A Supervisory Neural Network and Fuzzy Inference System Approach to an Unmanned Aerial Vehicle Tail-Chase Scenario

Timothy Arnett*, Kelly Cohen†, and Brandon Cook‡§
University of Cincinnati, Cincinnati, OH, 45220

David Casbeer¶
Controls Center of Excellence, Air Force Research Laboratory, WPAFB, OH

As the capabilities and implementation of multiple Unmanned Aerial Systems (UAS) operations increase, the need to develop reliable, verifiable, and high-performing vehicle control algorithms arises. There are several methods for achieving near optimal navigation control in multiple UAS scenarios. However, many approaches have difficulty identifying boundaries in the state-space where discontinuities in the control signal exist. Neural Networks (NNs) have been shown to be universal approximators of these decision boundaries and can therefore classify the space into discrete regions with differing lower level actions. In this study, a control system for a one-on-one UAS Tail-Chase scenario is developed. An NN was used to define a decision boundary for discrete selection of two different Fuzzy Inference Systems (FISs) for navigation and avoidance control. The parameters for both the NN and FISs are found using a Genetic Algorithm (GA) inside a custom simulation environment. After initial training, uncertainty was included in vehicle movements to improve generalization. The simulation results show that the system was successful in all test cases after adding uncertainty and demonstrate the efficacy of this approach.

I. Nomenclature

μ	=	membership value
ρ	=	turn rate noise
τ	=	simulation termination time
ψ	=	heading of vehicle
$\dot{\psi}$	=	turn rate of vehicle
c	=	input membership function center points
d	=	distance
e	=	error
E	=	scaled error
g	=	gravitational acceleration
h	=	Neural Network intermediate output (post-activation)
J	=	cost function value
k	=	scaling constant
n	=	number of simulation runs
N	=	load factor
R	=	turn radius
s	=	Neural Network layer aggregate
t	=	simulation time
u	=	control output
U	=	output membership function center points

*Graduate Student, Aerospace Engineering and Engineering Mechanics, 745 Baldwin Hall, AIAA Student Member.
†Professor, Aerospace Engineering and Engineering Mechanics, 745 Baldwin Hall, AIAA Associate Fellow.
‡Graduate Student, Aerospace Engineering and Engineering Mechanics, 745 Baldwin Hall, AIAA Student Member.
§Research Aerospace Engineer, NASA Ames Research Center
¶Technical Team Lead, AFRL, AIAA Associate Fellow.

v = UAS speed
w = Neural Network weights
x = Cartesian x location
$\dot{x}$ = speed of vehicle along Cartesian x direction
X = Neural Network input vector
y = Cartesian y location
$\dot{y}$ = speed of vehicle along Cartesian y direction
$\hat{y}$ = raw Neural Network output
$\bar{y}$ = piecewise decision boundary of Neural Network output

Subscripts

$avoid$ = avoidance mode
C = collision
i = simulation run index
I = intersection point
j = UAS index
l = Neural Network layer index
M = margin
nav = navigation mode
P = pursuit vehicle
PR = penalty region
RA = relative angle
RH = relative heading
RR = reward region
T = target vehicle

II. Introduction

As cooperative multiple Unmanned Aerial Systems (UAS) operations become more realizable, there is a need to develop reliable, verifiable, and high-performing algorithms for multiple vehicle control. One particular problem of interest is a Tail-Chase scenario, where one or more vehicles try to reach a desired state relative to another. For example, leader-follower swarming behavior can utilize this to allow flight formations. There are several solutions for Tail-Chase vehicle control in low number scenarios, such as one-on-one, two-on-one, or even one-on-multiple [1–3]. The main difficulty when solving these types of problems is finding the boundaries in the state-space where switching actions lie. For the application of a Tail-Chase scenario, the UAS controller must continuously decide the direction and magnitude to turn to reach a desired region relative to the other vehicle. One way to partition the space effectively is to use classification methods as a supervisory, outer loop control. The classes then determine which inner loop control system to enact. Neural Networks (NNs) have been shown to be both universal approximators and to be able to perform highly nonlinear classification with one or more hidden layers and nonlinear activation functions [4, 5].

There are also systems that can be used effectively as both inner and outer loop control, at the cost of complexity. These systems are able to approximate a continuous function to any arbitrary degree of accuracy [6–8]. However, these systems have less transparency than a hierarchical system that uses supervisory decision-making. In addition, they may have difficulties approximating discontinuities efficiently. As there have been recent efforts to formally verify NNs [9–11], the solution presented in this paper could be promising for complex UAS scenarios where high confidence in correct behavior is required. Additionally, as we know that discontinuities likely exist in the optimal control function, it may be more computationally efficient to have a supervisory logic that defines these boundaries (explicitly or implicitly) and subsequently use a continuous universal approximator on the compact set defined therein.

Developing an effective Tail-Chase controller is further complicated by the fact that the control algorithm is contingent on the other vehicle's behavior. For example, if two vehicles have the same Tail-Chase controller, neither vehicle will succeed in reaching a desired region without a performance advantage; i.e., vehicles with the exact same performance (speed, load factor, etc.) will not be able to close the distance to an arbitrary desired region. Additionally, one potential way to make a controller more robust to unknown behaviors is to introduce uncertainty into the target vehicle's movement. This prevents overfitting a solution to particular conditions and therefore improves confidence in

the system's ability to generalize.

