SAND2021-15465C

# Addressing Undesirable Emergent Behavior in Deep Reinforcement Learning UAS Ground Target Tracking

Hannah Lehman*, Shelby Hackett,† and John Valasek‡
*Texas A&M University, College Station, TX 77843-3141*

**Deep reinforcement learning algorithms have been used to produce agent policies for unmanned air systems using non-gimballed cameras that are tracking ground targets. Simplifying abstractions to the environment are often used which mandate a relatively small number of states and actions and sometimes produce undesirable emergent behavior. This paper investigates the use of a learning-based algorithm in conjunction with a flight controller to eliminate undesirable emergent behavior for the non-gimballed camera, fixed-wing unmanned air system ground target tracking problem. Approaches investigated consist of fidelity of dynamical model, reward structure shaping, low-level controller, and changing action space and duration. These approaches mitigate undesirable emergent behavior and result in a learning method that is stable during training, resilient to hyperparameter values, and produces a flight controller that is able keep the target in the image frame of the camera. Results presented in the paper show that a proper combination of these techniques can greatly reduce the likelihood of the agent performing undesirable emergent behavior, while still providing acceptable target tracking performance with minimal ringing and smooth learning.**

## Nomenclature

| | | |
|---|---|---|
| $\mathcal{M}$ | = | Set that characterizes a Partially Observable Markov Decision Process |
| $\mathcal{S}$ | = | Set of possible states (state-space) |
| $\boldsymbol{s}$ | = | Vector of states |
| $\mathcal{A}$ | = | Set of possible actions (action-space) |
| $\boldsymbol{a}$ | = | Vector of actions |
| $\mathcal{T}$ | = | Set of conditional transition probabilities between states |
| $r$ | = | Reward function |
| $\tau$ | = | Trajectory (set of sequential states and actions taken) |
| $H$ | = | Heaviside (unit step) function |
| $h$ | = | Altitude |
| $w_{i,j}$ | = | Weight in reward function |

## I. Introduction

Unmanned Air Systems (UAS) serve as methods of tactical surveillance, reconnaissance, and intelligence gathering. Currently, a tactical UAS requires four personnel, each with their own responsibilities, to perform a mission. In addition to individual responsibility, the team must be in constant communication with one another to complete the mission. This makes even well-trained teams at risk for errors caused by miscommunication, slow reaction time, and missed or squandered opportunity. The use of machine learning to assume individual and/or group responsibility is an opportunity to improve mission reliability and safety.

There have been a number of approaches to solve the target tracking problem with a fixed strap-down camera on a small UAS. Beard and Egbert [1] derived explicit roll-angle and altitude-above-ground-level (AGL) constraints for

*Graduate Research Assistant, Vehicle Systems & Control Laboratory, Aerospace Engineering Department. Student Member AIAA. lehman@tamu.edu.

†Graduate Research Assistant, Vehicle Systems & Control Laboratory, Aerospace Engineering Department. Student Member AIAA. shackett@tamu.edu.

‡Professor and Director, Vehicle Systems & Control Laboratory, Aerospace Engineering Department. Fellow AIAA. valasek@tamu.edu, http://vscl.tamu.edu/valasek

road tracking. Their derivation guarantees that the target will remain in the camera frame provided it is possible to use edge tracking of the roadway, plus other features and landmarks. Beard and Saunders [2] later extended this work and addressed the problem of tracking a ground-based target using a fixed camera that was pointing out of the wing of a UAS. Rather than planning explicit trajectories for the vehicle, a nonlinear image-based feedback guidance strategy was developed that maintained the target in the FOV of the camera. Knowledge of predetermined terrain features such as roads and buildings enhanced the tracking performance and the algorithms cannot track in a non-structured featureless environment.

Other approaches have used machine learning to approach the problem of target tracking. Valasek et al. [3] created a reinforcement learning algorithm approach for learning bank angle control policies based on current bank angle and the target's position in the image frame. There are three advantages to an RL based approach. First, all learning, computation, and tracking takes place in the camera's image frame to simplify computations, and prior information about the geometry of the target space (nodes or road network, landmarks, features, coordinates) are not required to do the tracking. This permits tracking of targets in completely featureless terrain environments. Second, knowledge of the target dynamics in the image frame is not needed. Third, determining an accurate dynamical model relating the target position and motion in the image frame to aircraft states and controls can be difficult, and the model will have uncertainties resulting from the modeling process. The modeled kinematic chain from target position in the inertial frame to the aircraft frame introduces computational errors that propagate with each successive rotation. By using an RL approach, the control policy can be determined without needing to first identify the entire system dynamics. A feature of this approach is that the learning agent will continue to learn, refine, and update the control policy previously learned offline during actual operation. Use of this approach does not require system models, although they can help offline learning, and the reinforcement learning eliminates the need for any heuristic or ad hoc approaches.

Noren et al. [4] detailed the initial flight testing of a previously trained control policy [3] for the autonomous tracking of ground targets. Noren's work showed that it is possible to train a reinforcement learning controller in a simulated environment and deploy the learned policy to hardware. Despite limitations imposed by planar motion and single-output controller constraints, satisfactory tracking performance was achieved in hardware. This ability to transfer from simulation to hardware has also been shown in the work presented in [5] in which a policy was initially trained offline in simulation and improved online on the vehicle during flight testing. In addition, this work showed that the agent learned behavior that a pilot would not have thought to perform, improving the tracking ability.

A phenomena called emergent behavior occurs when a trained agent learns to exploit the reward to produce novel, unexpected behavior. While this may be useful in some cases, the emergent behavior encountered in solving the ground target tracking problem is shown to be undesirable. This paper investigates various approaches to counteract the emergent behavior and produce a machine learning agent that will improve mission capability, safety, and reliability.

## II. Learning and Simulation Environment

The presented utilization of soft actor-critic algorithms to a learning-based flight controller frames the ground target tracking as a reinforcement learning problem. In Reinforcement Learning (RL), it is desired to train an agent to learn the parameters $\theta$ of a policy (or controller) $\pi_\theta$, in order to map the partially-observable environment's observation vector $\boldsymbol{o}$ (or state vector $\boldsymbol{s}$ in fully-observable environments) to agent actions $\boldsymbol{a}$. The performance of the agent is measured by a scalar reward signal $r$ returned by the environment. Fig. 1 illustrates this description as a diagram and Fig. 2 shows how it can be applied to the target tracking case.

