending the earlier federated model that works at the USS level into a true distributed model that works on the edge at the UAS level, as shown in Fig. 1.

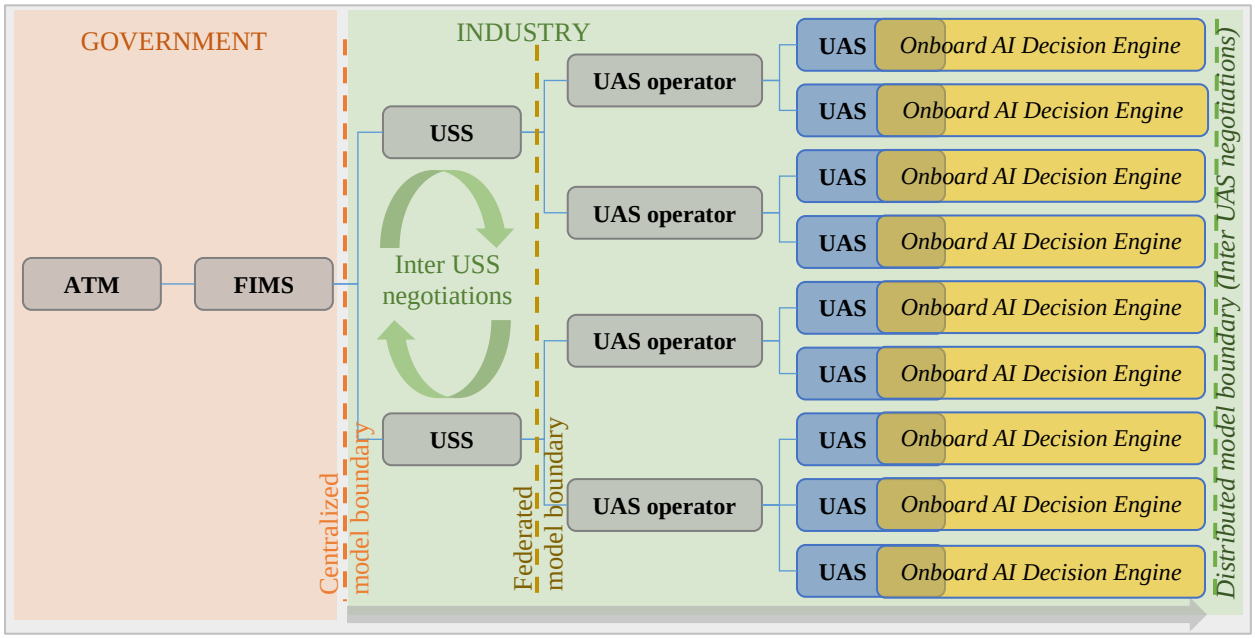

Fig. 1 A distributed model for UTM utilizing edge computing capability onboard UASs

Note that the UTM architecture described in Ref [7] includes the concept of vehicle-to-vehicle (V2V) communication capability that enables a UAS to broadcast relevant information (e.g., position) to nearby vehicles with cooperative equipment, allowing for impacted stakeholders (e.g., nearby operators in four-dimensional proximity to the UAS) to gain awareness of the situation and respond accordingly. In general, airborne separation is a critical aspect of air traffic management that has been studied by NASA and others, where vehicle technology was designed to predict and resolve conflicts based on ADS-B shared intent and similar information [8] [9] [10] [11] [12] [13].

The work presented in this paper envisions enhancing this capability to not only share information to gain collective situational awareness but also reason upon this information to build knowledge about the projection of the situation into the future. Such knowledge coupled with a suitable onboard AI decision engine, will enable the UASs to implement remedial measures either simply through proactive action or active inter-UAS negotiations. There are situations when USS-USS negotiation may be slow, insecure, or ineffective in capturing vehicle utilities, and better aided with direct UAS-UAS negotiation with minimal USS role for approval. Note that these USS's are envisioned as highly automated bots, so the latencies are expected to be much less to begin with, in comparison to human-centric operations. Nevertheless, UAS level decision making alleviates most of the risks including communication network congestion, latency, and brownout, etc.

While the aforementioned distributed model for UTM offers a more robust and low latency decision making framework that accommodates the utility functions of the UASs and their operators, the implementation of such a framework requires that the UASs have comprehensive knowledge about other UASs in addition to knowing their own individual states. Given the limited computing capability onboard the UASs, it is not realistic for them to process each element of such a big volume of data. Therefore, a custom *data-to-intent* synthesis approach, that allows rapid thickening of data into more relevant actionable information, is proposed in the following section.

III. Data Aggregation for AI-based Cooperative Control

Data sharing, aggregation, and interpretation are critical components of multi-agent, multi-nodal decision-making framework. A multitude of research approaches have been investigated in the past for decision making in 4D (three positional dimensions and one time dimension) trajectory management in ATC. Some examples are: trajectory based air traffic control and mixed operations with airborne self-separation [14], separation assurance, arrival sequencing and weather-cell avoidance solver called *Autoresolver* [15], 4D trajectory planning human-machine interface for flight

decks [16] [17], distributed trajectory flexibility preservation [18], scenario complexity metric formulation [19], and so on. Adding to this rich technological foundation**,** this paper presents a holistic approach of synthesizing data into *graphical data frames* that carry information regarding the agents' behavior and intent, in conjunction with the physical data from the agents. The motivation for such data aggregation method comes from the opportunity to utilize cutting edge deep learning techniques for rapid assessment and classification of conflicts in the 4D traffic scenarios.