This paper proposes a solution that has several advantages over other approaches. First, the ability to identify decision boundaries implicitly in the input space avoids the need to define them *a priori* as would need to be done for an analytical solution. Secondly, the scenario can be modified to include any arbitrary desired penalty and reward regions relative to another vehicle. This allows the accommodation of much more complex systems and behaviors while retaining tractability. Lastly, this closed-loop control method is robust to unknown behaviors in the target UAS. The solution continuously reacts to current state information and is trained to account for a certain level of state uncertainty. The benefit of utilizing uncertainty during training is that it allows for a smaller set of training cases while improving the ability to generalize over the state-space.

In Section III the specific problem being addressed is described. This is followed by a detailed explanation of the methodology of the proposed solution in Section IV. In Section V the results of the study are presented. Lastly, conclusions and future opportunities for expanding on this work are presented in Section VI.

III. Problem Description

The problem being considered is a one-versus-one Tail-Chase scenario where one UAS, the *pursuer*, is attempting to reach a Reward Region (RR) relative to another UAS, the *target*, while avoiding a Penalty Region (PR). The PR and RR are defined by ranges of the relative states of the two vehicles. The relatives states considered are the distance, d, the relative angle, ψ_{RA_P}, and the relative heading, ψ_{RH_P}. The relative states d, ψ_{RA_p}, and ψ_{RH_p} are defined in Eqs. (8), (9), and (10), respectively. Specifically, RR = $\{\mathbb{R}|d \leq 50,\ -\frac{\pi}{6} \leq \psi_{RA_P} \leq \frac{\pi}{6},\ -\frac{\pi}{6} \leq \psi_{RH_P} \leq \frac{\pi}{6}\}$. The PR is relative to the target vehicle such that PR = $\{\mathbb{R}|d \leq 50,\ -\frac{\pi}{6} \leq \psi_{RA_T} \leq \frac{\pi}{6}\}$. A visualization of this is shown in Fig. 1.

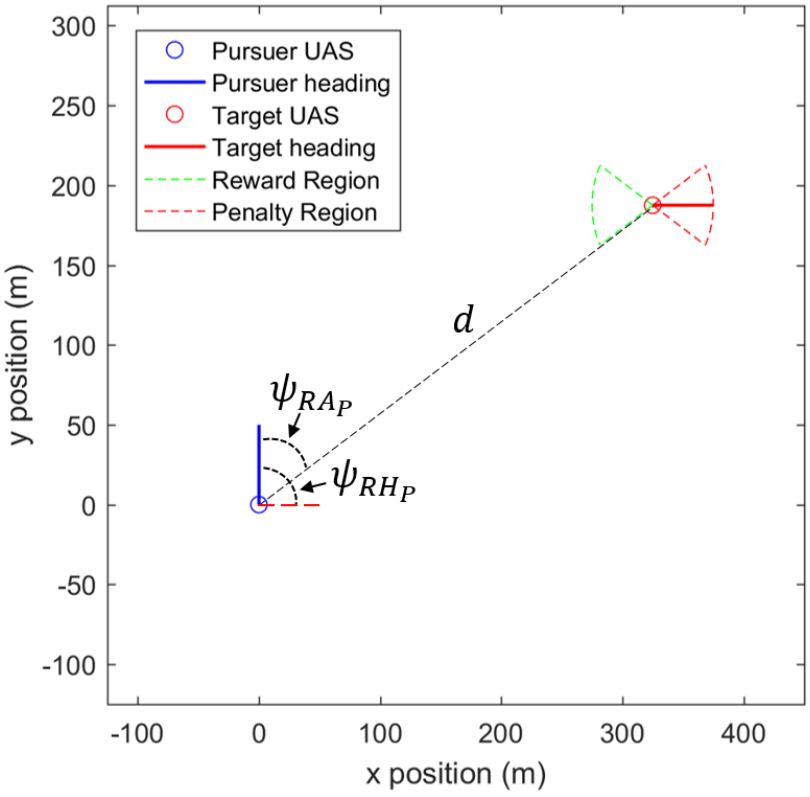

Fig. 1 Pursuer and Target UAS with Penalty and Reward Regions

The vehicles are constrained to 2-D level flight with constant speed and maximum turn rates, $\dot{\psi}_{max_j}$. The values were set such that the pursuer has an advantage in load factor and speed while keeping the same minimum turn radius, R_{min}. The maximum turn rates and minimum turn radii were calculated from Eqs. (1)-(2) with values of $v_P = 50\frac{m}{s}$ and

$v_T = 20\frac{m}{s}$ for the pursuer and target, respectively. This turns the vehicles into Dubins vehicles [12] and the dynamic model in global coordinates is given by Eqs. (3)-(5). These can then be integrated to obtain the global states, x_j, y_j, and ψ_j at each time step.

