A Learning-Based Guidance Selection Mechanism for a Formally Verified Sense and Avoid Algorithm

Swee Balachandran[1], Viren Bajaj[1], Marco A. Feliú[1], César A. Muñoz[2], María C. Consiglio[2]

***Abstract*— This paper describes a learning-based strategy for selecting conflict avoidance maneuvers for autonomous unmanned aircraft systems. The selected maneuvers are provided by a formally verified algorithm and they are guaranteed to solve any impending conflict under general assumptions about aircraft dynamics. The decision-making logic that selects the appropriate maneuvers is encoded in a stochastic policy encapsulated as a neural network. The network's parameters are optimized to maximize a reward function. The reward function penalizes loss of separation with other aircraft while rewarding resolutions that result in minimum excursions from the nominal flight plan. This paper provides a description of the technique and presents preliminary simulation results.**

I. Introduction

Formal methods for safety critical software has become the focus of many research efforts seeking increased reliability and ultimately more efficient regulatory acceptance of autonomous systems. Despite the enormous progress in formal verification techniques, compliance with Federal Aviation Regulations for the certification of safety critical software is still achieved by evidence from extensive testing, simulation, and flight tests. Although testing and simulation play a crucial role in the certification of critical flight software, formal verification of the software logic can eliminate non-trivial logic errors and also, coupled with modular software design, help cut down the maintenance cost of flight software development.

The Prototype Verification System (PVS) [1] was used to formally verify the logical correctness of the Detect and Avoid Alerting Logic for Unmanned Systems (DAIDALUS) [2], [3]. DAIDALUS is a software library that serves as a reference implementation of the detect and avoid (DAA) concept described in the RTCA DO-365 Minimum Operational Performance Standards for Unmanned Aircraft Systems [4]. DAIDALUS source code is available in both C++ and Java under NASAs Open Source Agreement. The DAIDALUS software library consists of algorithms that predict "well-clear" violations (as defined in RTCA DO 365) between between aircraft and provide maneuver guidance to the pilot in command in the form of maneuver ranges.

DAIDALUS was originally designed as an advisory tool for the pilot in command of an Unmanned Aircraft System (UAS). DAIDALUS provides four main types of guidance resolutions to avoid loss of separation of an ownship with near aircraft in the airspace: track, ground-speed, vertical-speed, and altitude resolutions. The DAA MOPS stipulates that, it is the responsibility of the pilot in command to choose an appropriate resolution maneuver among the four types of resolutions provided and execute it according to the established rules and procedures. The potential introduction of large number of small unmanned vehicles operating at low altitudes introduced a new set of technical challenges that may impede a pilot or operator to execute DAA tasks in a timely of effective manner. Additionally, pilot-vehicle communication challenges at low altitudes are likely to experience interruptions that can prevent a pilot initiation of conflict avoidance actions. Hence, the implementation of on board capabilities to enable varying degrees of mission and separation autonomy may be required for autonomous UAS operations.

In this context, the ability to autonomously select a single avoidance maneuver from the DAIDALUS resolution ranges would relieve the need for a human in charge of that selection. This capability can be incorporated in different operational concepts with varying degrees of autonomy, where the execution of the avoidance maneuver could be operators responsibility or part of a fully autonomous capability such as ICAROUS (Independent Configurable Architecture for Reliable Operations of Unmanned Systems) [5], [6]. ICAROUS comprises a set of core algorithms for path planning, geofence handling, traffic avoidance and decision making to enable autonomous operations of UAS.

This paper develops a Reinforcement Learning (RL) framework for DAIDALUS. In this framework, DAIDALUS provides ranges of collision resolution maneuvers whose bounds (e.g., target heading, ground speed, altitude, climb rate) are determined by its formally verified deterministic algorithm based on a geometric model. The policy obtained using this RL framework is then used to select an optimal resolution maneuver with respect to a given reward function. Since the learning mechanism only operates on formally verified resolutions, the formal properties of the DAIDALUS algorithm are preserved and safety of the UAS is ensured.