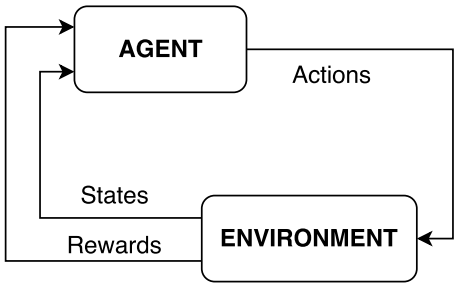


**Fig. 1 Reinforcement learning problem modeled in terms agent and environment.**

The goal of Reinforcement Learning (RL) is to learn a policy in a potentially non-deterministic environment. When

the environment is unknown, this becomes model-free RL, which can be formalized as a Partially Observable Markov Decision Process (POMDP) [6]. A POMDP $\mathcal{M}$ can be characterize by its state-space $\mathcal{S}$ (vector of states $\boldsymbol{s} \in \mathcal{S}$), an action-space $\mathcal{A}$ (vector of actions $\boldsymbol{a} \in \mathcal{A}$), a transition operator $\mathcal{T}$ (which defines the probability distribution $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t)$), and the reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ (or $r(\boldsymbol{s}, \boldsymbol{a})$. This definition is illustrated by Eq. 1:

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}. \tag{1}$$

To simplify derivation of the algorithm it is assumed to be a fully-observable environment. This is shown in Fig. 1. In Reinforcement Learning, at each time step $t$ of a time horizon $T$, a policy (or controller) $\pi$, parametrized by $\theta_t$, maps the current environment's states $\boldsymbol{s}_t$ to actions $\boldsymbol{a}_t$: $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)$. This action affects the current environment's states $\boldsymbol{s}_t$ which evolve to $\boldsymbol{s}_{t+1}$ based on the environment's transition distribution (dynamics) $\mathcal{T} = p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. The environment also returns a scalar reward function $r$ that evaluates the action taken $\boldsymbol{a}_t$ at the state $\boldsymbol{s}_t$: $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$.

During an episode, the sequence of states observed and actions taken over a number of time steps $T$ can be represented by a trajectory $\tau$:

$$\tau = \{\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_T, \boldsymbol{a}_T\} \tag{2}$$

The probability of experiencing a given trajectory $\tau$ in a Markov Decision Process can be written as:

$$\pi_\theta(\tau) = p_\theta(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_T, \boldsymbol{a}_T) \tag{3}$$

$$= p(\boldsymbol{s}_0) \prod_{t=1}^{T} \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) \tag{4}$$

The goal in policy gradient RL methods is to find the parameters $\theta^*$ of policy $\pi$ parametrized by $\theta$ that will maximize the objective $J(\theta)$, which represents the expected total reward to be received by this policy across all timesteps of the episode. This present work is primarily concerned with the Soft Actor-Critic algorithm, though it will compare results to other learning algorithms such as Deep Deterministic Policy Gradient (DDPG).

### A. Framing the Tracking Problem as Reinforcement Learning

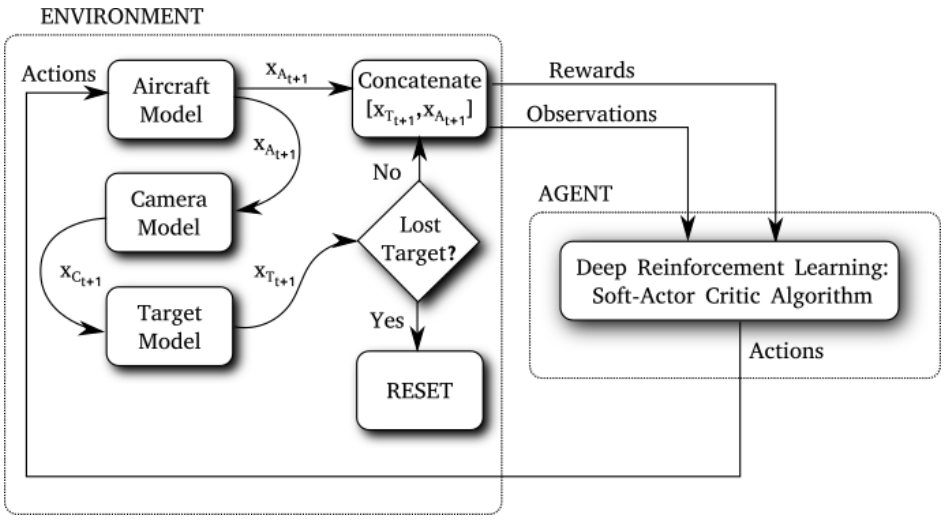


**Fig. 2 Intelligent motion video guidance for unmanned air system ground target tracking modeled as a reinforcement learning problem.**

In the target tracking case presented by this work, the environment comprises the aircraft, camera, and ground target model, shown in Figure 3 and explained in more details in Section II.B. Environment states are represented by the linearized aircraft states and target pixel positions $X_T$ and $Y_T$ in the image plane of the camera (Eq. 5). The aircraft dynamical states are $u$ , velocity, $\alpha$, angle-of-attack, $q$, body-axis pitch rate, $\theta$, pitch attitude angle, $\beta$, sideslip angle, $p$, body-axis roll rate, $r$, body-axis yaw rate, $\phi$, bank angle, $\psi$, heading angle, and $h$, altitude.

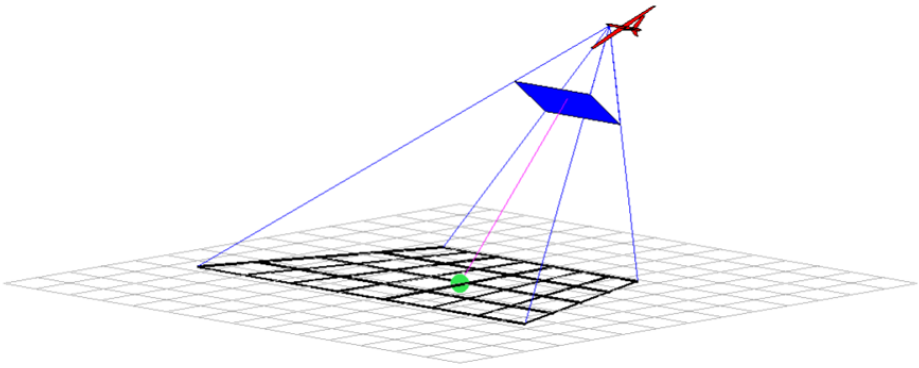

**Fig. 3 Geometry of an unmanned air system with a fixed-camera tracking a ground target in the camera image frame.**

$$\mathcal{S} = \{X_T, Y_T, u, \alpha, q, \theta, \beta, p, r, \phi, \psi, h\} \tag{5}$$

The agent controls the aircraft through elevator, throttle, aileron, and rudder actions, $\delta_e$, $\delta_T$, $\delta_a$, $\delta_r$ respectively (Eq. 6). The agent's actions are evaluated by a scalar value, the reward signal. The reward signal is computed based on the normalized Euclidean distance of the target to the center of the image plane (Eq. 7). Initial research was performed using the AV-8. This aircraft was deemed unsuitible for the desired task, and the modeling was first switched to the MQ-9 and later to the RMRC Anaconda to match the future flight testing platform.

$$\mathcal{A} = \{\delta_e, \delta_T, \delta_a, \delta_r\} \tag{6}$$

$$r = \left(1 - \frac{\sqrt{X_T^2 + Y_T^2}}{\sqrt{X_{T\,MAX}^2 + Y_{T\,MAX}^2}}\right)^2 \tag{7}$$

The learning-based flight controller optimizes the action selection by an iterative process in a simulated environment. The simulation runs many episodes, each starting with a ground target and the aircraft. The aircraft initializes in a random position in the inertial space and the ground target initializes in the image frame. The episode resets if the target leaves the image frame or if the aircraft performs any maneuver outside the safety limits as defined by the authors or the environment itself. In most cases these limits are determined by the linear range of the aircraft or limitations of the direction cosine matrices. In some cases additional constraints are added for g-loading or other flight restrictions.