$$\dot{\psi}_{max_j} = \frac{g}{v_j\sqrt{N_j{}^2 - 1}},\ j = P, T \tag{1}$$

$$R_{min} = \frac{v_j}{\dot{\psi}_{max_j}} \tag{2}$$

$$\dot{x}_j = v_j cos(\psi_j) \tag{3}$$

$$\dot{y}_j = v_j sin(\psi_j) \tag{4}$$

$$\dot{\psi}_j = u_j\ ,\ \ u_j \in \left[-\dot{\psi}_{max_j}, \dot{\psi}_{max_j}\right] \tag{5}$$

In order to develop a controller with fewer required states, the relative states of the vehicles, d, ψ_{RH}, and ψ_{RA}, are calculated from the global states. These are shown in Eqs. (6)-(10).

$$e_{x_P} = x_T - x_P \tag{6}$$

$$e_{y_P} = y_T - y_P \tag{7}$$

$$d = \sqrt{e_{x_P}^2 + e_{y_P}^2} \tag{8}$$

$$\psi_{RA_P} = \tan^{-1}\left(\frac{e_{y_P}}{e_{x_P}}\right) \tag{9}$$

$$\psi_{RH_P} = \psi_T - \psi_P \tag{10}$$

ψ_{RH_P} and ψ_{RA_P} are then corrected such that $\psi_{RH_P},\ \psi_{RA_P} \in \{\mathbb{R} | -\pi \leq \psi_{RH_P},\ \psi_{RA_P} \leq \pi\}$. This information shall be used to direct the pursuer to the RR without entering into the PR. To achieve this goal, a controller must use the relative state information to determine the appropriate corresponding turn rate.

IV. Proposed Solution

The proposed solution to this problem is to use a Neural Network to partition the state-space into discrete regions. Inside these regions, either a navigation controller or an avoidance controller will be employed to control the path of the vehicle. These two controllers direct the pursuer to either steer towards or away from the target. The controller for navigating towards the target is shown in Eq. (11).

$$\dot{\psi}_{nav_P} = \dot{\psi}_{max_P}\ sgn(\psi_{RA_P}) \tag{11}$$

The avoidance controller is more complicated due to needing more information about the relative states of the two UAS and the possible modes for steering away. This controller and its corresponding logic is detailed in Section IV.A.

A. Avoidance

The avoidance controller was developed with inspiration from the methods used in Ref. [13, 14]. It uses a similar logic based on the intersection point of the UAS. The intersection point is defined as the point of intersection of the projected current headings between the two UAS.

The distance to the intersection point, d_{I_j}, is then calculated by Eq. (12)-(14). A visualization of the intersection point (x_I, y_I) and the distances d_{I_P} and d_{I_T} are shown in Figure 2.

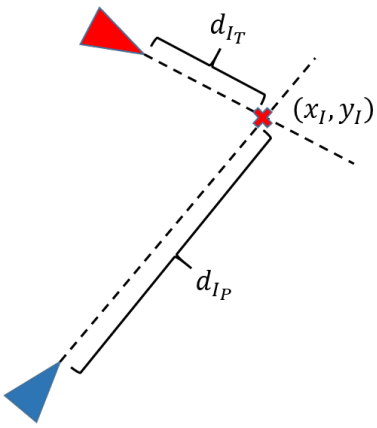

Fig. 2 Intersection point and distances from UAS

This distance is used to determine if the pursuit UAS should go in front of or behind the target UAS using the relationships shown in Eq. (15). An intersection distance margin, denoted d_M, is used in order to decide if there is sufficient distance to safely go in front of the target UAS.

$$x_I = \frac{y_P - y_T - x_P \tan(\psi_P) + x_T \tan(\psi_T)}{\tan(\psi_T) - \tan(\psi_P)} \tag{12}$$

$$y_I = \frac{y_P \tan(\psi_T) - y_T \tan(\psi_P) + (x_T - x_P)\tan(\psi_P)\tan(\psi_T)}{\tan(\psi_T) - \tan(\psi_P)} \tag{13}$$

$$d_{I_j} = \sqrt{(x_I - x_j)^2 + (y_I - y_j)^2} \tag{14}$$

$$front = \begin{cases} 1 & \text{if } (d_{I_P} + d_M) < d_{I_T} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

After the UAS decides if it should go in front, it then selects an appropriate Fuzzy Logic Controller (FLC) that determines the actual continuous output, $\dot{\psi}$. Note that FLCs are simply Fuzzy Inference Systems (FISs) that are specifically applied to control problems. The FLCs each utilize triangular membership functions, Ruspini partitioning, input normalization, the product method for rule association, and weighted average defuzzification, as desribed in Ref. [15]. This results in an output that is of the form shown in Eqs. (16)-(18). A general FLC structure and internal processes is shown in Fig. 3. The input set, $\boldsymbol{c}$, and output set, $\boldsymbol{U}$, have the forms shown in Eq. (19) and (20), respectively. (Note: a bold variable indicates it is a matrix of values.) Due to the properties of the FLCs in question, these sets represent the center points of their triangular membership functions. The values of these parameters, as well as the normalization and scaling gains, k_{in} and k_{out}, were found using the methods described in Section IV.C. Note that, due to normalization, the values for c_1 and c_4 were set to -1 and 1, respectively.

