Effectiveness of Warm-Start PPO for Guidance with Highly Constrained Nonlinear Fixed-Wing Dynamics

Christian T. Coletti, Kyle A. Williams, Hannah C. Lehman, Zahi M. Kakish, Daniel Whitten, Julie J. Parish
Sandia National Laboratories
Albuquerque, New Mexico
ctcolet@sandia.gov, kwilli3@sandia.gov, hclehma@sandia.gov, zmkakis@sandia.gov, wdwhitt@sandia.gov, jparish@sandia.gov

***Abstract*— Reinforcement learning (RL) may enable fixed-wing unmanned aerial vehicle (UAV) guidance to achieve more agile and complex objectives than typical methods. However, RL has yet struggled to achieve even minimal success on this problem; fixed-wing flight with RL-based guidance has only been demonstrated in literature with reduced state and/or action spaces. In order to achieve full 6-DOF RL-based guidance, this study begins training with imitation learning from classical guidance, a method known as warm-staring (WS), before further training using Proximal Policy Optimization (PPO). We show that warm starting is critical to successful RL performance on this problem. PPO alone achieved a 2% success rate in our experiments. Warm-starting alone achieved 32% success. Warm-starting plus PPO achieved 57% success over all policies, with 40% of policies achieving 94% success.**

***Index Terms*—Autonomous Systems, Reinforcement Learning, Unmanned Aerial Vehicles (UAVs), Artificial Neural Networks, Intelligent Control, Autonomous Vehicles**

I. Introduction

Guidance of unmanned aerial vehicles (UAVs) is an active research area that seeks to extend the flight capability of both fixed-wing and rotorcraft autopilots. With the growth and popularization of UAVs for research and commercial purposes, many new applications are arising that necessitate autonomous guidance of more complex objectives or more agile maneuvers. While rotorcraft initially seem to be good candidates for these applications due to the ability to hover and move in any direction, fixed-wing aircraft of comparable mass have increased range, payload capacity, and for some designs, maneuverability.

Common commercial autopilots for fixed-wing aircraft utilize linear techniques and therefore excel at low-agility guidance near trim flight regimes [1]. However, these guidance and control models cannot handle the advanced requirements of new applications. The linear, time-invariant models inexactly describe the nonlinear and unsteady dynamics of real aircraft, resulting in controllers with only local stabilization [2]. During high-agility flight, such as high roll angle maneuvers, the linearization error increases. Modern autopilots may be able to handle these regimes and some complex objectives, such as obstacle avoidance or mission optimization, but they require human-in-the-loop derivations for each application and have high computational cost at runtime compared to neural inference [3]. Mitigating the downsides of advanced guidance and control would greatly improve capabilities of UAVs and should enable new high-complexity applications.

Fig. 1. The Zagi Flying Wing UAV modeled in this study [4]

The complexities of advanced 6-DOF fixed-wing guidance (*e.g.* highly constrained dynamics and coupled control relationships) suggest that supervised machine learning (ML) and deep reinforcement learning (deep RL) are appealing research areas for the field. However, supervised learning typically requires large amounts of near-optimal training data to produce a model that can perform tasks. Furthermore, the agent will never perform beyond the ability shown in the training data, *e.g* if training data is taken from an expert pilot, the agent will never outperform the pilot. Additionally, many next generation applications of UAVs—*e.g.* coordinated teaming, camera driven control, obstacle avoidance and racing behaviours—have no derived guidance model or expert pilot available. In contrast, reinforcement learning (RL) learns to improve its *policy* (*i.e.* the mapping from state to action) based on a reward function designed for its environment and/or desired behavior. This allows RL policies to learn without any training data, provided the environment dynamics can be modeled. Modern deep RL algorithms such as Proximal Policy Optimization (PPO) [5] formalize the agent as the combination of *actor* and *critic* neural networks. The actor network transforms observed states to actions while the critic network estimates the value of a given state (*i.e.* the value function). The RL algorithm updates the actor and

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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. 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. Approved for Unclassified Unlimited Release (UUR) by SAND 2023-00237C

critic networks as agents explore the environment, attempting to optimize the policy and minimize loss via gradient descent. PPO is a state-of-the-art RL algorithm for this single-agent continuous-action application because it avoids excessively large policy updates [5]. Despite the advantages of RL, its application to guidance and control of 6-DOF fixed-wing flight has proven difficult (*e.g.* [6]).

Initial stages of fixed-wing RL training can be challenging due to sparse rewards, local extrema, and unproductive policy changes. One method of mitigating these challenges is to utilize warm-starting with an expert guidance model. Warm-starting is a form of imitation learning that uses the mean squared error between the actor and a designed guidance model to optimize the actor network, providing a lower bound to optimality [7]. Warm-starting shapes the initial behavior of the actor network, allowing subsequent RL training to sample from trajectories that are closer to intended behavior. While unused during warm-starting, the critic network is also optimized by experiencing the reward landscape; this provides a better estimate of the value function when beginning RL training. This reduces training time and enables successful convergence on 6-DOF fixed-wing guidance. Warm-starting trains a policy to the level of the designed guidance model; further reinforcement learning is needed to achieve more difficult objectives, *e.g.* coordinated teaming, camera driven control, obstacle avoidance and racing behaviours, as previously mentioned. This paper shows that using a very limited warm-start period (WaSP) enables an otherwise intractable RL for guidance problem to converge, which should enable learning more complex objectives via transfer learning in future work.

In order to investigate methods to mitigate the difficulty of training continuous nonlinear 6-DOF guidance of fixed-wing aircraft, this paper contributes an the implementation of a dynamic environment for continuous nonlinear 6-DOF guidance of fixed-wing aircraft (Section III), a proof-of-concept RL guidance controller trained with a very limited warm-start period (WaSP+PPO) (Section IV and V), and a comparative analysis of method performance (Section VI). The results form a baseline that will allow comparison when considering more complex objectives.