### B. Aircraft Agent Modeling

The learning controller that handles the full-state continuous target tracking case is trained using a full-state, continuous, non-real time simulation that incorporates models of the camera, target, and UAS agent. A six degree-of-freedom, Linear Time Invariant (LTI) state-space model is used for the aircraft model in the environment. The agent learns using the states defined in Eq. 5, and originally using actions consisting of low-level commands elevator $\delta_e$, aileron $\delta_a$, rudder $\delta_r$, and throttle $\delta_t$ (Eq. 6). The first model was a non-coupled LTI state-space model of the Cessna 172 which was built using stability & control derivatives available in the open literature [7]. This produced mediocre results since the desired full coupling between longitudinal and lateral/directional dynamics which the agent needed to experience during training was not present. This is because the coupling derivatives are typically not available for aircraft models in the open literature.

The model limitation was overcome using a higher fidelity model that is fully coupled between longitudinal and lateral/directional dynamics. The RMRC Anaconda (Figure 4) was selected since it is also the vehicle which has been used for validation & verification flight testing of the agents in the author's previous work, and it will be used in a similar role for the present work. The RMRC Anaconda model was synthesized using data collected from flight testing and the algorithms and approach for near real-time online system identification developed by the authors in [8], [9]. The model of the RMRC Anaconda synthesized and used is presented in the Appendix.


**Fig. 4 RMRC Anaconda platform external physical characteristics**

Camera specifications are shown in Table 1. The target is modeled as a point mass with planar motion on the XY plane. Policies are trained here for the case of stationary targets.

**Table 1 Camera model specifications**

| Parameter | Value |
|---|---|
| Resolution (pixels) | 1024x768 |
| Aspect Ratio | 4:3 |
| Horizontal Field of View (deg) | 90 |
| Vertical Field of View (deg) | 30 |
| Pan Angle w.r.t. Aircraft Frame (deg) | -90 |
| Tilt Angle w.r.t. Aircraft Frame (deg) | -20 |

## III. Emergent Behavior and Iterative Environment Changes

The goal of much reinforcement learning is to learn and implement solutions that humans may be unable to perform quickly or safely. These solutions may at times be beyond what a human operator would conceive or be able to execute [10]. An example of good emergent behavior is demonstrated in [5] by the agent's unorthodox reaction to unexpected target behavior during a test flight. By incrementally increasing separation distance from the target, rather than circling in the opposite direction and attempting to re-acquire it later, tracking was maintained without interruption in a manner that was unexpected by the human operator.

More often however, the agent learns to exploit the reward structure to the detriment of the mission goal [11]. Repeated instances occurred early on in the present work when the agent learned to exploit the reward structure by diving at the target, despite stable reward and episode length training results. This behavior is not indicated in the standard evaluation metrics, thus the environment must be modified such that evaluation reflects the desirability of the behavior of the agent. Diving is the emergent behavior this work investigates to avoid, and success will depend upon the ability to recognize diving behavior in standardized evaluation metrics, eliminate diving from agent time histories, and exhibit lengthy, robust tracking of ground targets.

### A. Initial Environment Changes and Reward Selection

Since this behavior was not present in [5] the primary cause of diving was thought to be the expanded action space and potentially the aircraft model being used. The aircraft model was changed to a vehicle at a non-hover flight condition as detailed in the previous section. The Cessna 172 model was implemented in the present work to give the agent the best possible chance to perform smooth actions. The controller agent using the new model showed improvement but still spiraled down to the target. The reward structure was then modified to add a penalty for extreme altitude drops as shown

in Eq. 8 where $r_i$ is the reward for the position in the image frame as defined before, h is the altitude, w are weights, and H is the Heaviside function. It is undesirable to completely restrict altitude and speed as in [5]. Therefore, the altitude reward was built with a slack-band around the ideal altitude. Changes in altitude within the margin to aid maneuvering are allowed, but any movement outside of the acceptable range will be penalized on a linear scale as shown in Fig. 5.

$$r = r_i - (w_{1,1}h + w_{1,2}) + (w_{1,1}h + w_{1,2})H(h - 450) \tag{8}$$

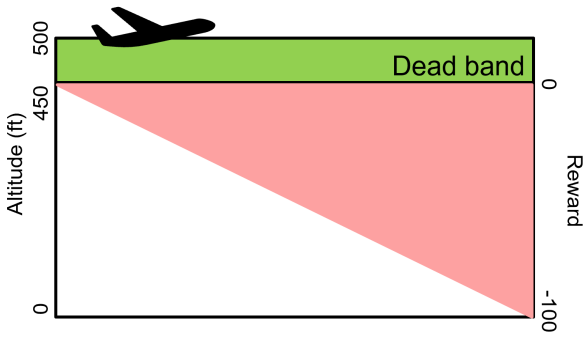


**Fig. 5 Slack-band altitude reward structure.**

Initial experiments were performed with other reward functions (eg. piecewise continuous) but the linear reward function produced the most consistent and simple agent reactions. This modified reward structure vastly reduced the incidences of diving and spiraling, but resulted in shorter than ideal episode lengths. The new environment with added altitude reward was initially trained on the original DDPG algorithm as in [12]. Despite no change to the image frame reward function, the addition of the altitude reward produced a higher average reward return. This is presumably due to better average tracking in the image frame while the episode is running. In addition, there is high variance in reward return. It is clear that exploration for a more stable algorithm would be beneficial.

Alternative algorithms were explored and comparisons were made to a number of algorithms from *Stable Baselines* [13]. The algorithms chosen were Soft Actor Critic, TD3, TRPO, and DDPG. Comparisons between the basic structures and capabilities of these algorithms are shown in [13]. In these comparison plots, each agent was trained separately from three random seeds and the results averaged across the three sets. The mean of the sets was plotted along with a shaded region of one standard deviation above and below the mean. This allows for ready comparison of both the average return as well as the volatility and stability of each agent.

Once trained, the resulting plots for reward moving average and mean episode length can be seen in Fig. 6. Despite similarities in reward and episode results, the agents chose vastly different methods by which to obtain those rewards. Some agents, such as TD3, chose to immediately cut thrust, while others such as TRPO immediately set thrust to 100%. DDPG agents universally had fairly prominent ringing (rapid oscillation of controls), while SAC in particular had the least ringing. These results of the SAC training show an ability to train a stable policy that reacts to changing states in order to maintain view of the target. SAC has the fastest initial training gains and results in the most stable end agent. For this reason, SAC was chosen as the algorithm of choice for further investigation of reward structure modifications and their effect on episode length.