$$\dot{\psi}_{avoid_P} = k_{out} \sum_{i=1}^{3} \mu_i U_i \tag{16}$$

$$\mu_i = \frac{\frac{\psi_{RA}}{k_{in}} - c_i}{c_{i+1} - c_i} \tag{17}$$

$$\mu_{i+1} = \frac{c_{i+1} - \frac{\psi_{RA}}{k_{in}}}{c_{i+1} - c_i} \tag{18}$$

$$\boldsymbol{c} = [\, c_1 \; c_2 \; c_3 \; c_4] \tag{19}$$

$$\boldsymbol{U} = [U_1 \; U_2 \; U_3 \; U_4] \tag{20}$$

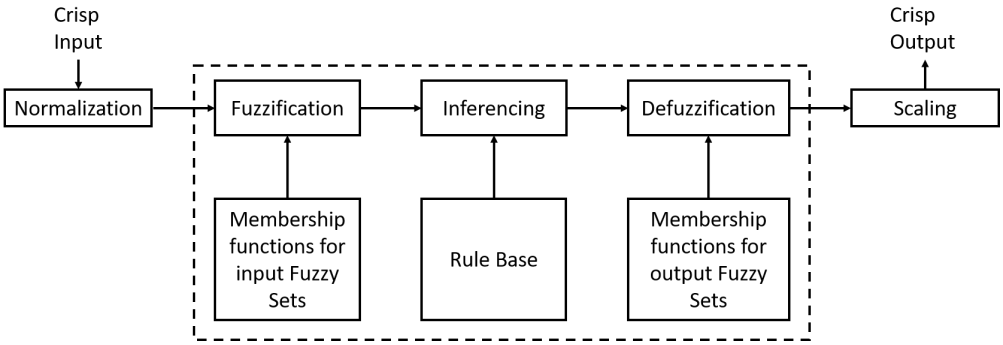

Fig. 3 FLC structure and internal processes

B. Neural Network

The constructed Neural Network is a fully-connected, feed-forward network with two hidden layers. NNs with multiple hidden layers have been shown to be universal approximators for nonlinear classification decision boundaries [16]. The hidden layers each contain five nodes, whereas the output layer has a single node. The activation functions within the hidden layers are hyperbolic tangent functions and the final output layer uses the sigmoid activation function. These functions are not typically used in deep-learning NN applications due to the problem of vanishing gradients. However, in this work a type of evolutionary reinforcement learning is used, and is therefore derivative-free. One benefit of using these atypical activation functions (such as a Rectified Linear Unit (ReLU) function) is that there have been recent efforts to formally verify NNs with ReLU activation [9]. This could help increase the confidence that the system would behave as intended over the entire input domain.

The inputs to the NN are d, ψ_{RH}, and ψ_{RA}. These inputs as well as a bias value are combined into vector form, denoted $\boldsymbol{X}$. The output, $\hat{y}$, is converted into a binary value, $\bar{y}$, by thresholding the output at 0.5. This allows the inputs to be separated into the two classes: avoidance and navigation. If the inputs are classified as avoidance, the avoidance subcontroller is activated, otherwise, the navigation controller is activated. The operations that occur within the NN, along with the activation in each layer, are shown in Eq. (21)-(24). (Note: s_3 is a scalar due to having a single output in the last neural network layer.) A visual representation of the NN architecture and pursuer logic with FLCs are shown in Fig. 4 and Fig. 5, respectively. Note that the bias is not depicted in Fig. 4.

$$\boldsymbol{s_l} = \begin{cases} \boldsymbol{w_l}^T \boldsymbol{X} & \text{if } l = 1 \\ \boldsymbol{w_l}^T \boldsymbol{h_{l-1}} & \text{if } l = 2, 3 \end{cases} \tag{21}$$

$$\boldsymbol{h_l} = \tanh(\boldsymbol{s_l}), \; l = 1, 2 \tag{22}$$

$$\hat{y} = \frac{1}{1 + e^{-s_3}} \tag{23}$$

$$\bar{y} = \begin{cases} avoidance & \text{if } \hat{y} > 0.5 \\ navigation & \text{otherwise} \end{cases} \tag{24}$$

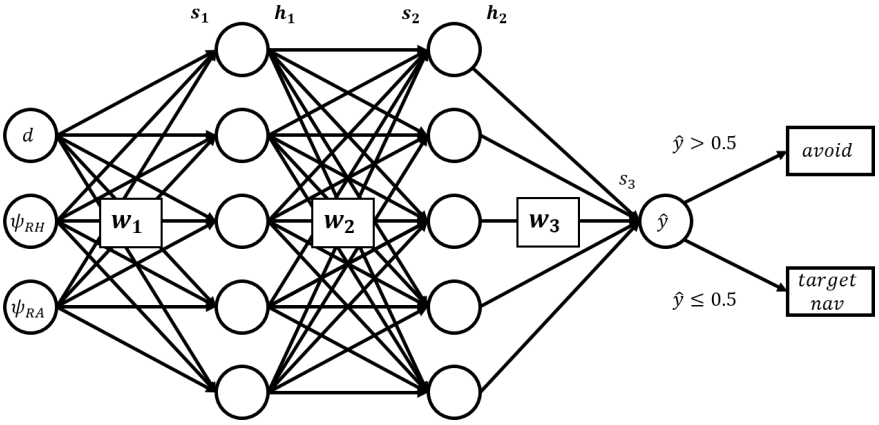

Fig. 4 Hybrid NN and FIS structure for supervisory control

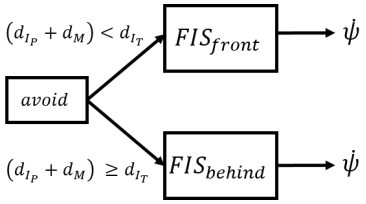

Fig. 5 Avoidance logic structure

Now that the structure of the control system has been created, the various parameters need to be found.