II. Related Work

Monaco, Ward, and Barto first investigated guidance of fixed-wing aircraft with RL. They achieved discrete pitch guidance [6], but left continuous full-rotational guidance of diving flight to future work. Other research into RL fixed-wing guidance and control in limited state space have also been successful, such as control of heading [8], pitch [9], or roll [10]. Some of these efforts improved upon existing controllers by reducing overshoot [11]. Further efforts increased the environment states controlled, including fixed-wing longitudinal control [12], and heading and altitude control stabilization [13]. This increased capability allowed many autonomous guidance applications to be investigated, including perching landing [14] [15], multi-agent flocking [16] [17] [18], target tracking [19], refueling [20], and collision avoidance/pursuit evasion [16] [21] [22].

Works considering 6-DOF RL guidance and control of fixed-wing aircraft have the following three approaches. First, some approaches use routines "composed by primitive actions" [23] [24] or otherwise use action abstraction to learn on commanded states rather than control surface deflections [19] (second approach of paper). Second, some works attempt to perform pre-defined maneuvers with varying success ("the pitch profile steadily deteriorated during the maneuver") [25] [26]. Finally, there are works that investigate full 6-DOF fixed-wing flight, finding "no agents were observed to exit their turn upon reaching the target heading, which is the desired behavior" [27] and that "the agent still failed to learn to orbit the target" [19] (first approach of paper). There is a need in the existing literature to establish capable 6-DOF guidance and control of fixed-wing aircraft using RL without abstractions. With the work of [26] establishing low-level control given commanded actions, there is remaining work to establish a full-state fixed-wing guidance trained with RL to produce the exact commands used in the paper.

Warm-starting has many forms [28], though it has typically been applied to non-robotic applications [29], which found warm-starting could hurt generalization in deep neural networks. Recently however, warm-starting has been used in dynamic contexts such as the cart-pole and lunar lander [30] as well as walker and robot manipulation [28]. To the authors' knowledge, there are no known applications of warm-starting to full 6-DOF fixed wing RL guidance and control. We show that warm-starting can enable RL on this problem class and we posit that RL will advance the state of the art on this problem class in future work.

III. Fixed Wing Dynamic Model

This study considers a training environment of a simulated fixed-wing UAV, defined using the nonlinear 6-DOF dynamics of fixed-wing aircraft [1] [31]. The UAV is modeled as a rigid body with mass m and inertia tensor $\boldsymbol{I}$ defined in the body frame of reference b with origin at the center of gravity (CG). This body has associated states, forces, and moments defined in Table I. The aircraft travels in the direction of the three-dimensional velocity vector $\mathbf{V}_T$, which is described in reference to the body by the angles α and β (Fig. 2).

$$\alpha = \tan^{-1}(w/u), \qquad \beta = \sin^{-1}(v/\mathbf{V}_T) \tag{1}$$

The UAV has position and orientation relative to a North-East-Down (NED) Earth-tangent frame assumed to be inertial, denoted by i. The body frame orientation is represented in the dynamic simulation using quaternions $\vec{\mathbf{e}} = [e_0, e_1, e_2, e_3]$ and rotation matrices (*e.g* the rotation matrix from the body frame to the inertial frame $\mathbf{R}_{i/b}$) in order to prevent mathematical singularities in computation. The same orientation is represented in Euler angles, $\vec{\mathbf{\Theta}} = [\phi, \theta, \psi]$, when used for guidance and control in later sections. Precise definitions and conversions between $\vec{\mathbf{\Theta}}$, $\vec{\mathbf{e}}$, and $\mathbf{R}$ can be found in [32].

TABLE I

BODY-FIXED STATE, FORCE, AND MOMENT NOTATIONS

	Roll Axis x_b	Pitch Axis y_b	Yaw Axis z_b
Velocity components	u	v	w
Force components	X	Y	Z
Orientation	ϕ	θ	ψ
Angular rates	p	q	r
Moment components	L	M	N

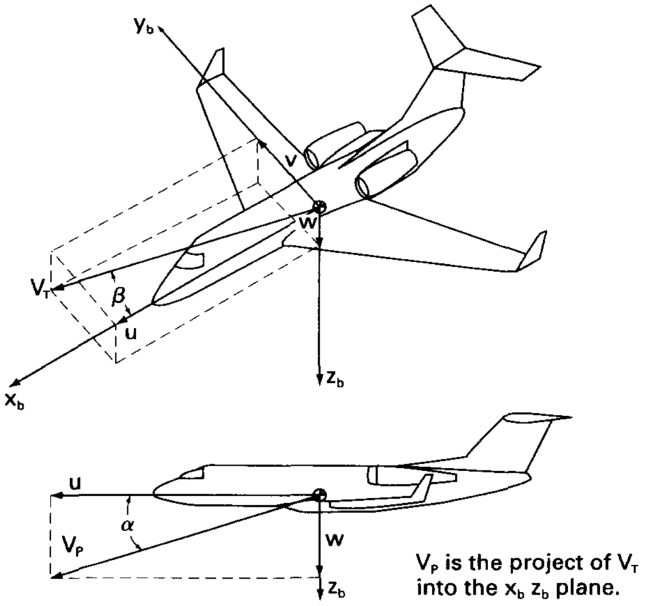

Fig. 2. Axes definition with body-frame velocity components, the 3D velocity vector V_T, and angles α and β [1].