Air traffic controllers today use a wide range of visual aids to monitor and manage the airspace (and ground facilities such as airports) traffic. These visual aids are designed to supplement the human intelligence with data, in an easily interpretable manner, so that the human controller can take timely decisions to ensure safety and efficiency. In the case of the distributed UTM model, where the AI does the decision making, the data ingestion framework needs to be optimized for the AI. For autonomous entities, such as UASs, the data aggregation needs to be AI-centric where large sets of information, both real-time and historical, can be quickly synthesized and processed onboard the vehicles.

This AI-centric data aggregation can be seen as analogous to the way the human brain aggregates information for quick decision making, by using relational models reinforced by selected critical features in the data and making decisions based on the best made predictions from often incomplete data. Classical cooperative control involving multiple agents in a connected network generally assume that the agents have complete information about their neighbors, and if not, then a fairly deterministic or fixed trust vector is used to complete the analytical model for the cooperative control implementation. While such analytical control frameworks provide a sound foundation for distributed control in UAM scenarios, to make it commercially usable additional considerations need to be incorporated in the UAM cooperative control framework. Such considerations entail the business utility functions as well as agent-to-agent negotiation with a time-based reward system that incentivizes integrative conflict resolution.

The following sub-sections introduce the foundation for agent-to-agent cooperative control with the additional considerations for real-world functionalization (in sub-section A), a graphical approach to data aggregation (in sub-section B), and a deep neural net based scenario learning and assessment framework (in sub-section C). This paper specifically focuses on the data aggregation and deep neural net-based data ingestion, learning, and inference aspects. The deep learning-based decision engine for distributed conflict resolution via pro-active measures and active negotiations, built on the fundamental idea and modification for cooperative control framework as mentioned in sub-section A below, will be discussed in a separate paper.

A. Foundation of Cooperative Control Modeling for Multi-Agent Scenarios

For a set of *N* agents in a particular airspace with identical dynamics, the state space model for each agent can be represented as Eq. (1), $where\ x_i \in \mathbb{R}^n\ is\ the\ state, u_i \in \mathbb{R}^m\ is\ the\ input, y_i \in \mathbb{R}^p\ is\ the\ measured\ output$

$$\dot{x_\iota} = Ax_i + Bu_i, y_i = Cx_i, \forall i \in \mathcal{N}, \tag{1}$$

In the general linear system in Eq. (1), it is assumed that the coefficients *A*, *B*, and *C* are stable and detectable.

In a leader-based cooperative control, the leader dynamics can be represented with $\dot{x_0} = Ax_0, y_0 = Cx_0$, which generates the desired target trajectory and acts as a command generator, observable from a subset of agent nodes in graph $\mathcal{G}$. In leaderless model, such as the one in discussion in this paper, $g_i = 0, \forall i \in \mathcal{N}$, we require that all eigenvalues of *A* are in the closed left-half complex plane [20], in other words all agents operations exhibit asymptotic stability in the linearized system with (A_i, B_i) stabilizable. $A = [a_{ij}]$ is the adjacency matrix, where a_{ij} is the edge weight such that $a_{ij} > 0\ if\ nodes\ (v_j, v_i) \in edge\ E\ and\ a_{ij} = 0\ otherwise$. Assuming that A is known for all agents, the distributed consensus algorithm can be represented as:

$$u_i = \sum_{j \epsilon N_i} a_{ij}(x_j - x_i) \tag{2}$$

Eq. (2) is also referred to as local voting protocol. Here it is assumed that each agent can obtain information about the state only of itself and its in-neighbors in N_i. Defining the weighted in-degree of node v_i as the *i*-th row sum of *A*, i.e.

$$d_i = \sum_{j=1}^{N} a_{ij} \tag{3}$$

Eq. (2) can be rewritten as:

$$u_i = -x_i \sum_{j \epsilon N_i} a_{ij} + \sum_{j \epsilon N_i} a_{ij} x_j = -d_i x_i + [a_{i1} \quad \dots \quad a_{iN}] \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \tag{4}$$

Using block matrix notations,

$$u = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}, D = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_N \end{bmatrix}, A = [a_{ij}] \tag{5}$$

the overall control input for the network can be represented as:

$$u = -Dx + Ax = -(D - A)x = -Lx \tag{6}$$

where *L* is the graph Laplacian matrix. As the UAM agents will operate under a set of guidance set forth by the FAA and delegated through the USSs, it can be assumed that these agents will most likely demonstrate uniform behavior and responses to the same events. Under this assumption, the overall system dynamics can be approximated with single-integrator dynamics, i.e. $\dot{x_\iota}(t) = u_i(t)$. The closed-loop dynamics, therefore, are given as:

$$\dot{x} = -Lx \tag{7}$$

The system matrix is –*L* and hence has eigenvalues in the left-half plane. Details regarding the above multi-agent cooperative control method can be found in [21].

The takeaway from Eq. (7) is that for such cooperative control to work effectively, states of all agents need to be collectively known, and there must be no malicious or non-cooperating agents. In the real world, however, the states of an agent may not be known with absolute certainty and also some rouge agents might be present. Therefore, the distributed cooperative control method needs to incorporate inter-agent trust. Defining $\xi_{ij}\epsilon[-1,1]$ as the trust that node *i* has for node *j*, the trust propagation can be represented as:

$$\dot{\xi}_i = \sum_{j\epsilon N_i} a_{ij}(\xi_j - \xi_i) \tag{8}$$

Where $(\xi_j - \xi_i)$ represents the difference of opinion between neighboring agents. With this trust dynamics the state dynamics can be written as:

$$\dot{x}_i = \sum_{j\epsilon N_i} \xi_{ij}.\, a_{ij}(x_j - x_i) \tag{9}$$

Note that the trust can change over time, for example it can become stronger with recognizing a pattern in agent behavior over time, or it can degrade if the agent behavior is more erratic. In Eq. (9), the term $\xi_{ij}.\, a_{ij}$, therefore, represents time-varying edge weight. The distributed control problem then essentially becomes the problem of figuring out and optimizing these time-varying edge weights. Linearization of trust, however, is generally difficult as this is highly volatile and sensitive to a lot of variables and unanticipated events. Studies have been conducted to model herd behavior in panic situations [22], but such studies are done in the context of humans, and their portability to autonomous machines such as UASs is not absolute. On the other hand, Boyd [23] showed that if some edge weights are negative, convergence speed is faster. Knowing that a_{ij} is a positive number, this means that if some trusts are negative or in other words there is distrust in the network then the consensus is reached sooner.