C. Learning Control Parameters

In order to learn the parameters of this system, a Genetic Algorithm (GA) was utilized. This method was chosen because the optimal control is unknown and therefore actual error data are not available for supervised learning methods. Thus, a derivative-free search method with a critic, or cost, function is needed. The parameters to be learned are the membership function center points $\boldsymbol{c}$ and $\boldsymbol{U}$, the gains for the FLCs k_{in} and k_{out}, and the NN weights $\boldsymbol{w_l}$. This constitutes an *individual* in the population that is being optimized by the GA. Better performing (i.e. lower cost) individuals are selected for mutation and recombination in order to iteratively refine the population towards a global optimum. The actual GA used was the default algorithm included in the Global Optimization Toolbox from Mathworks [17].

To facilitate this, a simulation environment was created based on the vehicle properties described in Section III. The control system structure was defined and tested over a number of initial conditions. These conditions were such that the target vehicle started at fifty different values of ψ_{RA_P} in $[-\pi, \pi]$, two different values for ψ_P at $(0, \pi)$, and an initial separation of 375 m. This gives one hundred trials of initial condition values for a single test run. Since the dynamics of the vehicle are functions of $\dot{\psi}$ (either directly or through coupling), these initial conditions were deemed sufficient for covering the state-space.

A cost function is needed to assign a performance metric value to a particular test run. In terms of a GA, this is the *fitness* value for a particular individual in the population. The only difference here is that the value is being minimized, so it is conventionally labeled as a cost. The cost function is shown in Eqs. (25)-(29). The value used for the collision distance, d_C, was 10 meters.

$$J = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\tau_i}\left(E_{x_i}(t)^2 + E_{y_i}(t)^2\right) + 100000\ k_{C_i} \tag{25}$$

$$E_{x_i}(t) = k_{PR}(t)e_{x_i}(t) \tag{26}$$

$$E_{y_i}(t) = k_{PR}(t) e_{y_i}(t) \tag{27}$$

$$k_{PR}(t) = \begin{cases} 1000 & \text{if in PR} \\ 1 & \text{otherwise} \end{cases} \tag{28}$$

$$k_{C_i} = \begin{cases} 1 & \text{if } d < d_C \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

This cost function therefore penalizes the pursuer for each time step that it exists within the PR. Although there is no explicit term that rewards it for reaching the RR, this happens implicitly by stopping the simulation once it is reached. Once the simulation terminates, the error terms in Eqs. (26)-(27) will no longer be accumulating. This is related to the path length, and therefore the final time, due to having a constant speed. Note that the simulation does have a maximum time limit which will terminate that trial if reached. The final term in Eq. (25) is a penalty for collisions. If there is a collision then the pursuer is assigned an extremely heavy penalty. This value is arbitrary, but was selected after empirical examination of early runs of the optimization algorithm. The value chosen was deemed sufficient to dominate the cost if a collision were to occur. An occurrence of a collision also constitutes a terminating condition for the simulation. Overall, this cost function describes a controller that drives the pursuer to get to the RR as quickly as possible while avoiding the PR and collisions. The cost function value is stored for each trial in a particular test run and then averaged to get the final cost for that particular individual in the population.

D. Uncertainty in Target Vehicle Movement

One concern with learning models numerically is overfitting to the training data. Additionally, although the training data may offer relatively dense coverage of the state-space, the cost function may also inadvertently drive the system towards overfitting. To help combat this, uncertainty was added to the system in the form of noise to the target vehicle's input. Initially, the system was tested in a scenario where the target vehicle's input, u_T, was set to be zero (i.e. it flies along a straight path). This ensured that the problem could be solved for a more simplified scenario. To help improve the robustness of the developed system, uncertainty was then added to the target's turn rate. This helped ensure better generalization in the learned model by accounting for possible states the vehicle can reach. This was achieved by augmenting Eq. (5) with a noise term, ρ, as shown in Eq. (30).

$$\dot{\psi}_T = u_T + \rho \tag{30}$$

The noise was added by sampling a truncated normal distribution, as described in Ref. [18], with zero mean and truncation points at $\pm\dot{\psi}_{max_T}$. The standard deviation of the distribution was chosen to be $\frac{\dot{\psi}_{max_T}}{3}$. While this was used throughout the scope of this study, there are potential benefits to testing the system with larger standard deviation values. For example, an increased standard deviation would force the controller to be more conservative due to an increased probability of higher turn rates.

V. Results

The methodology described in Section IV was implemented for a number of additional test cases. An example of the cost from Eq. (25) during the learning portion of the development is shown in Fig. 6. As the GA modified the parameters of the system, it can be seen that the mean fitness value decreased. This shows the search honing in on more fit individuals as higher cost solutions are removed. The final values found for all weights, gains, and membership functions are not displayed, but can be provided by the authors upon request.