A. Equations of Motion

The equations of the motion for a UAV are defined as a series of nonlinear first-order differential equations (*i.e.* state space representation) [31]. This includes the change in the position of the UAV body, b, relative to i, in the coordinate frame i, $\left[\dot{\vec{\mathbf{p}}}_{b/i}\right]_i$, defined as $\left[\vec{\mathbf{v}}_{b/i}\right]_b$ rotated to the frame i.

$$\left[\dot{\vec{\mathbf{p}}}_{b/i}\right]_i = \begin{bmatrix}\dot{p}_n \\ \dot{p}_e \\ \dot{p}_d\end{bmatrix} = \mathbf{R}_{i/b} \cdot \left[\vec{\mathbf{v}}_{b/i}\right]_b . \tag{2}$$

The body-frame acceleration of the UAV includes contributions from the combined aerodynamic and propulsive forces $\vec{F}_b = [X, Y, Z]$, coriolis acceleration, and body-frame gravity g_b. Wind is neglected in this study.

$$\left[\frac{^b d}{dt}\left(\vec{\mathbf{v}}_{b/i}\right)\right]_b = \begin{bmatrix}\dot{u} \\ \dot{v} \\ \dot{w}\end{bmatrix} = \mathbf{g}_b + \frac{1}{m}\vec{\mathbf{F}}_b - \left[\vec{\boldsymbol{\omega}}_{b/i}\right]_b \times \left[\vec{\mathbf{v}}_{b/i}\right]_b , \tag{3}$$

where g_b is the inertial-frame gravity (positive in NED) rotated to the body frame.

$$\mathbf{g}_b = \begin{bmatrix} g_{x_b} \\ g_{y_b} \\ g_{z_b}\end{bmatrix} = \mathbf{R}^T_{i/b} \cdot \begin{bmatrix}0 \\ 0 \\ g\end{bmatrix} \tag{4}$$

The change in quaternion-specified orientation $\left[\dot{\vec{\mathbf{e}}}_{b/i}\right]_i$ is a function of angular rates (note that e is used to distinguish from pitch rate q):

$$\left[\dot{\vec{\mathbf{e}}}_{b/i}\right]_i = \begin{bmatrix}\dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3\end{bmatrix} = \frac{1}{2}\begin{bmatrix}\xi & -p & -q & -r \\ p & \xi & r & -q \\ -q & -r & \xi & p \\ r & q & -p & \xi\end{bmatrix} \tag{5}$$

Quaternions could become problematically small or large numerically, therefore $\xi = 0.5\left(1 - ||\vec{e}||^2\right)$ is included along the diagonal to ensure near-unit quaternions. Finally, the change in angular velocity of the UAV is a function of the combined aerodynamic and propulsive moments $\vec{M}_b = [L, M, N]$ and the vector derivative of angular velocity.

$$\left[\frac{^b d}{dt}\left(\vec{\boldsymbol{\omega}}_{b/i}\right)\right]_b = \begin{bmatrix}\dot{p} \\ \dot{q} \\ \dot{r}\end{bmatrix} = \boldsymbol{I}^{-1}\left[-\left[\vec{\boldsymbol{\omega}}_{b/i}\right]_b \times \left(\boldsymbol{I}\ \left[\vec{\boldsymbol{\omega}}_{b/i}\right]_b\right) + \vec{\mathbf{M}}_b\right] \tag{6}$$

B. Body-Frame Forces and Moments

The UAV experiences nonlinear forces and moments acting on the CG that are functions of states and inputs. Aerodynamic forces and moments can be nondimensionalized into coefficients by dimensional factors wing area S, dynamic pressure Q, wing span $\flat$, and wing chord c. This results in the coefficient notation C_{-}, where $-$ is a force or moment (*e.g.* the pitch moment M is equal to $QSc\left(C_M\right)$). These forces and moments are then linearized with respect to states or inputs, which then become a sub-subscript variable in notation (*e.g.* the linear approximation of the contribution of angle of attack α to the pitching moment M is $QSc\left(C_{M_\alpha}\alpha\right)$). Similarly, C_{-_0} are coefficients that describe nondimensionalized contributions that have no state dependence, *e.g.* C_{M_0}. Some coefficients such as $C_X(\alpha)$ are not constant and have dependencies on the angle of attack α. These aerodynamic coefficients are estimated using the UAV model specified in [31]. The propulsive forces and moments are similarly modeled with scaling terms S_{pr}, C_{pr}, k_{mot}, k_{Tp}, and k_Ω (these are measured constants, see [31]). The formulation contains aerodynamic inputs ranging from $[-15°, 15°]$ for the aileron (δ_a), elevator (δ_e), and rudder (δ_r). For the aircraft considered, these are not separate control surfaces and the coefficients listed in [31] are considered to be analog quantities. The singular propulsive input, throttle (δ_T), ranges from $[0, 1]$. The total force from the contributions of aerodynamics and propulsion can then be described (where ρ is air density):

$$\vec{\mathbf{F}}_b = \begin{bmatrix}X \\ Y \\ Z\end{bmatrix} = \frac{1}{2}\rho S_{pr} C_{pr}\begin{bmatrix}\left(k_{mot}\delta_T\right)^2 - V_T^2 \\ 0 \\ 0\end{bmatrix} + QS \cdot \ldots$$
$$\begin{bmatrix} C_X(\alpha) + C_{X_q}(\alpha)\frac{c}{2V_T}q + C_{X_{\delta_e}}\delta_e \\ C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{\flat}{2V_T}p + C_{Y_r}\frac{\flat}{2V_T}r + C_{Y_{\delta_r}}\delta_r \\ C_Z(\alpha) + C_{Z_q}(\alpha)\frac{c}{2V_T}q + C_{Z_{\delta_e}}\end{bmatrix} \tag{7}$$

The total moment from both sources is then a similar equation of coefficients, states, and actions:

$$\vec{\mathbf{M}}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} -k_{Tp}\left(k_\Omega \delta_T\right) \\ 0 \\ 0 \end{bmatrix} + QS \cdot \ldots$$
$$\begin{bmatrix} b\left(C_{L_0} + C_{L_\beta}\beta + C_{L_p}\frac{b}{2V_T}p + C_{L_r}\frac{b}{2V_T}r + C_{L_{\delta_r}}\delta_r\right) \\ c\left(C_{M_0} + C_{M_\alpha}\alpha + C_{M_q}\frac{c}{2V_T}q + C_{M_{\delta_e}}\delta_e\right) \\ b\left(C_{N_0} + C_{N_\beta}\beta + C_{N_p}\frac{b}{2V_T}p + C_{N_r}\frac{b}{2V_T}r + C_{N_{\delta_r}}\delta_r\right) \end{bmatrix} \tag{8}$$