UAM scenarios, staged by multiple operators with different utility functions and in market competition, often preclude the availability of complete contextual insight into the individual agents' intent unless explicitly shared. The collective state and intent of the agents can thus be broadly categorized into public and private sets of information. Furthermore, the private set of information can be approximated on the basis of the trust, which in turn can be formulated based on the knowledge about past behavior of the agent, preferably via patterns. Additionally, the agent behavior can be moderated; that is, the trust can be engineered, through a suitable reward system for exhibiting desired behavior. With these additional considerations for distributed cooperative control in UAM scenarios, the state update function can be expressed as:

$$\dot{x_\iota} = f\left(x_{ip}, x_{iq}, \begin{bmatrix} x_{1p} & \delta_{1p} & r(t)_1 \\ \vdots & \vdots & \vdots \\ x_{np} & \delta_{np} & r(t)_n \end{bmatrix}, c_0, \sigma_0, r_0, t\right) \tag{10}$$

where, x_{ip} and x_{iq} are the public data and private utility functions in the current state of the i^{th} agent. The public data is known to every other agent in the neighborhood, whereas the private utility functions are known only to the agents in the same carrier or operator group. As shown in Eq. (10), the state change of an agent is a function of its own utility functions, and the public information of other non-group agents. The public utility functions of the non-group agents has an associate local trust denoted by δ_{ip} and an associated time-varying-reward denoted by $r(t)_i$. Furthermore, all agents are required to operate within certain global constraints denoted by c_0, which has an associated global trust σ_0, and reward r_0. This uncertainty could simply be a temporary communication brown out. While the vehicle dynamics are not explicitly expressed in Eq. (10), which is envisioned to serve as a high level decision model representation, it indirectly incorporates the vehicle specifications in the agents utility functions as well as the time budget t, which is derived from the agent's maneuverability and available resources. To summarize, the inner bracket in Eq. (10) can be seen as a black box that in conjunction constitutes the inter-agent trust ξ_{ij}. While it is possible to mathematically model ξ_{ij} based on the inner bracket parameters in Eq. (10), albeit with substantial complexity and consequent latency, this paper takes an approach to learn it via heuristic optimization method such as deep neural nets.

B. Data Thickening via Graphical Data Frames

Fig. 2 shows an example UAM scenario where two agents make course adjustments to avoid a weather event or a dynamic airspace closure notice issued by the city. Such course adjustments, however, can give rise to a potential conflict between the two agents. For a distributed model to work in such a scenario, the agents first need to be able to predict their trajectories and detect the conflict using onboard resources. While it is feasible to compute mathematically the possible trajectories for the agents, such computations, especially for multiple bi-lateral conflict detections, can quickly become computationally very complex. Current approaches to conflict detection compare one nominal trajectory for each flight along an intent such as the flight plan using a common speed, or in some cases very few trajectories such as along state projection, commanded state in the flight management system and the flight plan [8]. These assumptions may be suitable for traditional commercial operations where flights are forced to fly their intent as a contract and use consistent dynamics, however they are likely to prove brittle in UAM operations with volatile on-demand profiles that are unlikely to fly the same nominal trajectory every time. Furthermore, simple state projections for the agents can lead to more and more false positives, overwhelming the system. Alternatively, a large number of possible trajectories would need to be computed or separation buffers increased to account for the uncertainties, which is inefficient for high-tempo UAM operations. Machine learning offers an ability to test many possible trajectories and commit them to memory such that they are accounted for in the outcome of a neural network, without expensive re-computation every time. Finally, additional information regarding the agents' inherent utility functions, such as fuel usage preference, payload type, urgency etc., can improve the optimality of conflict resolution. Therefore, data enrichment with such utility function information is desirable as long as it does not significantly increase the computational burden on the onboard processor.