Despite significant differences in average reward the episode duration remains around 400 timesteps for all algorithms. The discrepancy between increasing reward and constant timeteps shows that the reward is poorly optimized for the actual results desired, i.e., a long tracking duration. Many episodes showed a tendency to dive towards the target but stopped before hitting the target or losing the aircraft. The altitude penalty was effective in preventing loss of the vehicle but failed to incentivize the agent to properly execute an orbit of the target to keep it in the image frame. Keeping the target in the center of the image frame is desirable but not the major objective of the problem. Therefore a number of changes to the reward structure were made to rectify this disparity between reward and goal. Many reward structures were tested based on various simplifications and penalties, and the results of these changes at the low-level commands are shown in Fig. 7. These changes resulted in a about a 100% increase in average episode length.

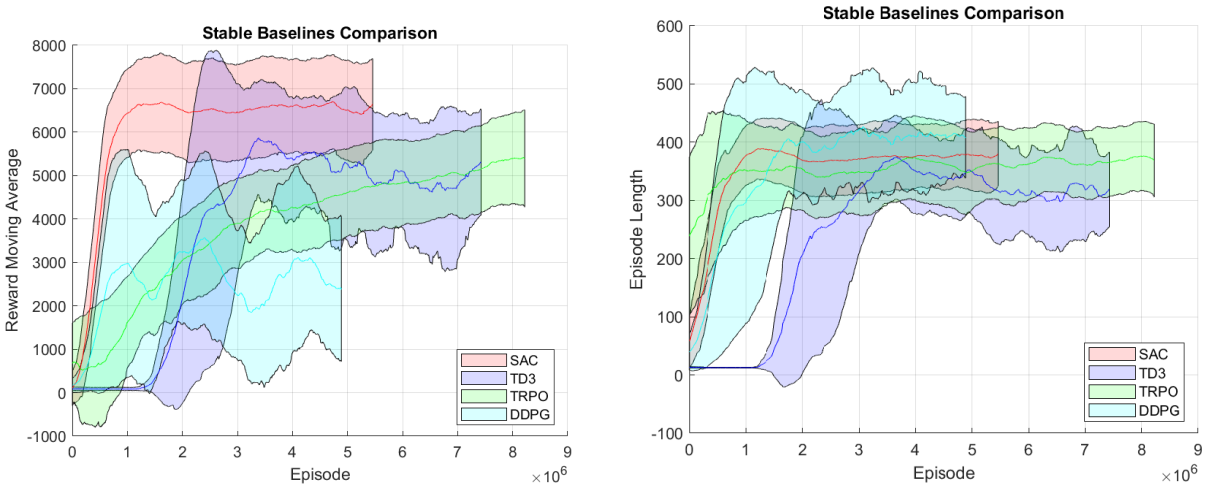


**Fig. 6 Comparisons between reward and episode length returns for various Stable Baseline models.**

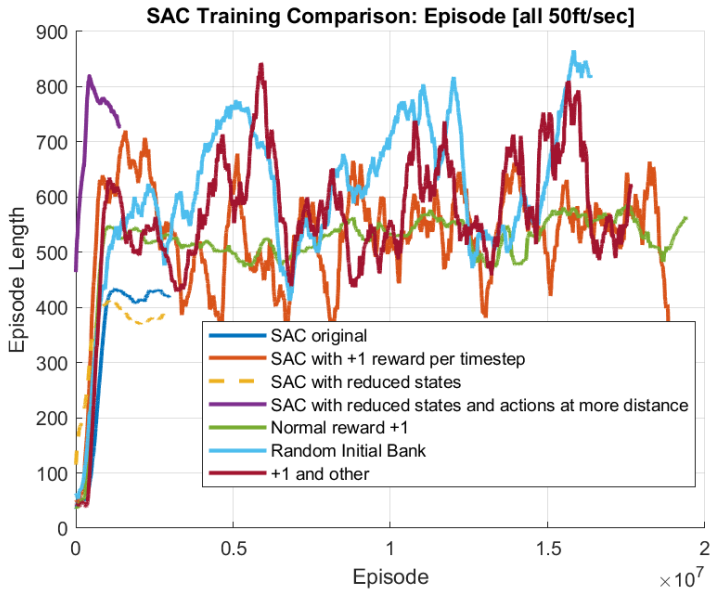


**Fig. 7 Comparison between several low level command agents.**

### B. Controller Implementation

However, the agent still failed to learn to orbit the target. One approach to attempt to gain this ability is to learn on environment states (or high-level actions) rather than actions. In brief, though many modern Neural Net approaches find more consistent results using low level inputs, the sample inefficiency of RL may require high level inputs [14]. Results of a similar episode length comparison after the controller was implemented are shown in Fig. 8.

An optimal Nonzero Setpoint (NZSP) flight controller ([15]) was synthesized and incorporated in the RMRC Anaconda model shown in Equations 22 and 24 to help mitigate diving behavior. The approach from [15] is summarized here with Equations 10 through 20. Using the optimal NZSP results in a new action space of commanded states as shown in Equation 9 with the optimal NZSP generating the lower-level commands to provide not only the desired tracking but also stability augmentation of the plant.

$$\mathcal{A} = \{u, q, \beta, \Phi\} \tag{9}$$

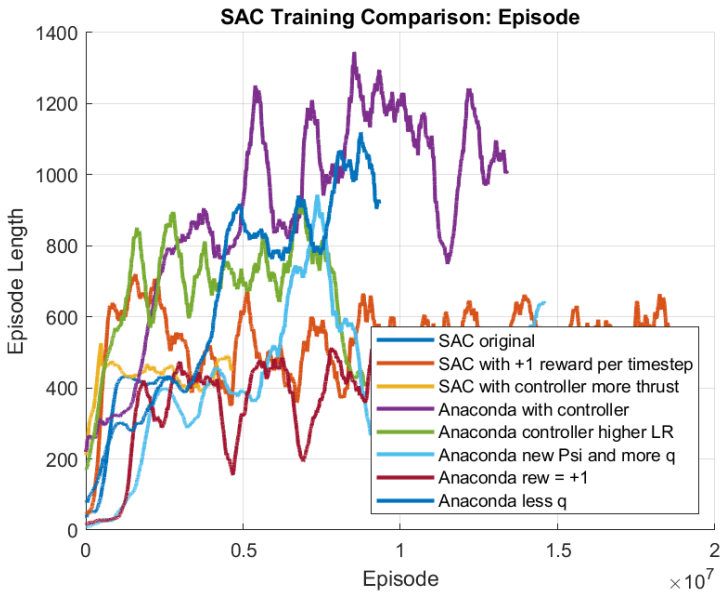


**Fig. 8 Comparison between original low level agents ('SAC') and select high level agents.**

The optimal Nonzero Setpoint (NZSP) is a command augmentation structue (CAS) which steers the plant to a terminal steady-state condition, with guaranteed state-tracking properties. For a linear time invariant system with $n$ states and $m$ controls,