IV. Guidance Model and Low-Level Controller

A. High-Level Guidance Model

In order to warm-start the RL agent, a guidance model is designed to receive state information and generate an action vector. This input state is first converted to a state relative to the target point (the guidance objective), x_{er}, and contains the target-relative state information $[p_{n_{er}}, p_{e_{er}}, p_{d_{er}}, \psi_{er}]$. The action vector contains the commanded target-relative heading ψ_{er_c}, side slip angle β_c, relative altitude, h_{er_c} and airspeed V_{T_c} (because wind is neglected, course angle χ and heading ψ are equivalent, as well as airspeed V_a and velocity V_T).

The guidance model was designed to achieve several objectives. Firstly, the model ensures that the UAV turns towards the target point only when able to do so, *i.e.* approximately when the target point is outside of the turn diameter achievable, d.

$$\psi_{er_c} = \begin{cases} \psi_{er} & \text{if } \psi_{er} < \pi/2 \text{ or} \sqrt{{p_{n_{er}}}^2 + {p_{e_{er}}}^2} > d \\ 0 & \text{if } \psi_{er} > \pi/2 \text{ and} \sqrt{{p_{n_{er}}}^2 + {p_{e_{er}}}^2} < d \end{cases} \tag{9}$$

Secondly, when rolling, the side-slip angle β must be near zero in order to maintain altitude, so the model sets $\beta_c = 0$. Thirdly, the model commands the target altitude based on the relative vertical position $p_{d_{er}}$, setting $h_{er_c} = -p_{d_{er}}$. Finally, changes to altitude must be reflected in the commanded velocity as airspeed has a large affect on rates of climb or descent, so a continuous function is defined:

$$V_{T_c} = V_{\text{cruise}} \cdot \left(1 + \tan^{-1}\left(\frac{-p_{d_{er}}}{|p_{d_{er}}|_{max} \cdot \pi}\right)\right), \tag{10}$$

where $|p_{d_{er}}|_{max}$ is the maximum expected altitude error and V_{cruise} is the cruise velocity.

B. Low-level Proportional-Derivative Controller

The low-level (actuation) controller receives the action vector $[\psi_{er_c}, \beta_c, h_{er_c}, V_{T_c}]$ from either the RL agent or guidance model and produces actuations $[\delta_a,\ \delta_e,\ \delta_r,\ \delta_T]$ which are the settings of the ailerons, elevator, rudder, and throttle, respectively. This low-level controller was designed with proportional-derivative (PD) control and successive loop closure techniques from classical control. The gains for the PD controllers were designed using root locus analysis [31] followed by manual tuning.

1) Lateral Low-Level Controller: In order to avoid unintended dynamic coupling, the aircraft changes heading by rolling instead of yawing. Successive loop closure converts commanded relative heading ψ_{er_c} into a commanded roll angle ϕ_c with proportional control. ϕ_c is limited to the range $[-\pi/4, \pi/4]$ so that it does not interfere with longitudinal control. The heading used in this calculation is target-relative, ψ_{er}.

$$\phi_c = k_{p_\psi}(\psi_{er_c} - \psi_{er}), \quad k_{p_\psi} = 0.71 \tag{11}$$

The aileron deflection δ_a is then specified using PD control with states ϕ and p:

$$\delta_a = k_{p_\phi}(\phi_c - \phi) - k_{d_\phi}p, \quad k_{p_\phi} = 0.17, k_{d_\phi} = 0.007 \tag{12}$$

Similarly, a proportional controller was used to control β with δ_r.

$$\delta_r = -k_{p_\beta}(\beta_c - \beta), \quad k_{p_\beta} = -0.075 \tag{13}$$

2) Longitudinal Low-Level Controller: A small aircraft must pitch to change altitude, therefore successive loop closure is used to translate commanded relative altitude h_{er_c} into a command in pitch orientation θ_c.

$$\theta_c = k_{p_h}(h_{er_c} - p_{d_{er}}), \quad k_{p_h} = 0.001 \tag{14}$$

The subsequent elevator deflection δ_e is specified using PD control with states θ and q.

$$\delta_e = k_{p_\theta}(\theta_c - \theta) - k_{d_\theta}q, \quad k_{p_\theta} = k_{d_\theta} = -0.7 \tag{15}$$

Throttle δ_T is controlled with δ^*, the trim throttle setting, and proportional control with velocity V_T.

$$\delta_T = \delta^{T^*} + k_{p_v}(V_{T_c} - V_T), \quad \delta^{T^*} = 0.76, k_{p_v} = 0.06 \tag{16}$$

V. Methods

This problem is formalized as a fully observable, deterministic Markov Decision Process (MDP) [33] with state space S and action space A. The state uses target-relative position coordinates and heading.

$$S = \{p_{n_{er}}, p_{e_{er}}, p_{d_{er}}, u, v, w, \phi, \theta, \psi_{er}, p, q, r\} \tag{17}$$

Similarly, the action space uses a target-relative commanded altitude and heading:

$$A = \{\psi_{er_c}, \beta_c, h_{er_c}, V_{T_c}\} \tag{18}$$

The RL algorithm observes the states s_t at some time t and takes some action a_t based on its policy. The agent then receives a reward r_{t+1} based on the new state s_{t+1} as determined by the action taken and the environment dynamics. This custom environment is constructed using the OpenAI Gym environment framework [34].