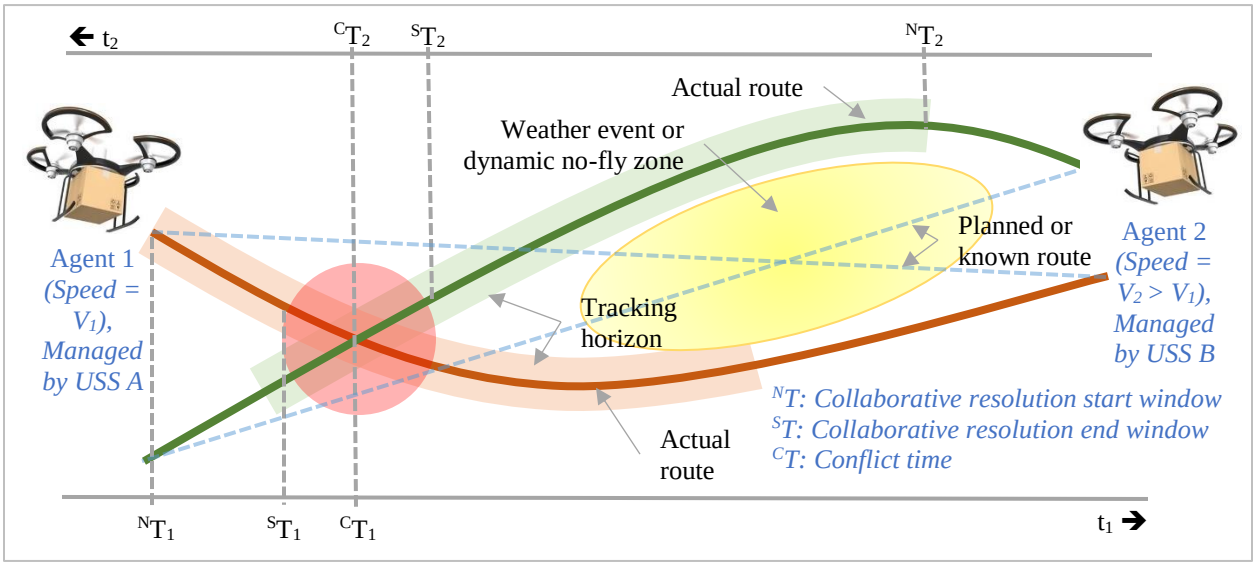

Fig. 2 Multi-agent scenario classification using graphical data frames

One potential approach to answer the above question is to utilize pixel manipulation to encode and embed agent behavior. As the total number of pixels in fixed dimensional images is the same, regardless of the content of the image, the processing burden will remain the same. While the image's 2D surface is used to represent the physical status of the agent (such as position, speed, heading, waypoints etc.), careful manipulation of pixels for agent icon shapes and colors can be used to represent the agent's non-physical attributes (for example: known behavior based on past experience, preference for payload comfort over fuel saving, flexibility to maneuver etc.). In some sense, this behavior encoding approach through graphical data frames is comparable to existing data visualization techniques such as heat maps, histogram plots, network diagrams, and word clouds etc. used for overall context representation. The difference in the case of the presented data-intent-behavior encoding approach is that the resulting graphical data frames are designed to be ingested by machines and not humans.

C. Steps for Learning and Interpretation of Graphical Data Frames by Deep Neural Net

In this paper, a deep neural net based heuristic optimization approach of integrated data synthesis and inference is used that allows localized derivation of spatiotemporal decisions with partial data from each agent in UAM scenarios. Given the high variability in data and multi-order solution space, the deep neural net learning approach is envisioned to enable the agent's onboard AI module to predict and resolve emerging operational conflicts by employing collaborative bi-/multi-lateral time-dependent strategies in real-time.

Deep learning [24] utilizes large neural networks, with multiple layers between the input and output layers of an artificial neural network. In recent times, significant progress has been made in developing and tuning such large neural networks to learn visual data, such as images and videos. Ground traffic forecasting and incident detection using deep learning approach has been reported in multiple studies [25] [26] [27]. Ma *et al.* have recently demonstrated the use of a convolutional neural network to learn the traffic as an image [28].

The deep learning work flow used in this paper goes through the following steps:

1) The data is sourced from a custom developed simulator that simulates different scenarios to gather the starting graphical data frames and the corresponding ground truths after simulating the scenario over time.
2) The collected data is automatically labeled with the ground truth from simulation, for learning
3) A transfer learning technique is used where a deep neural net with proven performance is taken as the starting point and then it is trained and fine-tuned with the labeled graphical data frames. During training the model is tested periodically with an evaluation dataset to make sure that the model accuracy is gradually improving
4) Once the model accuracy plateaus, training is stopped and it is deployed for real time classification of data

Note that the purpose of the deep learning approach is to capture the intent and behavior, and not just the operational data, of the entities in considerations. It is assumed that the governing dynamics and influencing parameters for the agent behavior is not publicly known, therefore, the behavior is only observable and not controllable globally. With an appropriate reward system, however, the behavior can be changed during bi-/multi-lateral negotiations. In fact, in some cases with making the information available and easily interpretable, the behavior can be altered. As mentioned earlier, the utilization of the presented deep learning based collective behavior learning and scenario evolution predictions for a distributed UTM decision engine will be discussed in a separate paper.

For the deep neural learning and prediction feasibility study, in this paper, three specific neural nets were selected. The first one is a residual neural network or ResNet [29]. ResNets utilize skip connections or shortcuts to jump over some layers to avoid the problem of vanishing gradient that effectively prevents the value change for weights. ResNets solve this issue by reusing activations from a previous layer until the adjacent layer learns its weight. This approach simplifies the network and speeds up learning as there are fewer layers used during the initial training stage. The network then gradually restores the skipped layers as it learns the feature space and towards the end of training all layers are expanded. ResNet was released in 2015 by Microsoft Research Asia with three architecture realizations i.e. ResNet-50, ResNet-101 and ResNet-152. For the study in this paper, ResNet101 was used, which has 101 deep layers.

The second neural network that was selected for the presented study is developed by Oxford University's Visual Geometry Group in 2014, and known as VGG class of deep neural networks [30]. Neural networks prior to VGG used bigger, such as 7 by 7 pixels or 11 by 11 pixels, receptive fields. The receptive field is the part of the image that is visible to one filter at a time. VGG networks implemented smaller receptive fields (3 by 3 pixels) and increased the depth of the neural network to show improvements to the classification accuracy. Two variations of VGG, i.e. VGG-16 and VGG-19 are used most widely. For the study in this paper, VGG-19 was used, which has 19 deep layers.

The third neural network that was used in the presented study is called Inception [31]. The inception deep convolutional architecture was introduced by Google in 2015. Multiple iteration of this architecture has been made following the first version or Inception-v1 to include batch normalization in Inception-v2 and additional factorization in Inception-v3. The main idea with Inception architecture is to reduce number of connections or parameter without decreasing the network efficiency. For the study in this paper, Inception-v3 was used which is 48 layers deep.