The rest of this paper is organized as follows. Section II highlights several relevant areas of research related to the use of formally assured sense and avoid algorithms. Section III provides background information about the various tools and algorithms used in this work. Section IV details the learning framework, the reward formulation and the training mechanism. Section V presents the results of the described learning framework. Section VI provides concluding remarks and discusses future work.

[1]National Institute of Aerospace, Hampton, Virginia 23666. swee.balachandran@nianet.org.

[2]NASA Langley Research Center, Hampton, Virginia 23666. cesar.a.munoz@nasa.gov.

II. Related Work

The *Traffic Alert and Collision Avoidance System* (TCAS) is a family of airborne devices that are designed to reduce the risk of mid-air collisions between aircraft equipped with operating transponders [7]. TCAS has evolved through extensive development and a number of versions since its initial operational evaluation in 1982. TCAS II, the current generation of TCAS devices, is mandated in the US for aircraft with greater than 30 seats or a maximum takeoff weight greater than 33,000 pounds. The formal verification of TCAS logic first appeared in a paper by Lygeros and Lynch [8].

The Advanced Collision Avoidance System (ACAS-X) [9], the next generation of collision avoidance algorithms was developed with a goal of improving TCAS performance and reliability. Unlike TCAS, ACAS-X uses a model-based decision-theoretic framework where traffic resolutions and advisories are optimized using a reward function taking into account the encounter dynamics. However, the use of a decision-theoretic framework introduces multi-dimensional lookup tables thus making verification and validation a challenging task. Verification and validation of ACAS-X resolutions is an ongoing research activity [10]. An adaptation of ACAS-X for small UAS called ACAS-Xu is in development [11].

Unlike ACAS-X, the present work does not directly attempt to optimize traffic resolutions using a reward function. Instead, different types of resolution maneuvers (i.e., track, ground speed, vertical speed, and altitude) required to maintain well clear are first determined by analytical algorithms that are formally verified in PVS. Subsequently, the selection of a suitable resolution from the set of formally verified resolutions is obtained using a learning mechanism thus preserving the formal properties of the DAIDALUS algorithm, i.e., the selected maneuver is conflict free and satisfies the specified mission constraints.

III. Background

A. DAIDALUS

DAIDALUS (Detect and AvoID Alerting Logic for Unmanned Systems) is a software implementation intended to satisfy the operational and functional requirements detailed in NASA's DAA concept of integration for UAS [2]. In particular, DAIDALUS provides algorithms for three distinct functionalities: 1) *detection*: determines the current, pairwise well-clear status of the ownship and all aircraft inside its surveillance range, 2) *manuever guidance*: computes ranges of maneuvers that a pilot-in-command (PIC) may take that will either cause the aircraft to maintain or increase separation from the well-clear violation volume, or allow for recovery from loss of separation in a timely manner within the performance limits of the ownship aircraft, and 3) *alerting*: determines the corresponding alert type, based on a given alerting schema.