The aircraft selected for study is the Zagi Flying Wing aircraft shown in Fig. 1. The defining aerodynamic, propulsive, and descriptive measurements for this aircraft are taken as specified in [31]. The dynamic environment will interface

with the PPO and warm-start agents as shown in Fig. 3. The full relative state vector $\mathbf{x}_{er}$ with Euler angle orientation is scaled to intended element ranges of $[-1, 1]$. This observation vector is used as the input vector for both the RL agent and warm-start agent, the latter descales the inputs before calculation.

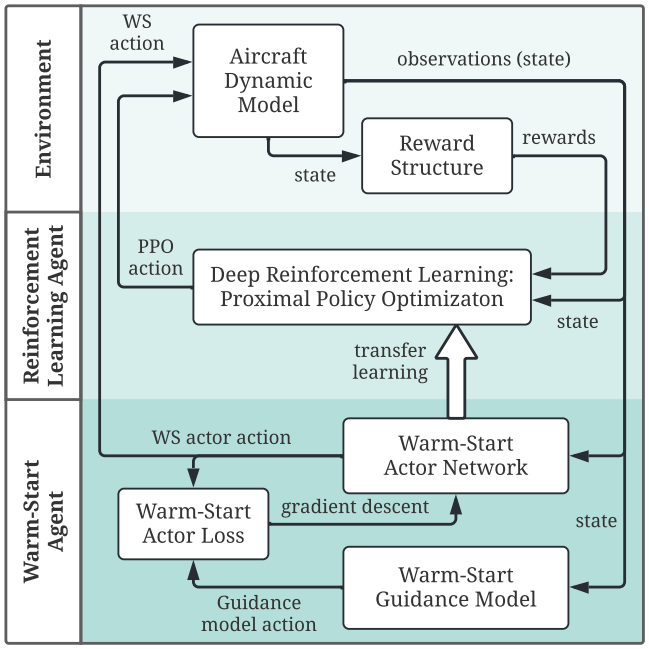

Fig. 3. Agent-environment configuration for the flight guidance problem with PPO and warm-starting

A. PPO and Warm-Start Optimization

Proximal Policy Optimization (PPO) was selected due to the appealing proximal aspect of PPO, *i.e.* the intentionally conservative policy update "which attains the reliable performance of TRPO, while using only first-order optimization" [5]. Similarly, [35] highlights PPO as the best performing algorithm for quadcopter control, which uses the a similar nonlinear kinematics model as in Section III-A.

1) The Mechanics of PPO: Of all probability distributions that map states to actions (or policies π), the optimal policy π^* can be stated as an argument maximum of the expected reward over a trajectory over all policies [36]. This trajectory τ is a series of actions and associated states sampled from the environment. By shaping rewards, *i.e.* defining what outcomes result in a positive or negative reward over a trajectory, $R(\tau)$, the RL algorithm will learn to generate trajectories that result in intended goals.

$$\pi^* = \arg\max_{\pi} \underset{\tau\sim\pi}{\mathbb{E}} \left[R(\tau)\right]. \tag{19}$$

Proximal Policy Optimization (PPO) [5] uses a *surrogate objective* that seeks to conservatively update the policy based on the advantage, an estimate of the relative value of an action compared to the expected value over a distribution of actions, as opposed to the reward directly. This is evaluated with the *action-value function* Q^π, an expectation of return for the distribution of explored actions from state s given that you act according to policy π, and the *value function* V^π, an expectation of return if you start in state s and always act according to policy π [36].

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{20}$$

The policy learns through *gradient descent*, where the actor and critic networks are optimized in the direction of negative loss gradient. When optimizing based on the advantage, PPO clips potentially detrimental changes to the policy, removing the incentive for change if the probability ratio of the new policy to the old policy, $r_t = \pi/\pi_{old}$, is outside a region around the current point of optimization for a given state-action pair. This region is defined as $[1-\epsilon_{PPO}, 1+\epsilon_{PPO}]$ where ϵ_{PPO} is a chosen hyperparameter. Minimization of the *actor loss*, L_a, (a form of error) will allow the actor to produce high-value actions.

$$L_a(\pi) = \mathbb{E}\left[\min\left(r_t A^\pi, \text{clip}\left(r_t, 1-\epsilon_{PPO}, 1+\epsilon_{PPO}\right) A^\pi\right)\right] \tag{21}$$

In order to calculate advantage for use in actor loss, an estimate of the value function, denoted $V_\phi(s)$, is also learned through gradient descent. Use of the sum of discounted future rewards $\hat{R}$, and minimization of *critic loss*, L_c, allows the critic network to estimate the value function.

$$L_c(V_\phi(s)) = \mathbb{E}\left[\left(V_\phi(s) - \hat{R}\right)^2\right] \tag{22}$$

For this study, the neural networks (NNs) used for actor and critic are feed-forward multi-layer perceptrons (MLP). Noise is added to the output of the actor to encourage exploration during training, but it is deterministic during evaluation.

2) The Mechanics of Warm-Starting: As noted in [37], PPO may progress slowly and become stuck in local optima, which results in a poor or incapable policy. Hyperparameter tuning, reward shaping, and curriculum learning can help overcome such difficulties, but if they do not achieve intended performance, warm-starting should also be tried. Warm-starting is a form of imitation learning that trains the actor network to produce outputs that match a designed model (*e.g.* a guidance model) by using the mean squared error between the actor network's generated output $\boldsymbol{\mu}_\pi$, the mean of the distribution, and the models' command $\mathbf{x}_c$.

$$J(\pi) = \mathbb{E}\left[\left(\boldsymbol{\mu}_\pi - \mathbf{x}_c\right)^2\right] \tag{23}$$

Minimizing this alternative actor loss directly trains the actor network, while allowing the critic network to learn by experiencing the state-space and reward landscape. It can then be disabled and an RL algorithm can begin training on the initialized actor and critic networks, giving the algorithm a range of feasible and high value actions from the start. This transfer learning can lead to convergence on RL problems that would stall on uninitialized networks, as we show in the results.