IV. Concept of Operation Use Case and Performance Metrics

FAA's NextGen Concept of Operations (ConOps) for UTM [7], in its appendix E, outlines an inventory of use cases in multiple technology capability levels (TCLs). For the feasibility analysis of the presented approach, this paper takes up an embodiment of a TCL2 use case from this inventory that entails beyond visual line of sight (BVLOS) operation in uncontrolled airspace. Furthermore, this paper performs such analysis with the distributed model, i.e. beyond the federated USS-based model, for de-confliction within operation volumes.

Fig. 3 (a) shows a rendering for the TCL2 use case, where the airspace under consideration is used by UAM agents as well as conventional air vehicles. Two intersecting corridors in such environment are highlighted in Fig. 3 (b), which constitutes the use case scenario for the approach presented in this paper.

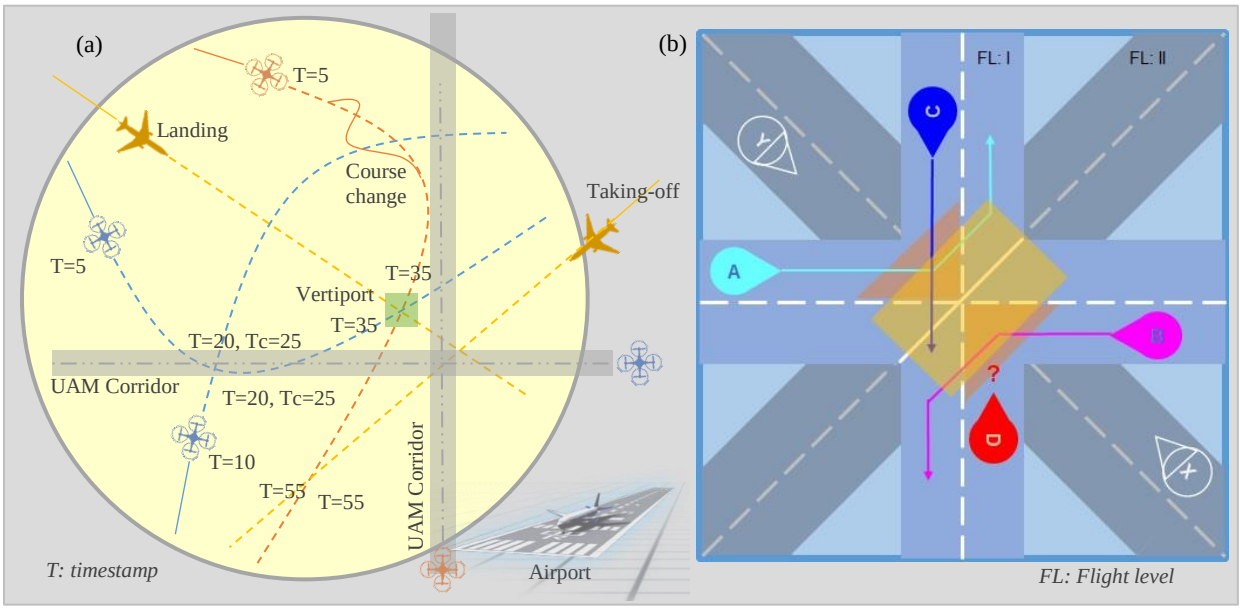

Fig. 3 ConOps use case: (a) Broad view of the diverse airspace, (b) Corridor intersection scenario

Fig. 3 (b) shows two pairs of corridors at two different flight levels (altitudes). In this scenario, the conflict assessment focus is put on one flight level alone. Agents not at this flight level are represented by hollow icons, for example: *'X'* and *'Y'* in Fig. 3 (b). Agents that are already at or going to be at the flight level of consideration are represented by color filled icons, for example: *'A'*, *'B'*, *'C'* and *'D'*. Furthermore, we assign certain baseline behavior for the agents approaching the corridor intersections, as follows:

- Agents of designations *'A'* and *'B'* always turn (either to the left or to the right) at the intersection
- Agents of designation '*C'* always go straight through the intersection
- Agents of designation '*D'* go straight through the intersection in most cases (90% of the time).
- Agents turning left stay closer in for shortest turn
- Agents turning right stay farther out for shortest turn
- Agents can approach the intersection from any one of the eight directions from these two corridor pairs
- Agents *'A'* and *'B'*, and agents *'C'* and *'D'* always come from opposing directions
- Additionally, agents *'A'* and 'C' are managed by USS *1* and agents *'B'* and *'D'* are managed by USS *2*

The agent to agent separation threshold is defined as follows:

- Intra-network separation threshold – Agents from the same operator/servicer, for example *'A'* and *'C'*, can maintain a comparatively shorter separation distance assuming that such agents have complete knowledge about each other's state and intent through their common servicer.
- Inter-network separation threshold – Agents from different operators/servicers, for example *'A'* and *'B'*, will need to maintain a comparatively longer separation distance assuming that such agents do not have complete knowledge about each other's state and intent.

The intersection brings the agents closer, where some proximities are converging (for example: agents *'A'* and *'C'* in the scenario shown in Fig. 3 (b)) and some are diverging (for example: agents *'A'* and *'B'* in the scenario shown in Fig. 3 (b)). A custom simulator has been developed to create the scenario and simulate it with the above behaviors.

V. Custom Simulator for Scenario Creation and Ground Truth Collection

A custom simulator and GUI (see Fig. 4) has been developed in the MATLAB® App Designer environment, which allows multi-agent simulation in conops scenarios by selection of parameters and controls interactively.