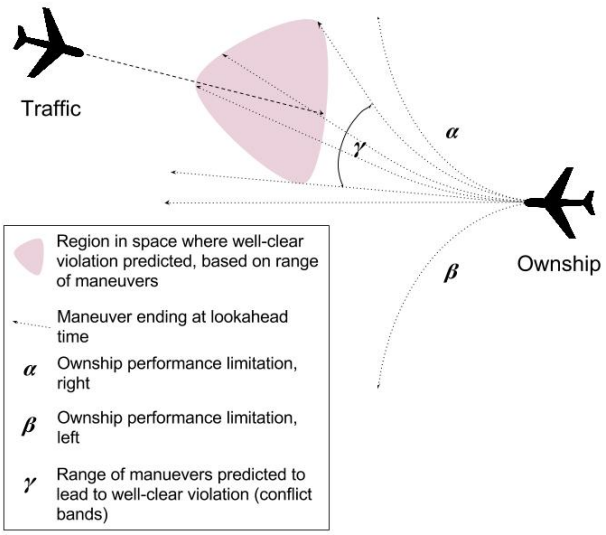

Fig. 1: DAIDALUS sense and avoid

B. ICAROUS

ICAROUS (Independent Configurable Architecture for Reliable Operations of Unmanned Systems) [5], [6], [12] is a distributed software architecture designed to provide UAS with decision making capabilities to operate autonomously. The core functionalities include maintaining safe separation with other intruders in the airspace, conforming to geospatial airspace constraints (keep-in/keep-out geofences), avoiding obstacles and performing route planning when alternate flight routes are essential to accomplish mission goals (path planning). This work specifically focuses on the sense and avoid functionality of ICAROUS which uses the DAIDALUS library described above for conflict detection and maneuver guidance. Given the position and velocity of the ownship and any intruders in the airspace, DAIDALUS provides maneuver guidance in the form of conflict-free ranges of track, speed, altitude and vertical speed resolutions from where the path planning functions in ICAROUS selects one maneuver for execution by the autopilot. In addition, the path planning function computes a "return to path" maneuver to enable the ownship to return to the original mission after the conflict has been resolved. This maneuver is also free of conflicts and compliant with mission constraints. Since the resolutions provided by DAIDALUS are not aware of mission constraints such as geofences, obstacles, and path deviation constraints, ICAROUS has to select a resolution that is optimal for the current mission. Currently, ICAROUS does this by selecting a user defined default resolution for all cases.

One of the main advantages of using a modular design in ICAROUS with publish-subscribe architecture is that it preserves the properties of formally verified modules. In particular, any single maneuver selected from the maneuver guidance ranges computed by DAIDALUS is guaranteed to be conflict free and correct regardless of the selection logic. However, an optimizing selection technique may introduce

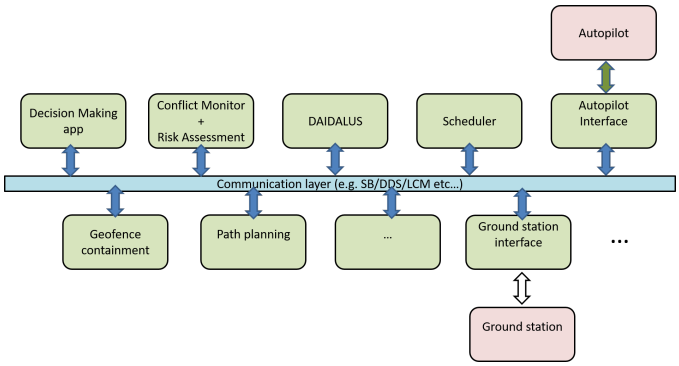

Fig. 2: ICAROUS architecture

non-deterministic behaviors such as non-convergence. The impact of these behaviors need to be understood before they are incorporated in safety critical applications.

C. Reinforcement learning

Reinforcement learning (RL) [13] is a paradigm where an agent, operating in an unknown environment, learns to achieve a specific goal by taking repeated actions and evaluating the outcomes of each actions. A utility function is used to evaluate each action from a given state thus providing feedback on how useful a given action is when executed from that specific state to achieve the goal at hand. By selecting only the actions that provide maximum utility, the agent can achieve its goal. More formally, a reinforcement learning agent can be described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ represents the state space of the agent, $\mathcal{A}$ represents the set of available actions, $\mathcal{R}$ represents the utility function, and γ represents the discount factor ($\gamma \in [0, 1]$.) The utility of a given state is defined as the expected discounted cumulative reward. The discount factor determines the tradeoff between immediate rewards over future rewards in the utility function. Given a current state $s_t \in \mathcal{S}$, the agent selects action $a_t \in \mathcal{A}$ according to a policy π. On executing the action, the agent transitions to the next state and receives a reward r_t. Due to the uncertainty in the environment and possible outcomes of the available actions, this transition is stochastic. The agent continues to accumulate rewards by continuing this process until it either reaches a goal or a failure state. Reinforcement learning tries to converge on an optimal policy, i.e., one that maximizes the expected discounted cumulative reward (Equation (1)) the agent receives. Note that $\mathbb{E}$ denotes the expectation and R denotes the reward received at the current time step. In this work, learning is done without any prior knowlege of the uncertainty associated with the environment or the state transitions.

$$\mathcal{R} = \mathbb{E}\left(\sum_{t=0}^{t=N} \gamma^t R(s_t, a_t)\right) \tag{1}$$

IV. Learning Optimal SAA Resolutions in ICAROUS

In this paper, an RL agent is integrated into ICAROUS to learn the optimal resolution from the various types of resolutions provided by DAIDALUS, namely track, ground speed, altitude, and vertical speed resolution. The optimality criteria is discussed in Section IV-C.