B. Reward Shaping

With the environment, inputs, actions, and algorithm defined, reward shaping is needed in order to appropriately describe the intended goals of an agent's guidance objectives. The reward for an agent entering a small distance from the target is 1, which describes the intended behavior.

$$R_{tgt} = 1, \text{if agent enters } \left(\left|\left| \left[\dot{\vec{\mathbf{p}}}_{b/_i} \right]_i \right|\right|_2^2 \leq r^2 \right) \tag{24}$$

If the NN generates an action outside of the realizable command space (*e.g.* negative airspeed) then the agent is penalized $R_{ba} = -1$ for that action. Finally, the convex penalty $R_{x_{er}}$ is applied for incremental encouragement of advantageous exploration. Smaller states are encouraged for distance convergence and smoother flight.

$$R_{\mathbf{x}_{er}} = -\mathbf{x}_{er}\mathbf{Q}\mathbf{x}_{er}, \tag{25}$$

This is a convex penalty where $\mathbf{Q}$ is a diagonal matrix with elements $4 \cdot 10^{-7} \cdot [10, 10, 10, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$, designed such that over the expected number of steps needed to reach the target, the sum of the $R_{x_{er}}$ penalties is limited to $\sum_0^{T_f} R_{x_{er}} < 1$. The total reward for a time step is then:

$$R_{total} = R_{tgt} + R_{ba} + R_{x_{er}} \tag{26}$$

C. Training Considerations

Across hyperparameters, networks with hidden layer sizes of 5x512 nodes performed best, with smaller networks converging more slowly, and larger networks producing diminishing returns. Thus, a hidden network size of 5x512 was chosen. Hyperparameters were hand-tuned to the values in Table II to avoid policy collapse and optimize agent performance. The discount factor, γ, was further tuned considering the large amount of time steps needed for the average agent to reach the target [5].

TABLE II
HYPERPARAMETER VALUES

Lrn.rate	Std.dev.	γ	λ_{GAE}	ϵ_{PPO}	Crt.dis.	Minibatch	Epochs$_{PPO}$
$2 \cdot 10^{-5}$	.05	.99995	0.95	.05	.5	1024	10

The environment and algorithm are then parallelized to simultaneous fixed-wing simulation aircraft *agents* by vectorizing the calculations. The guidance policy is trained through a series of episodes that begin at a randomized state or *pose* and move according to the state and action until either the singular terminal condition (*i.e.*, the target is reached) or until the end of the episode. These initialized poses are a Gaussian distribution with mean and standard deviation as follows:

TABLE III
RANGE FOR ONE STANDARD DEVIATION OF INITIALIZED STATES

position	u velocity	orientations	rotation rate
$0 \pm 500m$	17 ± 13 m/s	$0 \pm \pi/4$ (yaw: $0 \pm \pi$) rad	0 ± 1 rad/s

D. Actor Evaluation

Four actor evaluation tests were used to measure what percentage of 64 agents reach the target from random initial conditions with a set seed. The standard test (Test 1) is identical to training. After a sufficient number of time steps, the policy is evaluated on percentage of successful runs. The three other tests are similar to the standard test except for the following differences. The state generalization test (Test 2) expands initial conditions by a factor of 1.5. The noise test (Test 3) includes noise on target sensing in the form of target movement, moving position a standard deviation of 1m per time step in each direction. Finally, in the motion generalization test (Test 4), the location of the target follows a circular pattern with low-frequency vertical motion at a speed of 11 m/s. These four tests evaluate the effectiveness of a policy to learn guidance and generalize to unseen conditions.

VI. RESULTS AND DISCUSSION

Three sets of ten policies were trained to compare the effectiveness of PPO with and without a limited warm-start period (WaSP). The first set was trained with PPO only for 33000 time steps on 32 parallel agents, which took about four hours an Intel i7-8650U CPU. The second set was trained with a warm-start period (WaSP only) for 512 time steps on 32 parallel agents and saved for evaluation. The brevity of this period intentionally restricts the performance of the WaSP only policies to act as a starting point for subsequent PPO training. These warm-started policies are then trained using PPO for 32000 time steps on 32 parallel agents thereby resulting in PPO policies with a limited warm-start period (WaSP+PPO). The training time for WaSP+PPO was approximately 4 total hours on an Intel i7-8650U CPU.

A. Comparison of Trained Policy Sets

The 10 PPO only policies, 10 WaSP only policies, and 10 WaSP+PPO policies were evaluated using the standard test (Test 1) of Section V-D. The results are collected in Fig. 4, showing the distribution of policies sorted by percentage of agents that reached the target.

Results show that PPO only policies are only able to guide an average of 2% of agents to the target. WaSP policies averaged 32% with a relatively even distribution, and WaSP+PPO policies have an average performance of 57%, but with an approximately bimodal distribution. The successful cluster contains 5 policies that have an average of 95% of agents reach the target while the failed cluster has 4 policies that complete an average of 10% of agents.

As seen in Fig. 5, PPO only policies failed to properly command heading, similar to [27] and [19]. Agents often spiral out from initial conditions, in a way which might be locally optimal, as some agents can hit the target with this approach by falling from a random starting location. In contrast, WaSP+PPO policies demonstrated guidance that took into account the target's position. Investigation of failed WaSP+PPO trajectories show a tendency to spiral around the target, untrained to the long term requirement to fly past