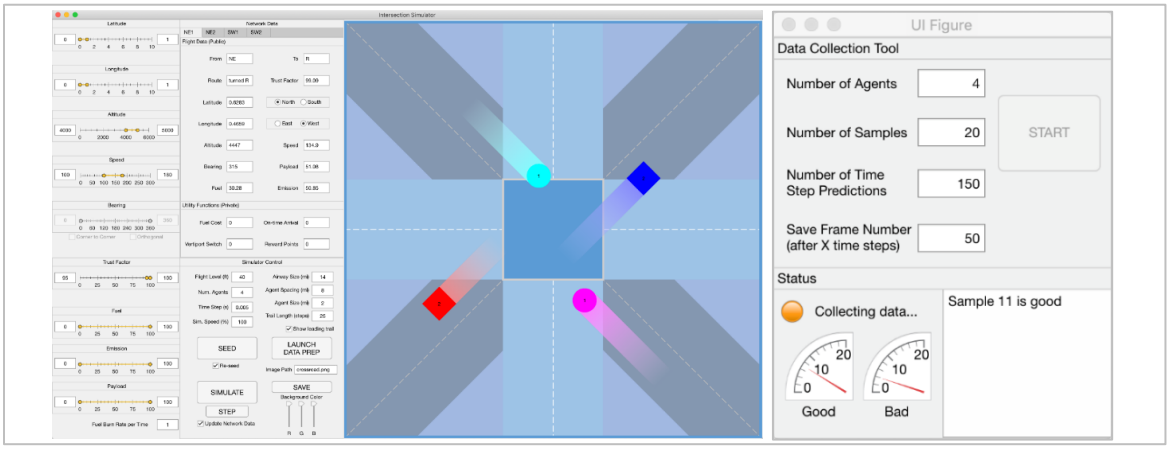

Fig. 4 (a) Custom developed simulator, (b) Data collection module

Each agent is initialized with a position, altitude, speed, bearing, trust factor, and utility functions (or business priorities) for fuel emission, payload, and fuel burn rate. The simulator controls include the flight level of interest, time step, speed, airway size, agent spacing, agent size, and length of the leading path projection. Any agents outside of the specified flight level appear as hollow circles or squares to indicate they are out of the altitude range of interest. When initialized, the program chooses random values within the ranges specified in the left panel as the agents' starting state. When simulating begins, the state of each agent and map is updated once for each time step.

The airway, with a total width set in the simulator control panel, is randomly set in a vertical and horizontal shape or in a diagonal cross shape. In this example, red and blue squares are used to indicate these agents' intent to continue straight ahead through the intersection and may choose either the left or right lane. Magenta and cyan circles are used to indicate the agents' intent to turn left or right and may only choose the left or right lane accordingly. The gradient rectangle in front of the agent in the direction of motion projects the path from the current position at time t to the agent's projected position at time $(t + \Delta t)$, where Δt is a user-defined offset. $\Delta t = 40$ in the presented study.

A data collection module of the custom simulator collects a set of samples labeled as *"good"*, in which no violation of the separation standards occurred within a certain projected time period, and a set of samples labeled as *"bad"*, in which a violation of the separation standards did occur. The separation standard is analogous to the same in terminal airspace traffic control, which is the designated center-to-center distance for two specific aircrafts. With N as the number of samples and T as the number of time step predictions, the program initializes the simulator and updates it T time steps. If a conflict occurs at or before time step T, the sample is labeled *"bad"* and if no conflict occurs, the sample is labeled *"good"* and moved into the respective dataset. This process repeats until at least N *"good"* and N *"bad"* samples have been labeled and stored. The save frame number specifies up to which time step to simulate to capture the states of the agents in the map.

Using MATLAB® Deep Learning Toolbox™, the synthesized data is learnt by a convolutional neural network (CNN). The dataset is split into 70% training, 20% validation, and 10% testing sets. The validation dataset is used for testing throughout training to ensure the model's accuracy is improving. The testing dataset is put aside for testing after training to ensure the model is not over-fit to the training data, in other words the model does not lose its generality in classifying new data frames. The architectures of three commonly used CNNs in image classification are tested: ResNet-101, VGG-19, and Inception-v3. The final layers of each model are replaced to classify images labeled *"good"* and *"bad"*. The models are trained on the full synthesized dataset with *6N* samples, comprised of *2N* white background, *2N* transparent background, and *2N* crossroad image background samples. The idea behind using three different backgrounds for the same data frame is to suppress the impact of the background as a feature and put more emphasis on the agents and their behavior represented by their shape, color, and future path projections. Results using data augmentation, Bayesian optimization, and grayscale-only images are discussed in the following section. More information about the deep neural networks used, and the overall learn-to-classify workflow can be found in [32].

ResNet-101, VGG-19, and Inception-v3 architectures were used for comparison of results. Tests were done on each of these models with 4, 6, and 8 agents in the scenario. Graphical data frames were collected in RGB and

grayscale image formats. A simple averaging method was applied to the RGB images to convert them to grayscale. Additionally, to improve the diversity of the data, a two phased data augmentation step was used. The phase I of the data augmentation is the use of three backgrounds, i.e. crossroad image, transparent, and white, as mentioned earlier. The phase II of the data augmentation involved random x-axis reflections, y-axis reflections, and rotations between [-15º, 15º]. Finally, a Bayesian optimization step was used to find the following five training options that resulted in the lowest validation error: initial learning rate, learn rate drop period, learning rate drop factor, L_2 regularization factor, and momentum. Training of the deep neural nets, i.e. ResNet-101 with 101 layer, VGG-19 with 19 layers, and Inception-v3 with 48 layers, was conducted using CUDA®, which is a parallel computing platform and programming model developed by NVIDIA® for general computing on graphical processing units (GPUs).