A. State and Action Space Formulation

The state space (s) of the RL agent consists of the relative position ($\vec{p}_r$) and velocity ($\vec{v}_r$) of the intruder with respect to the ownship:

$$s = [\vec{p}_r, \vec{v}_r] \in \mathbb{R}^6 \tag{2}$$

$$\vec{p}_r = \vec{p}_i - \vec{p}_o \tag{3}$$

$$\vec{v}_r = \vec{v}_i - \vec{v}_o \tag{4}$$

where $\vec{p}_i$ and $\vec{v}_i$ are the 3-D position and velocity of the intruder and $\vec{p}_o$ and $\vec{v}_o$ are the 3-D position and velocity of the ownship.
The action space (a) of the RL agent consists of the types of resolution obtained from DAIDALUS:

$$a = \{a_t, a_s, a_h, a_v\} \tag{5}$$

where a_t, a_s, a_h, a_v represent the track, ground speed, altitude and vertical speed resolutions respectively. Track and ground speed represent resolutions in the horizontal dimension (x-y), whereas altitude and vertical speed represent resolutions in the vertical dimension (z).

B. Policy Parameterization

The policy $\pi(\theta) : s \rightarrow a$ that assigns a suitable action for every given state is parameterized by $\theta \in \mathbb{R}^n$. In this framework, θ represents the weights and biases of a fully connected neural network. The neural network is comprised of an input layer with six nodes corresponding to the dimension of the state space, two hidden layers with twenty nodes each, and a softmax (Equation 6) output layer [14] consisting of four nodes corresponding to the dimension of the action space (Equation (5)). The hidden layers all use the Rectified Linear Unit (ReLU) activation function (Equation 7) [14]. The output layer provides a probability distribution over the action space, and the weights of the network (the parameters of the policy) are chosen to maximize the cumulative discounted reward.

$$\sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for i=1,...,K and z} \in \mathbb{R}^K \tag{6}$$

$$f(x) = max(0, x) \tag{7}$$

C. Reward Formulation

The utility function used to guide learning is crafted to encapsulate the properties of an optimal resolution. An optimal resolution is considered to be one that:

1) Minimizes Severity of Loss of Well-Clear (SLoWC). SLoWC is a metric used to assess the most serious instance of LoWC in an encounter and is defined at each time step by Equation (8).

$$\texttt{SLoWC} = (1 - \texttt{RangePen} \oplus \texttt{HMDPen} \oplus \texttt{VertPen}) \times 100\% \quad (8)$$

where the operator $\oplus$ is defined as:

$$x \oplus y \equiv \sqrt{x^2 + (1-x)^2 y^2} \quad (9)$$

and `RangePen`, `HMDPen`, `VertPen` are defined in [4].

2) Minimizes Path Deviation.
`Path Deviation` $= h_d + v_d$, i.e., the sum of the horizontal (h_d) and vertical (v_d) cross track deviations from the flight plan segment between points A and B (Figure 3), which can be defined as:

$$\begin{aligned}
\vec{r}_{AB} &= B - A \\
\vec{r}_{AO} &= \vec{p}_o - A \\
h_d &= |\vec{r}_{AO} \cdot \perp \vec{r}_{AB}| \\
v_d &= \left| r_{z_{AO}} - r_{z_{AB}} \left(\frac{r_{x_{AO}} \cdot r_{x_{AB}} + r_{y_{AO}} \cdot r_{y_{AB}}}{||\vec{r}_{AB}||} \right) \right|
\end{aligned}$$

where $\perp \vec{r}_{AB}$ represents the perpendicular to $\vec{r}_{AB}$, r_x, r_y, r_z represent the components of the $\vec{r}$, and h_d, v_d represent the magnitude of the horizontal and vertical deviations from the flight plan segment AB.