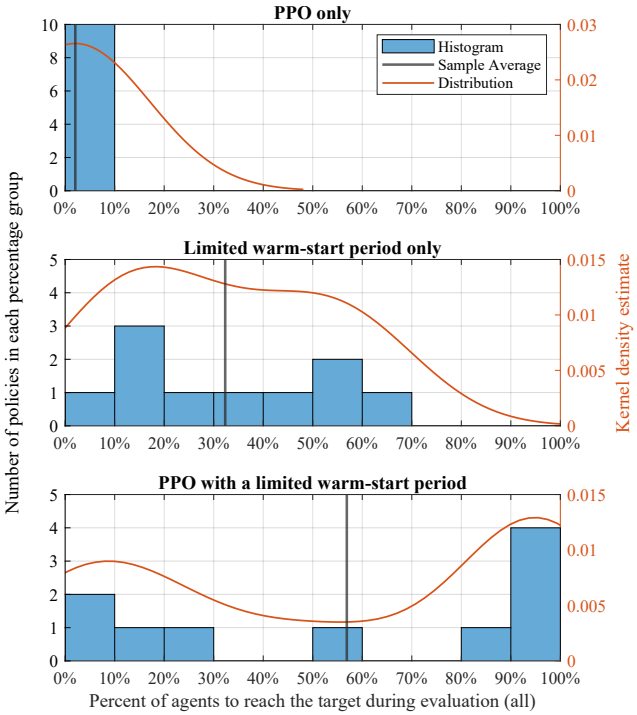

Fig. 4. Histogram of evaluated policies for each training method (note y axes ranges). Results show an improvement with averages of 2% for PPO only, to 32% for WaSP only, then to 57% for WaSP+PPO.

the target until outside the vehicle's turning diameter. The guidance model in Section IV-A is designed to account for this in Equation 9. This behavior was learned by successful WaSP+PPO policies and some WaSP+PPO trajectories were able to improve upon the designed guidance model with a tighter turn and a shorter path to target.

B. Comparison of Best Actors from Each Method

To serve as a consensus standard of comparison, an additional set of policies was trained using the Stable-Baselines 3 (SB3) [38] implementation of PPO without any warm-starting for 10^6 time steps. As a best case analysis, the most successful agents from PPO only, WaSP only, and WaSP+PPO were evaluated on tests 1-4.[1] The guidance model from Section IV-A, the best PPO only policy from SB3, and a random action baseline were also included as benchmarks.

The WaSP+PPO method had marked improvement over all other RL methods in the standard test, state generalization test, and noise test, but performed marginally in the motion generalization test. In contrast, the PPO only methods performed similarly to the random action baseline. The results are comparable between the parallelized version of PPO [36] and the SB3 implementation [38].

Comparing the WaSP+PPO policies to the performance of the designed guidance model showed comparable performance for the standard test, state generalization test, and noise test, but not the motion generalization test. This is likely due to the fact that the WaSP+PPO agent was not trained for motion, while PD control is characteristically robust; the classical guidance model works across all initial conditions, is robust to sensing noise, and can track a moving reference. These tests represent baseline performance metrics for RL fixed-wing guidance for relatively simple tasks. Successful execution of these objectives consequently enables application of RL guidance to more complex use cases that require more generalized behavior than PD guidance affords (*i.e.*, cases where no explicit expert system baseline exists).

[1]The performance of these polices during testing is included as a supplementary downloadable video. To aid in video clarity the number of agents was reduced from 64 to 32 and the evaluation period was abbreviated. This supplementary video is available at autonomy.sandia.gov/warmstart.

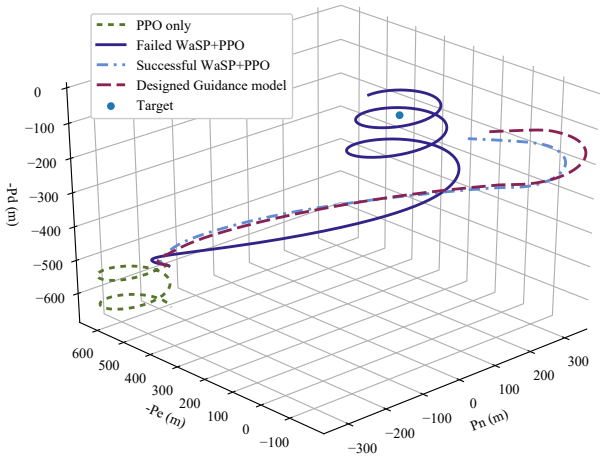

Fig. 5. 3D trajectories from PPO only, failed WaSP+PPO, successful WaSP+PPO, and designed guidance model agents. The trajectories start in the same initial condition.

TABLE IV
PERFORMANCE OF BEST ACTORS AND DESIGNED GUIDANCE MODEL

Algorithm evaluated	Std. test (Test 1)	Gen. test (Test 2)	Noise test (Test 3)	Motn. test (Test 4)
Random actions	3.1%	1.6 %	3.1%	1.6%
PPO only	3.1%	3.1%	4.7%	1.6%
SB3 PPO only [38]	4.7%	3.1%	4.7%	3.1%
WaSP only	60.9%	57.8%	51.6%	9.4%
WaSP+PPO	**100%**	**100%**	**98.4%**	**37.5%**
G. model (no ML)	100%	100%	100%	61.2%

VII. CONCLUSION

This paper demonstrates successful use of a warm-start period for training on dynamic environments as well as a proof-of-concept PPO-trained guidance autopilot for fixed-wing UAVs. PPO with a limited warm-start period was able to train a 6-DOF fixed-wing guidance loop that, on average, improved performance of standard PPO policies by a factor of 28 and performance of warm-start period only policies by a factor of 1.7. Though variation exists, this technique results in some policies which consistently achieve 100% success at the intended task and is able to generalize to new initial conditions and noise. The more agile flight found by some warm-start period plus PPO agents suggest that training RL policies with this method can improve performance over the

guidance model initially imitated. Finally, this method may improve the effectiveness of RL on other highly constrained nonlinear dynamic environments, such as robotic locomotion, pick and place tasks, or autonomous vehicles.

A. Future Work

Currently planned work aims to mature a hardware implementation of an RL guidance autopilot to achieve RL-guided fixed-wing flight[2]. Further efforts may use this method in conjunction with the low-level RL controller in [26] to investigate a full-stack (*i.e.* state to action) fixed-wing RL controller, or even an end-to-end system (*i.e.* full stack neural agents with sensor input). These advances in 6-DOF fixed-wing RL should result in improvements over state-of-the-art in guidance under highly-complex objectives.