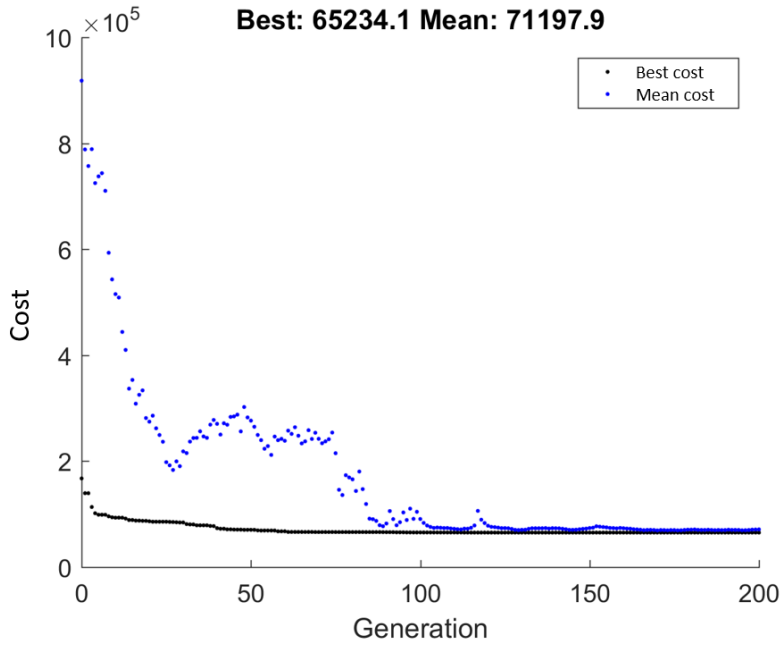

Fig. 6 Average and best individual cost vs. generation during GA learning

Figure 7 shows the performance of the system, with no turn rate uncertainty, found through learning. The same initial conditions, as described in Section IV.C, are shown. It can be seen that in all cases, the pursuer successfully reaches the RR and avoids the PR. To further validate the system performance and generalization capabilities, many more initial conditions were tested that were not seen during training. By increasing the number of trials tested, the performance of the system with a higher granularity with respect to the state-space can be examined. A total of 10,000 trials were run with varying initial relative positions and headings between the target and pursuer. It was discovered that there were collisions in 1.48% of these cases. These collisions occurred in a few concentrated ranges of initial relative angles: $\{\mathbb{R}| -179° \leq \psi_{RA_P} \leq -172°, -143° \leq \psi_{RA_P} \leq -137°, 171° \leq \psi_{RA_P} \leq 179°\}$. These collisions could be due to the NN selecting the wrong subcontroller, the FLC subcontrollers learning incorrect behaviors, or some combination of both. To further improve the performance of the controller, additional training could be performed in order to expand coverage of the input space. Outside of the collision cases, the pursuer always successfully reached the RR.