3) Reaches the destination waypoint.
Taking d_c as the radius within which the vehicle is considered to be at waypoint B, reaching the destination can be defined by the function:

$$\texttt{Reached} = \begin{cases} 0 & ||p_o - B|| > d_c \\ 1 & otherwise \end{cases} \quad (10)$$

In order to satisfy these optimality criteria, the reward function is defined as follows.

$$\begin{aligned}
R_{SLoWC} &= \max_{t<T}(\texttt{SLoWC}) \\
R_{pathdev} &= \max_{t<T}(\texttt{Path Deviation}) \\
R_{reach} &= \texttt{Reached} \\
R &= \alpha_1 R_{SLoWC} + \alpha_2 R_{pathdev} + \alpha_3 R_{reach} \quad (11)
\end{aligned}$$

where, t represents the simulation time step for a given episode and T represents the total duration of the episode.

The weights $\alpha_1, \alpha_2, \alpha_3$ are chosen to emphasize the relative importance of the individual reward. Negative rewards become penalties. The values α_1 and $\alpha_2 < 0$ are constrained so that high values of SoLWC and Path Deviation are penalized, and $\alpha_3 > 0$ so that reaching the destination is rewarded. Specifically, the following values are considered $\alpha_1 = -2000, \alpha_2 = -1000, \alpha_3 = 1000$.

D. Learning

A simulation environment is set up where the ICAROUS software architecture can interact with a simulated ownship and intruder traffic vehicle. The ownship is set up to follow a flight plan segment between two waypoints. The intruder's initial condition is defined by the range, bearing, and altitude from the ownship along with its ground speed, heading, and vertical speed. The intruder's initial conditions are sampled such that a loss of separation (well-clear violation) is bound to happen at a predetermined point on the ownship's flight path segment. A single simulation, or trial, is run by sampling an initial condition for the intruder that could result in a well-clear violation for the ownship at some point along its flight path. A trial is said to be complete when the ownship reaches the destination waypoint. In case the ownship does not reach the final destination, the trial is considered complete after a predefined amount of time.

When a conflict occurs, the relative positions and velocities between the intruder and ownship are used to query the network described in *Policy Parameterization* (Sec. IV-B). The network in turn outputs a distribution over the actions. An action is sampled according to this distribution and executed to resolve the well-clear conflict. The position and velocities of the intruder are recorded for each trial. At the end of each trial, the reward function described in Sec. IV-C can be used to compute the effectiveness of the resolution. In this work, the reward functions are defined as terminal reward values, i.e., only received after each trial. Consequently, the cumulative reward is given by Equation (11). The discount factor γ is set to 1.

The data collected from running a batch of trials are then used to adjust the weights of the network. The training error of the network is defined by the cross-entropy loss:

$$\texttt{loss} = -\sum_{i=0}^{N} a_i \log(p_i) R_i, \quad (12)$$

where N represents the number of trials in a given batch, a_i represents the action that was selected in i$^{\text{th}}$ trial and p_i represents the probability of action a predicted by the neural network in the the i$^{\text{th}}$ trial. R_i represents the cumulative reward obtained in the i$^{\text{th}}$ trial. The gradient of the above loss function ensures that the network weights are adjusted so that actions that lead to higher cumulative rewards become more likely consequently reducing the probability of actions that lead to lower cumulative rewards. Training is repeated for several batches until the average cumulative reward for a given batch is above a desired threshold.