$$\begin{aligned} \mathbf{x} &= A\mathbf{x} + B\mathbf{u}; \qquad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u} \\ \mathbf{x} &\in \mathbb{R}^n, \quad \mathbf{u} \in \mathbb{R}^m, \quad \mathbf{y} \in \mathbb{R}^m \end{aligned} \tag{10}$$

it is desired to command some of the initial outputs $\mathbf{y}$ to steady-state terminal output values $\mathbf{y}_m$ and keep them there as $t \to \infty$. If these terminal outputs are trim states, denoted by $^*$, then at the terminal steady-state condition the system is characterized by

$$\begin{aligned} \mathbf{x}^* &= A\mathbf{x}^* + B\mathbf{u}^* \equiv 0 \\ \mathbf{y}_m &= H\mathbf{x}^* + D\mathbf{u}^* \\ \mathbf{x}^* &\in \mathbb{R}^n, \quad \mathbf{u}^* \in \mathbb{R}^m, \quad \mathbf{y}_m \in \mathbb{R}^m \end{aligned} \tag{11}$$

For guaranteed tracking, the number of commanded outputs $\mathbf{y}_m$ must be less than or equal to the number of controls $m$. Error states and error controls are defined as

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{x} - \mathbf{x}^* \\ \tilde{\mathbf{u}} &= \mathbf{u} - \mathbf{u}^* \end{aligned} \tag{12}$$

where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ are the error between the current state and control respectively, and the desired state and control respectively. The state equations can be written in terms of these error states as

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{x} - \mathbf{x}^* = A\mathbf{x} + B\mathbf{u} - (A\mathbf{x}^* + B\mathbf{u}^*) \\ \tilde{\mathbf{x}} &= A\tilde{\mathbf{x}} + B\tilde{\mathbf{u}} \end{aligned} \tag{13}$$

with quadratic cost function to be minimized

$$J = \frac{1}{2}\int_0^\infty \left[\tilde{\mathbf{x}}^T Q\tilde{\mathbf{x}} + \tilde{\mathbf{u}}^T R\tilde{\mathbf{u}}\right] dt \tag{14}$$

The optimal control which minimizes Eqn.14 is obtained by solving the matrix algebraic Riccati equation for infinite horizon

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{15}$$

resulting in

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{16}$$

A feedback control law in terms of the measured states is obtained by converting $\tilde{\mathbf{u}}$ back to $\mathbf{u}$, giving

$$\mathbf{u} = (\mathbf{u}^* + K\mathbf{x}^*) - K\mathbf{x} \tag{17}$$

with $\mathbf{u}^*$ and $\mathbf{x}^*$ constants. They are solved for directly by inverting a quad partition matrix deduced from Eqn.11

$$\begin{bmatrix} A & B \\ H & D \end{bmatrix}^{-1} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$
$$\begin{bmatrix} \mathbf{x}^* \\ \mathbf{u}^* \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{y}_m \end{bmatrix} \tag{18}$$

and then solving for

$$\begin{aligned} \mathbf{x}^* &= X_{12}\mathbf{y}_m \\ \mathbf{u}^* &= X_{22}\mathbf{y}_m \end{aligned} \tag{19}$$

Upon substitution in Eqn.17 the control law implementation equation becomes

$$\mathbf{u} = (X_{22} + KX_{12})\mathbf{y}_m - K\mathbf{x} \tag{20}$$

For the optimal control policy $\mathbf{u}$ to be admissible, the quad partition matrix must be invertible. Therefore, the equations for $\mathbf{x}^*$ and $\mathbf{u}^*$ must be linearly independent, and the number of outputs or states that can be driven to a constant value must be less than or equal to the number of available controls. An advantage of this controller is the guarantee of perfect tracking of a number of outputs equal to the number of controls, independent of the value of the gains, provided they are stabilizing. The gains can be designed using any desired technique, and only affect the transient performance, and not the guarantee of steady-state performance.

In Figure 8 it is shown that the introduction of a controller increases the episode length from about 400 timesteps to around 1350, a 220% increase. An example of one episode using the controller and additional penalties is shown in Fig. 9.

### C. Action Space Reduction and Extended Action Duration

Another method of increasing agent performance is reducing environment complexity. It is important for the fidelity of the simulation to maintain all aircraft dynamic complexity, but it is possible through the use of a controller, to reduce the number of actions that the agent must control. An example of an agent trained with reduced action space is shown in Fig. 10. This agent only learns one command, bank angle, and allows the rest of the actions to be handled by the controller. It is apparent from this episode that the emergent diving behavior is no longer exhibited and some turning behavior has been learned, although it does not complete a full turn.

A significant delay between action taken and reward payoff is another hurdle. If the agent needs to turn and it takes several seconds to get into a turn, the agent must purposefully choose to continue to turn for hundreds of timesteps before it receives the reward. An increased action duration (or a decrease in action duration) would allow for an action and its resulting reward to be more immediately linked. The combination of action duration increase and action space reduction proved to have a significant effect on agent behavior and episode duration. An example of an agent trained using both reduced action space and extended action duration is shown in Fig. 11 in which the action duration has been changed from 100Hz to 2Hz. This command rate was chosen based on necessary update rates from [4] and performs well when compared to 10, 5, and .5 Hz models. This agent is able to handle different vehicle movement cases as shown in Fig. 12 which tracks a randomly moving target. A comparison between these approaches and the approaches discussed previously is shown in Fig. 13. This simplification and sampling approach results in a 49 fold increase in average episode length.