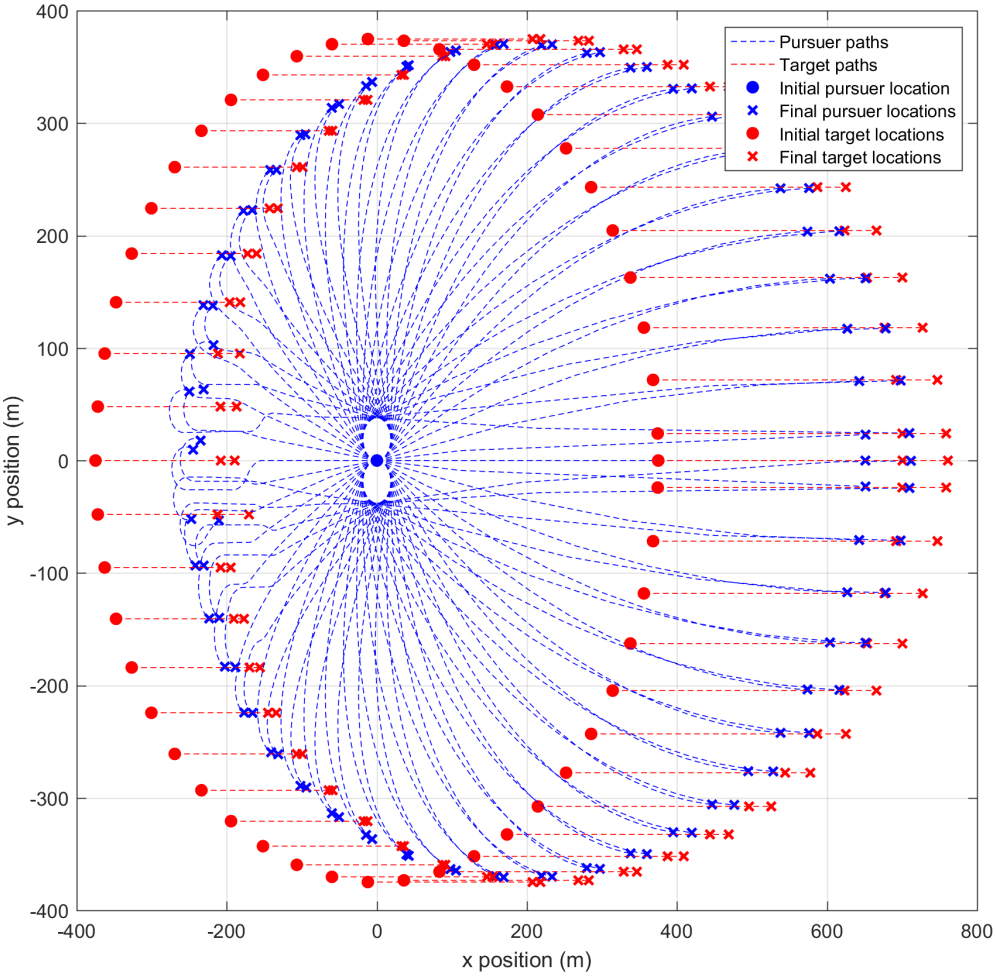

Fig. 7 Simulation results for a test run of the best individual found during learning (no uncertainty)

Once the system had been trained for the case with no uncertainty, uncertainty was added to the target UAS's turn rate, as described by Eq. (30). The same parameters were then again trained using a GA. For this second round of training, the final solution from the no uncertainty case served as an individual in the initial population for the uncertainty case.

Figure 8 shows the system performance after being trained with uncertainty in the target UAS's turn rate. Note that although the system was trained such that the target UAS was performing a random walk, in order to compare these results to those shown in Fig. 7, the same test cases were used. That is, there is no uncertainty in the target's turn rate in the test cases shown. It can be seen from the results that the system is still able to reach the RR in all trials tested. The same 10,000 trials were also evaluated as in the no uncertainty case. No collisions occurred in these testing instances. Thus, the additional training with uncertainty in the target's turn rate lead to a more robust control system.

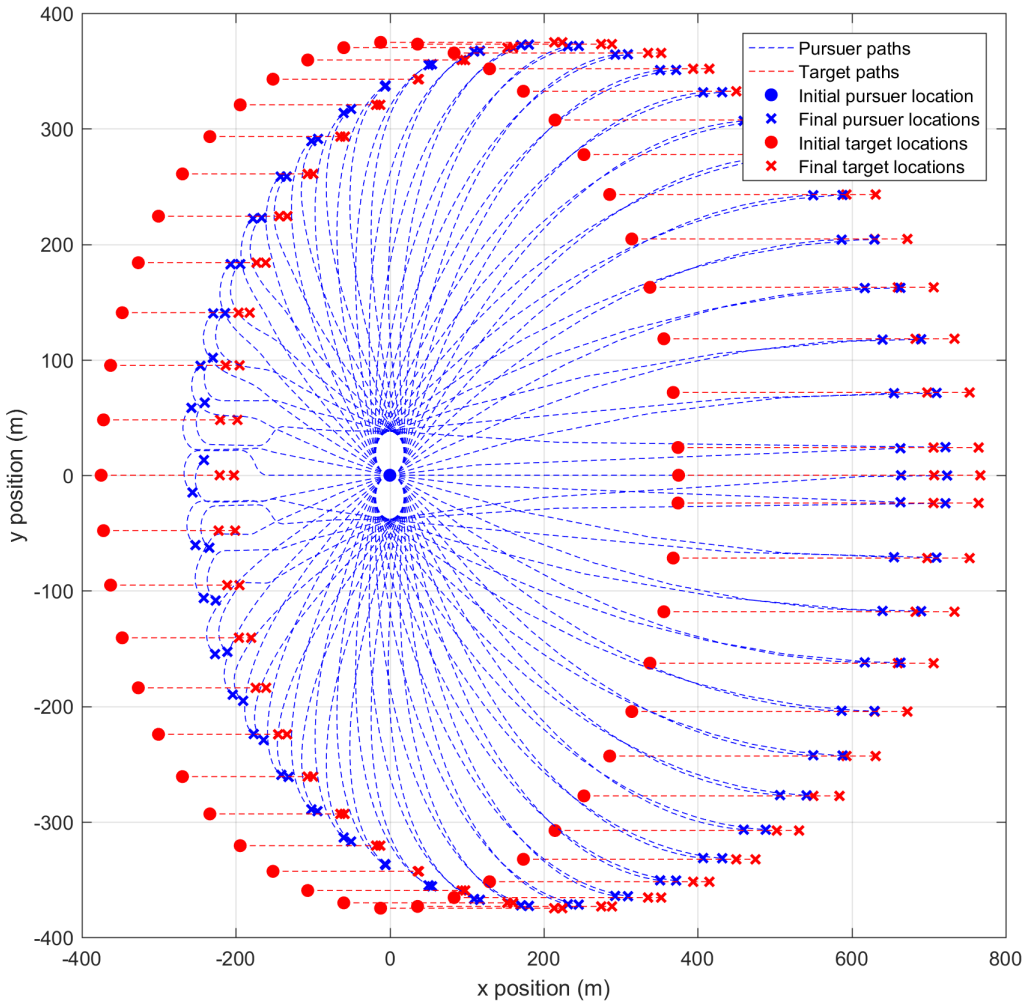

Fig. 8 Simulation results for a test run of the best individual found during learning (with uncertainty)

VI. Conclusion

In conclusion, a method was presented for obtaining near optimal solutions to a one-versus-one Tail-Chase UAS scenario by combining an NN supervisor with lower level FLC control. A GA was utilized in order to learn the parameters of both the FLCs and the NN with no precise knowledge about the target UAS or its PR. The methodology could be extended for any arbitrary one-versus-one Tail-Chase scenario by adjusting the RR, PR, and cost function. Overall, this represents a novel method for finding discontinuities in a complex input space and then enacting appropriate low-level continuous control. One possible application for this type of control includes leader-follower formation control in swarming agent systems.