V. Results

To illustrate the effectiveness of the reinforcement framework described in the previous section, a subset of initial conditions that result in loss of separation are selected. Figure

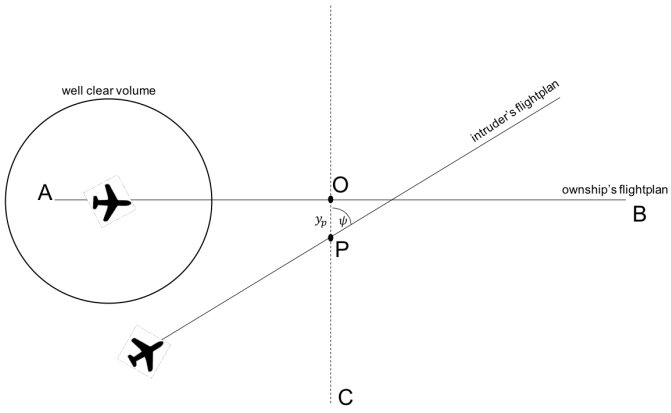

Fig. 3: Set Up for simulating encounters

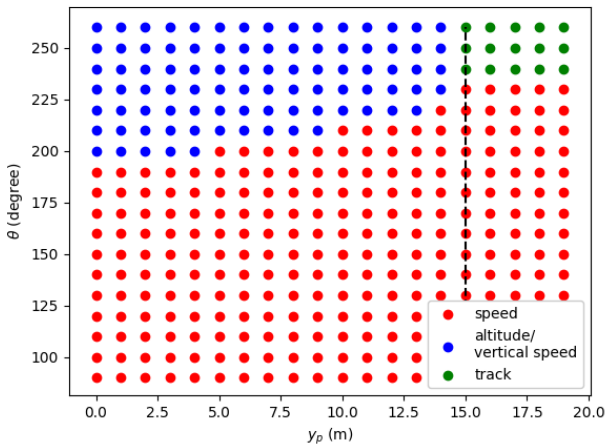

Fig. 4: Network outputs

3 illustrates the parameters that govern the sampling of these initial conditions. The intruder's range and bearing are chosen such that it reaches some point P on line OC when the ownship is at point O. Point P is selected to lie within a distance of the well-clear volume radius. This ensures that a loss of separation happens when the ownship is at O (assuming no resolutions are executed). The distance OP is referred to as the intercept distance y_p, and the angle ψ represents the heading of the intruder. In this setting, the ownship is always flying from point A to B, i.e., at a track angle of 90 degree relative to true north. Initial conditions for the intruder are chosen based on the randomly sampled values for ψ and y_p. When $y_p = 0$, a perfect collision occurs regardless of the intruder's track. $y_p \neq 0$ and $\psi = 90$ or $\psi = 270$ represent intruder flight paths that are parallel to the ownship flight path. For the purpose of illustration, a cylindrical well-clear volume of radius $20m$ and height $5m$ is selected.

Although using a single policy π to capture policy variations across the entire input space may be possible by employing an appropriate network architecture, for the purpose of illustration, however, the input space is partitioned into several subsets such that each subset is associated with a corresponding policy, resulting in a multi-parameter policy. This approach is akin to the idea of gain scheduling of flight controller gains in typical autopilot design.

Figure 4 illustrates the policy's prediction for initial conditions generated by sampling the two parameters ψ and y_p. Initial conditions resulting from $y_p < 15m$ are governed by policy π_1 and $y_p > 15m$ are governed by π_2.

When $y_p < 15m$, speed resolutions are favored by π_1 for $\psi \in [90, 220]$. According to Equation (11), speed resolutions yield the highest reward in this regime because it is possible to avoid loss of separation while avoiding any flight plan deviations. However, when $\psi \in [220, 270]$ (i.e., roughly head-on encounter geometries), this results in encounters where the intruder encroaches into the well-clear volume of the ownship regardless of any changes in speed prescribed by the speed resolution. Consequently, the policy π_1 favors vertical resolutions (altitude or vertical speed). According to Equation (11), track resolutions executed in this regime results in larger deviations from the flight plan compared to altitude resolutions, making them less favorable than vertical resolutions.

When $y_p > 15m$, track resolutions become more favorable for head on encounters according to π_2. In this regime, separation from the intruder is facilitated by minimal deviation from the flight plan using track resolutions. Although vertical resolutions would enable well-clear separation, these resolutions result in larger deviations from the flight plan in this regime.