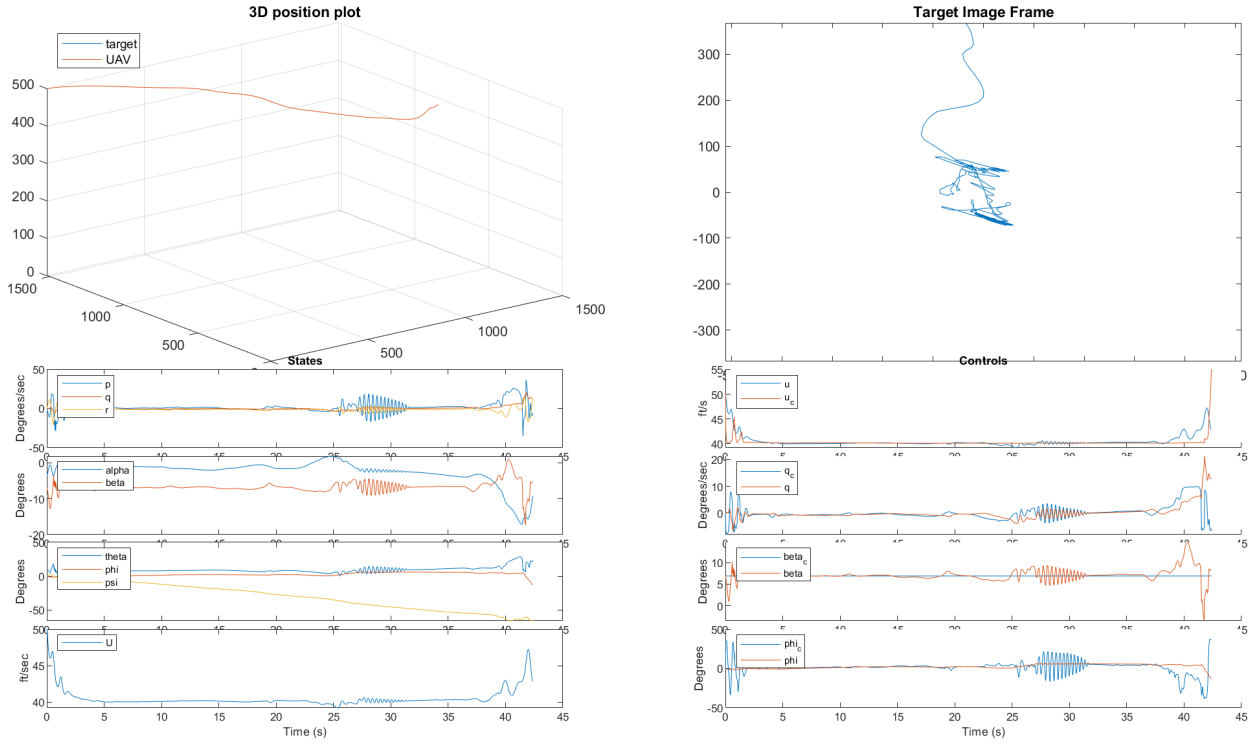


**Fig. 9 Example of a high-level command agent**

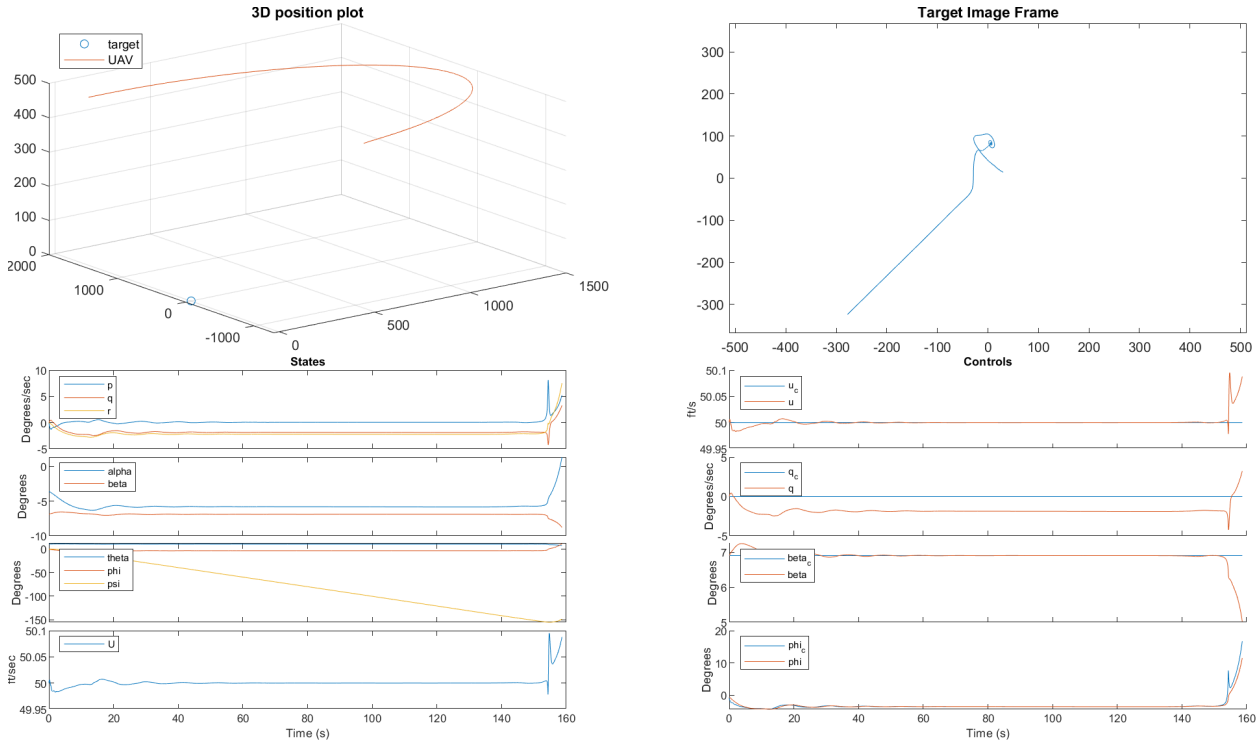


**Fig. 10 Example run using reduced action space**

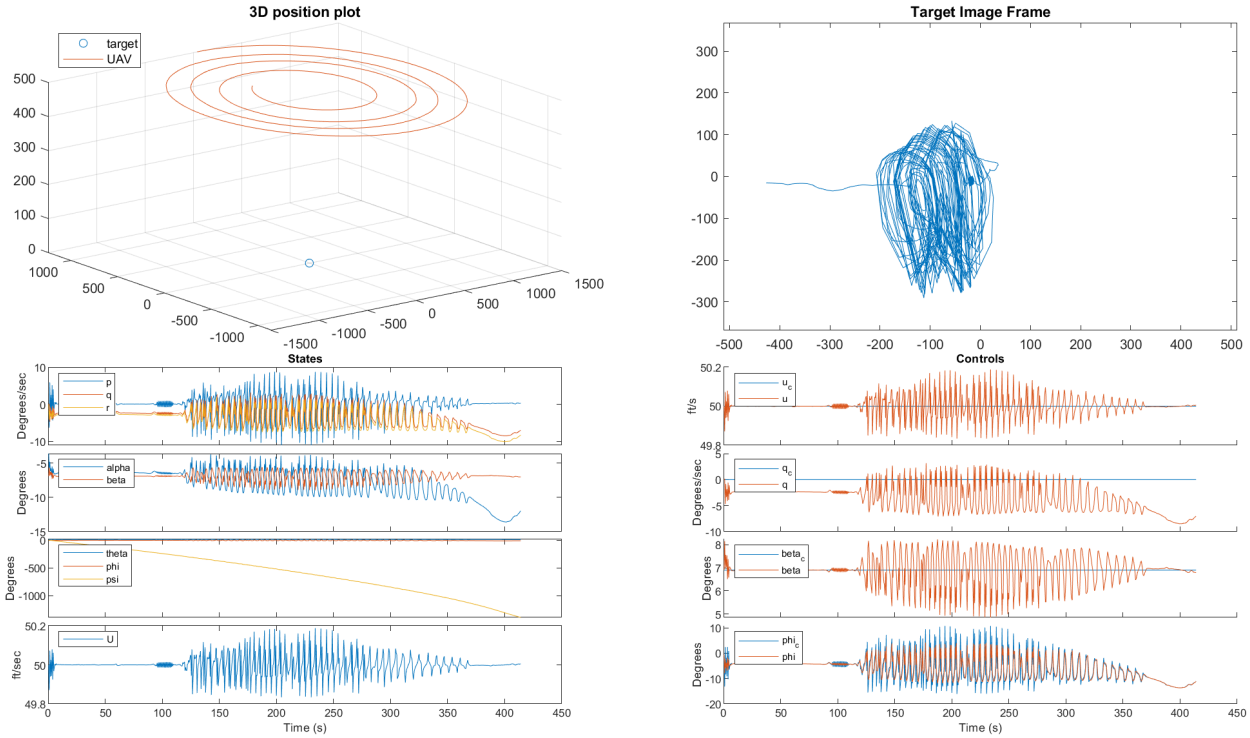


**Fig. 11 Example run using reduced action space and 2Hz action duration**

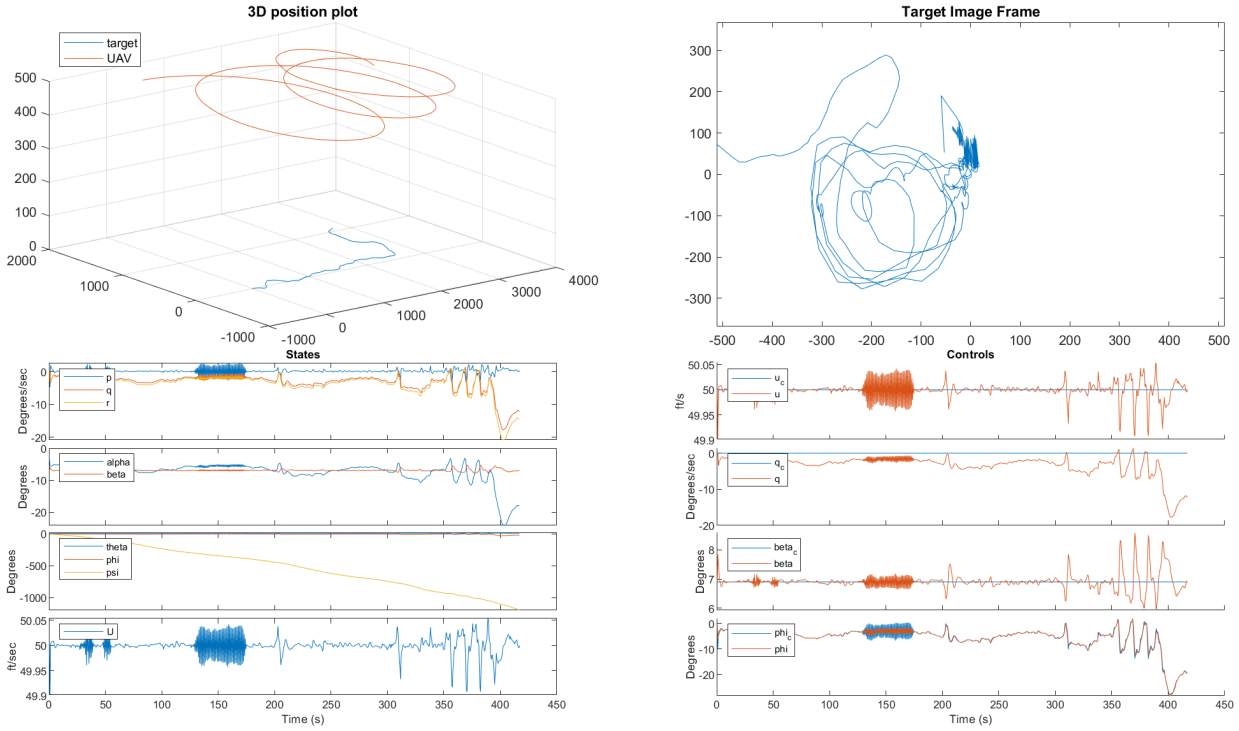


**Fig. 12 Example run using reduced action space and 2Hz action duration tracking a randomly moving target**

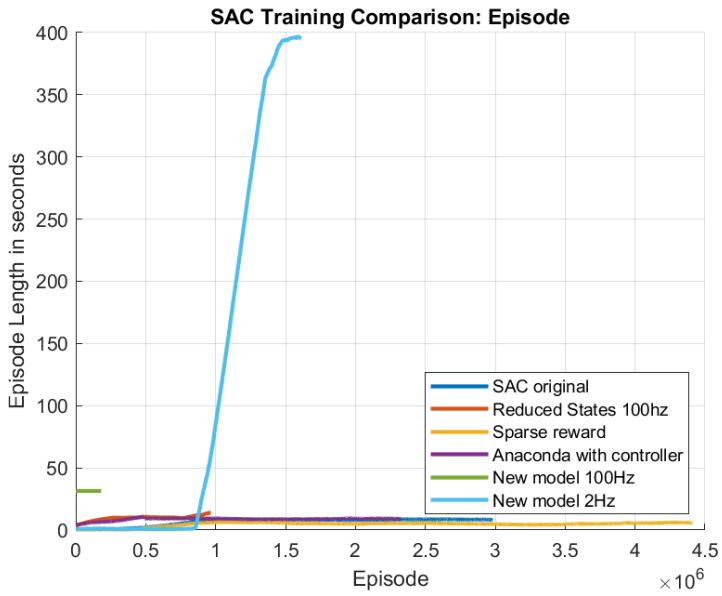


**Fig. 13 Episode length comparison of models in seconds**

## IV. Conclusions

This paper investigated an approach to mitigate undesirable emergent behavior by utilizing a learning method that is stable during training, resilient to hyperparameter values, and produces a flight controller that is able keep the target in the image frame of the camera. Based upon results presented in the paper, the following conclusions are drawn:

1) For the problem investigated, use of four actions was shown to produce undesirable emergent behavior in terms of inadmissible aircraft responses and trajectories. These behaviors utilize the action space in ways that lead to state values that must be discouraged through the use of penalties and modifications to the learning environment.
2) Average episode length was increased by a factor of approximately 49 and impacting the ground was completely eliminated by an appropriate combination of a more appropriate vehicle model, learning high-level commands to a flight controller, reduced action space and action sampling, and modifying the reward structure to better reflect the desired behavior.
3) Soft Actor Critic performed better than Deep Deterministic Policy Gradient when new anti-diving metrics were introduced. Deep Deterministic Policy Gradient was shown to have difficulty learning complex maneuver behaviors when using low-level commands. Soft Actor Critic was able to achieve approximately a factor of 49 longer episodes with minimal ringing due to use of the extended action duration, and exhibited reduced learning volatility.