This approach could potentially be scaled up for one-on-multiple or even multiple-on-multiple UAS scenarios. This would require other supervisory and complementary functions in order to provide target assignments to the pursuing vehicles. Also, to improve scalability, another function is needed to ignore potential target vehicles outside of a sphere of influence. This could also be written into the target assignment protocol. Additionally, the cost function would need to be modified in order to account for multiple penalty regions.

To improve the confidence in correctness, the Neural Network could use other activation functions that are more amenable towards formal verification, such as ReLU. Being able to definitively show that there are no collision cases and the pursuer always reaches the reward region, regardless of state value, is invaluable for ensuring system correctness and effectiveness. The approach could also be extended by considering other inner loop control methods for different

scenarios. Furthermore, in cases where the inner loop control reduces to a multiple classification problem (discrete control), the NN output could instead be used directly for control using a Softmax function. Lastly, in an effort to validate the algorithms and methods presented within, the algorithms could be tested on actual flight systems. These flight systems would be small, fixed-wing RC aircraft controlled by sending waypoint updates from a ground station. The test results would then be collected, examined, and compared with the simulation results.

Acknowledgments

This research was conducted with Government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.

References

[1] Chung, C. F., and Furukawa, T., "A reachability-based strategy for the time-optimal control of autonomous pursuers," *Engineering Optimization*, Vol. 40, No. 1, 2008, pp. 67–93.

[2] Hashemi, A., Casbeer, D. W., and Milutinović, D., "Scalable value approximation for multiple target tail-chase with collision avoidance," *Decision and Control (CDC), 2016 IEEE 55th Conference on*, IEEE, 2016, pp. 2543–2548.

[3] Milutinović, D., Casbeer, D. W., and Pachter, M., "Markov inequality rule for switching among time optimal controllers in a multiple vehicle intercept problem," *Automatica*, Vol. 87, 2018, pp. 274–280.

[4] Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," *Neural networks*, Vol. 2, No. 5, 1989, pp. 359–366.

[5] Hornik, K., "Approximation capabilities of multilayer feedforward networks," *Neural networks*, Vol. 4, No. 2, 1991, pp. 251–257.

[6] Zadeh, L. A., "Fuzzy logic, neural networks, and soft computing," *Communications of the ACM*, Vol. 37, No. 3, 1994, pp. 77–85.

[7] Wang, L.-X., "Fuzzy systems are universal approximators," *Fuzzy Systems, 1992., IEEE International Conference on*, IEEE, 1992, pp. 1163–1170.

[8] Castro, J. L., and Delgado, M., "Fuzzy systems with defuzzification are universal approximators," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 26, No. 1, 1996, pp. 149–152.

[9] Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J., "Reluplex: An efficient SMT solver for verifying deep neural networks," *International Conference on Computer Aided Verification*, Springer, 2017, pp. 97–117.

[10] Ehlers, R., "Formal verification of piece-wise linear feed-forward neural networks," *International Symposium on Automated Technology for Verification and Analysis*, Springer, 2017, pp. 269–286.

[11] Kuper, L., Katz, G., Gottschlich, J., Julian, K., Barrett, C., and Kochenderfer, M., "Toward Scalable Verification for Safety-Critical Deep Networks," *arXiv preprint arXiv:1801.05950*, 2018.

[12] Dubins, L. E., "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, Vol. 79, No. 3, 1957, pp. 497–516.

[13] Cook, B., Arnett, T., Rich, B., and Kivelevitch, E., "UAS Collision Avoidance, Navigation, and Target Assignment in a Congested Airspace Using Fuzzy Logic," *AIAA Infotech@ Aerospace*, 2015, p. 2031.

[14] Cook, B., Arnett, T., and Cohen, K., "A Fuzzy Logic Approach for Separation Assurance and Collision Avoidance for Unmanned Aerial Systems," *Modern Fuzzy Control Systems and Its Applications*, InTech, 2017, Chap. 12, pp. 225–256.

[15] Arnett, T., "Verification of Genetic Fuzzy Systems," Master's thesis, University of Cincinnati, 2016.

[16] Cybenko, G., "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, Vol. 2, No. 4, 1989, pp. 303–314.

[17] *Matlab and Global Optimization Toolbox R2016a*, The MathWorks Inc, 2016. Https://www.mathworks.com/help/gads/genetic-algorithm.html.

[18] Botev, Z. I., and L'Ecuyer, P., *Simulation from the Normal Distribution Truncated to an Interval in the Tail*, GERAD, École des hautes études commerciales, 2016.