VI. Simulation Results and Summary of Findings

The following simulation results were obtained from the training of a dataset consisting of 4,000 graphical data frames with equal distribution of *"conflict"* and *"no conflict"* classes. Each image was included in the dataset with three different backgrounds as mentioned earlier. So the total number of images used in the deep neural net training and testing is 12,000. An NVidia Quadro RTX 6000 GPU with 24GB GDDR6 frame buffer and 4,608 CUDA cores, in conjunction with an Intel Xeon platinum 8160, 3.70 GHz, 24 core CPU was used for the training and classification. Table 1 includes a comparison between the three CNN networks with 4, 6, and 8 agents and shows the effects of data augmentation and Bayesian optimization on the model's accuracy.

Table 1 Accuracy of three neural net models with varying number of agents, data augmentation, & optimization

Model	Accuracy with Phase I Data Augmentation Only	Accuracy with Phase I & II Data Augmentation	Accuracy with Phase I Data Augmentation and Bayesian Optimization
ResNet-101, 4 agents	95.6%	84.9%	96.4%
VGG-19, 4 agents	97.2%	96.3%	96.3%
Inception-v3, 4 agents	98.8%	97.5%	96.0%
ResNet-101, 6 agents	88.4%	84.6%	93.3%
VGG-19, 6 agents	73.3%	83.5%	95.4%
Inception-v3, 6 agents	91.5%	85.8%	92.2%
ResNet-101, 8 agents	86.5%	85.3%	84.2%
VGG-19, 8 agents	81.5%	84.2%	92.0%
Inception-v3, 8 agents	84.4%	76.2%	84.3%

The phase II data augmentation applied to the training dataset generally decreased the overall accuracy of the models by 2%, with as much as an 11% decrease. This may be caused by the specific data augmentation method used where, unlike common augmentation, the training dataset does not increase in size but is simply altered, coupled with the possibility that the agents in the rotated images, especially in transparent or white background, loose their motion directionality resulting in training ambiguity. As a different improvement method, Bayesian optimization [33] was applied to the models trained with phase I data augmentation only. Hyper-parameter tuning generally increased the overall accuracy of the models by 4%. The drop in model accuracy due to the increase in the number of agents in the scenario was the smallest, i.e. 4%, for VGG19 network. This drop in accuracy can be attributed to multiple factors including agent representation, path projection, and the data augmentation methods used. These factors will be further investigated in our future work to improve the overall accuracy with higher number of agents in the scenario.

Table 2 shows the performance of the three neural nets with 3-channel color data frames vs. single channel grayscale data frames. ResNet-101 and VGG-19 were more robust to this change in comparison to Inception-v3 model.

Table 2 Accuracy of three models trained on color versus grayscale images with four agents

Model	RGB	Grayscale
ResNet-101	95.6%	94.2%
VGG-19	97.2%	98.3%
Inception-v3	98.8%	88.8%

Next a mixed scenario, where any number of agents can approach the corridor intersection, was evaluated. For this the three neural network models were trained with the combination of the datasets for 4, 6, and 8 agents, i.e. with a total of 36,000 data frames. Separately a test dataset with an equal number of data frames from each of the two classes

was prepared using the simulator for scenarios consisting of 4 to 8 agents with the three backgrounds. This test dataset consists of 9,000 data frames that the trained models have not seen earlier. For each one of the three deep neural networks used in this study, Table 3 shows the training time per epoch, and classification accuracy and time.

Table 3 Comparison of three deep neural networks in training and classification of mixed number of agents

Model	Training Time/Epoch	Classification Accuracy	Classification Time/Data Frame
ResNet-101	30 minutes	89.2%	3.5 milliseconds
Inception-v3	26 minutes	89.2%	6.3 milliseconds
VGG-19	8.5 minutes	89.4%	7 milliseconds

The maximum number of epochs for each model training was set to be 15 and an initial learning rate of 0.0002 was used. The overall accuracy can be boosted further with training to more epochs, and with a lower learning rate.

VII. Conclusion and Future Works

This paper presented a conflict prediction and classification approach using synthesized graphical data frames, learnt by deep neural networks. A custom simulator was developed to create the data frames by integrating the physical state as well as the utility functions of the agents. Ground truth regarding scenarios were collected by simulating the scenario up to a fixed time step into the future and detecting conflicts based on violation of agent-to-agent separation criteria. Based on these simulation results, the scenarios were labeled appropriately for training. Three different neural networks were trained with such datasets and their performance in term of accurately classifying new scenarios were recorded. Preliminary findings validate the feasibility of the proposed graphical data synthesis and AI-based holistic scenario assessment approach. Further fine-tuning of the deep neural network models and training hyper-parameters as well as data augmentation methods, is expected to significantly boost the overall classification accuracy. Being a heuristic machine learning method, the presented capability is not intended to replace other deterministic methods for safety-critical conflict detection and resolution, but rather to complement them with (a) accounting for much larger datasets potential futures learned into the neural net, (b) early precursor prediction with fast computation, and (c) elimination of false positives based on a wider set of behavioral as well as physical vehicle properties.

Future work will involve refinement of the deep neural net structures to improve their efficiency. Additionally, the data synthesis process will be investigated further, in conjunction with feature learning, to identify the most influential ways to synthesize the collective behavior of the agents in particular scenarios. Last but not least, implementation of the presented technology on portable edge devices, such as NVidia Xavier edge-GPU module [34] with TensorRT optimization [35], will be explored to enable scenario classification and implementation of decision engines onboard the UAM agents.

References

[1] FAA, "Introduction to TCAS II (Version 7.1)," 2011.

[2] EuroControl, "ACAS X – the future of airborne collision avoidance," *NETALERT - the Safety Nets newsletter,* June 2013.

[3] J. K. Kuchar and L. C. Yang, "A Review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems,* vol. 1, no. 4, pp. 179-189, 2000.