Future work involves the integration of relative actions and newer linear models and reward structures. Additionally, the integration of hyperparameter optimization techniques such as Optuna will be investigated. Results of this work will be validated with flight testing of an unmanned air system.

## V. Acknowledgement

This research was funded by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. The technical monitors are Daniel Whitten and James E. Pagan. This support is gratefully acknowledged by the authors. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

## References


[1] Egbert, J., and Beard, R. W., "Low Altitude Road Following Constraints Using Strap-down EO Cameras on Miniature Air Vehicles," *2007 American Control Conference*, 2007, pp. 353–358. doi:10.1109/ACC.2007.4282767.

[2] Saunders, J., and Beard, R. W., "Visual Tracking in Wind with Field of View Constraints," *International Journal of Micro Air Vehicles*, Vol. 3, No. 3, 2011, pp. 169–182. Doi: 10.1260/1756-8293.3.3.169.

[3] Valasek, J., Kirkpatrick, K., May, J., and Harris, J., "Intelligent Motion Video Guidance for Unmanned Air System Ground Target Surveillance," *Journal of Aerospace Information Systems*, Vol. 13, No. 1, 2016, pp. 10–26. Doi: 10.2514/1.I010198.

[4] Noren, C., Valasek, J., G. Goecks, V., Rogers, C., and Bowden, E., "Flight Testing of Intelligent Motion Video Guidance for Unmanned Air System Ground Target Surveillance," *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018. doi:10.2514/6.2018-1632.

[5] Valasek, J., Lehman, H., and Goecks, V. G., "Online Intelligent Motion Video Guidance for Unmanned Air System Ground Target Surveillance," *AIAA Scitech 2019 Forum*, 2019. doi:10.2514/6.2019-0135.

[6] Bertsekas, D. P., *Dynamic Programming and Optimal Control*, 2nd ed., Athena Scientific, 2000.

[7] Roskam, J., *Airplane Flight Dynamics and Automatic Flight Control Part I*, Design, Analysis, and Research Corporation, Lawrence, KS, 1994.

[8] Lu, H. H., Rogers, C., Goecks, V. G., and Valasek, J., "Online Near Real Time System Identification on a Fixed-Wing Small Unmanned Air Vehicle," *2018 AIAA Atmospheric Flight Mechanics Conference*, 2018. doi:10.2514/6.2018-0295, URL `https://arc.aiaa.org/doi/abs/10.2514/6.2018-0295`.

[9] Leshikar, C., and Valasek, J., "System Identification Flight Testing of Inverted V-Tail Small Unmanned Air System," *2022 AIAA Atmospheric Flight Mechanics Conference*, 2022.

[10] "In Two Moves, AlphaGo and Lee Sedol Redefined the Future," `https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/`, 2016. Accessed: 2021-06-04.

[11] "Faulty Reward Functions in the Wild," `https://openai.com/blog/faulty-reward-functions/`, 2016. Accessed: 2021-06-04.

[12] Goecks, V. G., and Valasek, J., "Deep Reinforcement Learning on Intelligent Motion Video Guidance for Unmanned Air System Ground Target Tracking," *AIAA Scitech 2019 Forum*, 2019. doi:10.2514/6.2019-0137.

[13] Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y., "Stable Baselines," `https://github.com/hill-a/stable-baselines`, 2018.

[14] Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., and Levine, S., "Learning to walk via deep reinforcement learning," *arXiv preprint arXiv:1812.11103*, 2018.

[15] Valasek, J., Gunnam, K., Kimmett, J., Tandale, M. D., Junkins, J. L., and Hughes, D., "Vision-Based Sensor and Navigation System for Autonomous Air Refueling," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 5, 2005, pp. 979–989. doi:10.2514/1.11934, URL `https://doi.org/10.2514/1.11934`.


## Appendix: Aircraft Linear Model

### A. RMRC Anaconda

The six degree-of-freedom fully-coupled linear time invariant state-space model of the RMRC Anaconda is obtained by system identification flight testing about a steady, level, 1g trimmed flight condition. All angular quantities are in radians. The trim states and values are shown in Table 2. The state vector is

$$x = \begin{bmatrix} \delta u & \delta\alpha & \delta q & \delta\theta & \delta\beta & \delta p & \delta r & \delta\Phi \end{bmatrix} \tag{21}$$

The state matrix is

$$A = \begin{vmatrix} -0.1101 & -0.9545 & -2.6177 & 30.8596 & 2.8778 & -0.4403 & 1.7681 & 3.1469 \\ -0.0006 & -0.6756 & -0.4518 & 0.174 & -0.3125 & 0.0265 & 0.1319 & 0.394 \\ 0.0426 & 10.787 & -0.2738 & -5.1785 & 3.8784 & -0.1857 & -0.1066 & 1.4919 \\ 0.0004 & 0.0631 & 0.9637 & -0.0339 & 0.0259 & 0.0061 & -0.0153 & 0.0199 \\ 0.0004 & 0.0429 & -0.0068 & 0.2514 & -0.7343 & -0.0777 & -1.0891 & 0.6173 \\ 0.0059 & 9.3587 & 3.7037 & 0.694 & -5.4006 & -3.2869 & 4.4875 & -10.7015 \\ 0.0095 & 0.1539 & 0.4569 & -5.0034 & 17.1677 & -0.9996 & -0.5098 & 2.006 \\ 0.0003 & 0.1739 & 0.0859 & -0.021 & -0.1022 & 0.8977 & -0.0797 & -0.0796 \end{vmatrix} \tag{22}$$

The control vector is

$$u = \begin{bmatrix} \delta_e & \delta_T & \delta_a & \delta_r \end{bmatrix} \tag{23}$$

The control influence matrix is

$$B = \begin{vmatrix} -6.9241 & 15.7696 & -0.3614 & -6.8906 \\ 0.9769 & -0.0071 & 0.0786 & -0.6223 \\ -23.4937 & -0.1652 & 0.4653 & 0.4633 \\ -1.7197 & -0.0111 & 0.0336 & 0.1646 \\ 0.2327 & 0.0483 & 0.071 & 1.1513 \\ 9.0601 & -0.6782 & 23.6621 & -12.1168 \\ -3.4857 & -0.2528 & -2.5434 & -11.8281 \\ 1.3545 & -0.0809 & 2.294 & -1.052 \end{vmatrix} \tag{24}$$

**Table 2 RMRC Anaconda trim states and controls**

| Trim Parameter | Value |
|---|---|
| $U_1$ (ft/s) | 69.3 |
| $H_1$ (ft) | 500 |
| $\alpha_1$ (deg) | -3.6 |
| $\theta_1$ (deg) | 10.4 |
| $\beta_1$ (deg) | 6.9 |
| $\Phi_1$ (deg) | -0.6 |
| $\delta_T$ (%) | 60 |
| $\delta_e$ (deg) | 0 |
| $\delta_a$ (deg) | 0 |
| $\delta_r$ (deg) | 0.22 |