The policy learned by the RL framework described in this work depends on the well-clear configuration parameters used by the detection logic in DAIDALUS. The well-clear parameters determine the size of the well-clear volume, the look ahead time, the vehicle maneuvering characteristics etc. Thus the policy should be determined using a well-clear configuration appropriate for the ownship and the mission.

VI. Conclusions

This paper illustrated the use of a Reinforcement Learning framework within the context of the ICAROUS architecture to select traffic avoidance options that satisfy mission specific optimization goals. This RL framework is used to compute a policy that would enable a UAS to autonomously select a suitable resolution from different types of resolutions prescribed by the Sense and Avoid algorithm called DAIDALUS. The policy learned by the RL framework proposed in this paper would prevent a well-clear violation while ensuring mission progress and minimum deviation from the flight plan. Furthermore, this method preserves the formal properties of each resolution guaranteed by DAIDALUS.

This work parameterized the RL policy to select an SAA resolution using fully connected neural networks. The input space was partitioned into subsets such that each subset was associated with a policy resulting in a database of policies - each optimized to select the optimal resolution for its input space. Results, as shown in Figure 4, indicate

sensible resolutions consistent with the reward formulation in Equation (11). This multiple policy approach helps capture variations in resolutions across a given input space more efficiently than a single policy would have. Alternately, other methods such as Support Vector Machines (SVM) could also be used to parameterize the policy, which remains an area of future investigation.

The RL formulation described in this work focused on policies for a single intruder whose current state could result in a loss of separation. Increasing the number of intruders that the policy can handle, although possible, can be computationally expensive.

REFERENCES

[1] S. Owre, J. M. Rushby, and N. Shankar, "Pvs: A prototype verification system," in *International Conference on Automated Deduction*. Springer, 1992, pp. 748–752.

[2] C. Muñoz, A. Narkawicz, G. Hagen, J. Upchurch, A. Dutle, and M. Consiglio, "DAIDALUS: Detect and Avoid Alerting Logic for Unmanned Systems," in *Proceedings of the 34th Digital Avionics Systems Conference (DASC 2015)*, Prague, Czech Republic, September 2015.

[3] A. Narkawicz, C. Muñoz, and A. Dutle, "Sensor uncertainty mitigation and dynamic well clear volumes in DAIDALUS," in *Proceedings of the 37th Digital Avionics Systems Conference (DASC 2018)*, London, England, UK, September 2018.

[4] RTCA, "Do-365 minimum operational performance standards (MOPS) for detect and avoid (DAA) systems," May 2017.

[5] M. Consiglio, C. Muñoz, G. Hagen, A. Narkawicz, and S. Balachandran, "Icarous: Integrated configurable algorithms for reliable operations of unmanned systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–5.

[6] S. Balachandran, C. Muñoz, M. Consiglio, M. Feliú, and A. Patel, "Independent configurable architecture for reliable operation of unmanned systems with distributed on-board services," in *Proceedings of the 37th Digital Avionics Systems Conference (DASC 2018)*, London, England, UK, September 2018.

[7] RTCA SC-147, "RTCA-DO-185B, Minimum operational performance standards for traffic alert and collision avoidance system II (TCAS II)," July 2009.

[8] J. Lygeros and N. Lynch, "On the formal verification of the TCAS conflict resolution algorithms," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 2. IEEE, 1997, pp. 1829–1834.

[9] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-generation airborne collision avoidance system," *Lincoln Laboratory Journal*, vol. 19, 2012.

[10] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, and A. Platzer, "Formal verification of ACAS X, an industrial airborne collision avoidance system," in *Proceedings of the 12th International Conference on Embedded Software*. IEEE Press, 2015, pp. 127–136.

[11] G. Manfredi and Y. Jestin, "An introduction to ACAS Xu and the challenges ahead," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016, pp. 1–9.

[12] M. Consiglio, B. Duffy, S. Balachandran, C. Muñoz, and L. Glaab, "Sense and avoid characterization of the independent configurable architecture for reliable operations of unmanned systems," in *Proceedings of the 13th USA/Europe Air Traffic Management R&D Seminar, ATM 2019*, 2019.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[14] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.