[4] G. Manfredi and Y. Jestin, "An introduction to ACAS Xu and the challenges ahead," in *IEEE/AIAA 35th Digital Avionics Systems Conference (DASC),* Sacramento, CA, 2016.

[5] Vertical Flight Society, "eVTOL Aircraft Directory," 2017. [Online]. Available: https://evtol.news/aircraft/.

[6] M. Skowron, W. Chmielowiec, K. Glowacka, M. Krupa and A. Srebro, "Sense and avoid for small unmanned aircraft systems: Research on methods and best practices," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering,* vol. 233, no. 16, pp. 6044-6062, 2019.

[7] FAA, "Concept of Operations - Unmanned Aircraft System (UAS) Traffic Management (UTM) v2.0," 2020.

[8] H. Idris, G. Enea and T. A. Lewis, "Function Allocation between Automation and Human Pilot for Airborne Separation Assurance," in *13th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems,* Kyoto, Japan, 2016.

[9] T. Prevot, T. J. Callantine, P. U. Lee, J. Mercer, V. Battiste, E. Palmer and N. Smith, "Cooperative Air Traffic Management: A Technology Enabled Concept for the Next Generation Air Transportation System," *Cooperative Air Traffic Management: Concept and Transition,* 2005.

[10] C. Tomlin, G. J. Pappas and S. Sastry, "Conflict Resolution for Air Traffic Management: A Study in Multiagent Hybrid Systems," *IEEE Transactions on Automatic Control,* vol. 43, no. 4, pp. 509-521, 1998.

[11] J. Hoekstra, J. Groeneweg and G. v. Ronald, "Airborne Separation Assurance Validation with Multiple Humans-in-the-loop," in *Air Traffic Management (ATM) Research and Development*, Budapest, Hungary, 2003.

[12] R. C. J. Ruigrok and B. Korn, "Combining 4D and ASAS for Efficient TMA Operations," in *7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Belfast, Northern Ireland, 2007.

[13] H. Idris, R. Vivona and D. Wing, "Metrics for Traffic Complexity Management in Self-Separation Operations," *Air Traffic Control Quarterly,* vol. 17, no. 1, pp. 95-124, 2009.

[14] T. Prevot, P. Lee, N. Smith and E. Palmer, "ATC Technologies for Controller-Managed and Autonomous Flight Operations," in *AIAA Guidance, and Control Conference and Exhibit*, San Francisco, CA, 2005.

[15] H. Erzberger, T. A. Lauderdale and Y.-C. Chu, "Automated conflict resolution, arrival management, and weather avoidance for air traffic management," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering,* vol. 226, no. 8, pp. 930-949, 2011.

[16] B. J. A. Van Marwijk, C. Borst, M. Mulder, M. Mulder and M. M. Van Paassen, "Supporting 4D Trajectory Revisions on the Flight Deck: Design of a Human–Machine Interface," *International Journal of Aviation Psychology,* vol. 21, no. 1, pp. 35-61, 2011.

[17] R. E. Klomp, R. Riegman, C. Borst, M. Mulder and M. M. van Paassen, "Solution Space Concept: Human-Machine Interface for 4D Trajectory Management," in *Thirteenth USA/Europe Air Traffic Management Research and Development Seminar*, Vienna, Austria, 2019.

[18] H. Idris, D. Delahaye and D. Wing, "Distributed Trajectory Flexibility Preservation for Traffic Complexity Mitigation," in *Eighth USA/Europe Air Traffic Management Research and Development Seminar*, Napa, CA, 2009.

[19] M. Xue and M. Do, "Scenario Complexity for Unmanned Aircraft System Traffic," in *AIAA Aviation 2019 Forum*, Dallas, TX, 2019.

[20] J. H. Seo, H. Shim and J. Back, "Consensus of high-order linear systems using dynamic output feedback compensator: Low gain approach," *Automatica,* vol. 45, no. 11, pp. 2659-2664, 2009.

[21] F. L. Lewis, H. Zhang, K. Hengster-Movric and A. Das, Cooperative Control of Multi-Agent Systems, Springer, 2014.

[22] D. Helbing, I. Farkas and T. Vicsek, "Simulating dynamical features of escape panic," *Nature, Letter,* vol. 407, pp. 487-490, 2000.

[23] S. Boyd, "Convex optimization of graph Laplacian eigenvalues," *Proceedings of the International Congress of Mathematicians,* vol. 3, pp. 1311-1320, 2006.

[24] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[25] L. Zhu, F. Guo, R. Krishnan and J. W. Polak, "A Deep Learning Approach for Traffic Incident Detection in Urban Networks," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, Hawaii, 2018.

[26] J. Bai and Y. Chen, "A Deep Neural Network Based on Classification of Traffic Volume for Short-Term Forecasting," *Mathematical Problems in Engineering,* 2019.

[27] L. Fridman, J. Terwilliger and B. Jenik, "DeepTraffic: Crowdsourced Hyperparameter Tuning of Deep Reinforcement Learning Systems for Multi-Agent Dense Traffic Navigation," in *32nd Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, 2018.

[28] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang and Y. Wang, "Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction," *Sensors,* vol. 17, no. 4, pp. 818-833, 2017.

[29] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[30] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations*, San Diego, CA, 2015.

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, 2016.

[32] Mathworks, "Deep Learning with Images," 2018. [Online]. Available: https://www.mathworks.com/help/deeplearning/deep-learning-with-images.html.

[33] J. Snoek, H. Larochelle and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," *Advances in neural information processing systems,* pp. 2951-2959, 2012.

[34] NVidia, "Jetson AGX Xavier," 2019. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/.

[35] NVidia, "NVidia TensorRT," 2019. [Online]. Available: https://developer.nvidia.com/tensorrt.