REFERENCES

[1] Robert C. Nelson. Flight Stability and Automatic Control. McGraw-Hill, 1998. Image used with permission, license ID 1145864-1.

[2] Hassan K. Khalil. Nonlinear Systems. Prentice Hall, 2002.

[3] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678, 2016.

[4] Zagi LLC. Zagi hp electric flying wing, 2016. zagi.com.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[6] Jeffrey F. Monaco, David G. Ward, and Andrew G. Barto. Automated aircraft recovery via reinforcement learning: Initial experiments. In Proceedings of the 10th International Conference on Neural Information Processing Systems, NIPS'97, page 1022–1028, Cambridge, MA, USA, 1997. MIT Press.

[7] Reed Jensen. Co-adaptive human machine teaming with a reinforcement learning agent, 2019. Found at: ilp.mit.edu/sites/default/files/2020-01/Jensen_FINAL_ILP_20191113_ver_9_compressed.pdf.

[8] Steve Kimathi. Application of reinforcement learning in heading control of a fixed wing uav using x-plane platform. International Journal of Scientific and Technology Research, 6:285–290, 02 2017.

[9] Daichi Wada, Sergio A. Araujo-Estrada, and Shane Windsor. Unmanned aerial vehicle pitch control using deep reinforcement learning with discrete actions in wind tunnel test. Aerospace, 8(1), 2021.

[10] Jae-Hung Han, Dong-Kyu Lee, Jun-Seong Lee, and Sang-Joon Chung. Teaching micro air vehicles how to fly as we teach babies how to walk. Journal of Intelligent Material Systems and Structures, 24(8):936–944, 2013.

[11] Yan Zhen and Mingrui Hao. Aircraft control method based on deep reinforcement learning. In 2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS), pages 912–917, 2020.

[12] Jun H. Lee and Erik-Jan Van Kampen. Online reinforcement learning for fixed-wing aircraft longitudinal control. In AIAA Scitech 2021 Forum, 2021.

[13] Sheng Zhang, Xin Du, Juan Xiao, Jiangtao Huang, and Kaifeng He. Reinforcement learning control for 6 dof flight of fixed-wing aircraft. In 2021 33rd Chinese Control and Decision Conference (CCDC), pages 5454–5460, 2021.

[14] Rick Cory and Russ Tedrake. Experiments in fixed-wing uav perching. In AIAA Guidance, Navigation and Control Conference and Exhibit, 08 2008.

[15] Antony Waldock, Colin Greatwood, Francis Salama, and Thomas Richardson. Learning to perform a perched landing on the ground using deep reinforcement learning. Journal of Intelligent and Robotic Systems, 92(3-4):685–704, 2017.

[16] Chao Yan, Xiaojia Xiang, Chang Wang, and Zhen Lan. Flocking and collision avoidance for a dynamic squad of fixed-wing uavs using deep reinforcement learning. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4738–4744. IEEE, 2021.

[2]A neural-network-in-the-loop flight is included in the supplementary video available at autonomy.sandia.gov/warmstart.

[17] Shao-Ming Hung and Sidney N. Givigi. A q-learning approach to flocking with uavs in a stochastic environment. IEEE Transactions on Cybernetics, 47(1):186–197, 2017.

[18] Chang Wang, Chao Yan, Xiaojia Xiang, and Han Zhou. A continuous actor-critic reinforcement learning approach to flocking with fixed-wing uavs. In Proceedings of The Eleventh Asian Conference on Machine Learning, volume 101 of Proceedings of Machine Learning Research, pages 64–79. PMLR, 17–19 Nov 2019.

[19] Hannah Lehman, Shelby Hackett, and John Valasek. Addressing undesirable emergent behavior in deep reinforcement learning uas ground target tracking. In AIAA Scitech 2022 Forum, 2022.

[20] John Valasek, Kiran Gunnam, Jennifer Kimmett, Monish D. Tandale, John L. Junkins, and Declan Hughes. Vision-based sensor and navigation system for autonomous air refueling. Journal of Guidance, Control, and Dynamics, 28(5):979–989, 2005.

[21] Bogdan Vlahov, Eric Squires, Laura Strickland, and Charles Pippin. On developing a uav pursuit-evasion policy using reinforcement learning. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pages 859–864, 2018.

[22] Qiao Cheng, Xiangke Wang, Jian Yang, and Lincheng Shen. Automated enemy avoidance of unmanned aerial vehicles based on reinforcement learning. Applied Sciences, 9(4), 2019.

[23] Adrian P Pope, Jaime S Ide, Daria Mićović, Henry Diaz, David Rosenbluth, Lee Ritholtz, Jason C Twedt, Thayne T Walker, Kevin Alcedo, and Daniel Javorsek. Hierarchical reinforcement learning for air-to-air combat. In 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pages 275–284. IEEE, 2021.

[24] Hao Wang and Weijia Wang. Autonomous control of fixed-wing unmanned aerial system by reinforcement learning. In 2020 3rd International Conference on Unmanned Systems (ICUS), pages 911–916, 2020.

[25] Shanelle G. Clarke and Inseok Hwang. Deep reinforcement learning control for aerobatic maneuvering of agile fixed-wing aircraft. In AIAA Scitech 2020 Forum, 2020.

[26] Eivind Bøhn, Erlend M. Coates, Signe Moe, and Tor Ame Johansen. Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In 2019 International Conference on Unmanned Aircraft Systems (ICUAS), pages 523–533, 2019.

[27] Gordon Rennie. Autonomous Control of Simulated Fixed Wing Aircraft using Deep Reinforcement Learning. PhD thesis, Master's thesis. The University of Bath, Bath, United Kingdom, 2018.

[28] Ching-An Cheng, Xinyan Yan, Nolan Wagener, and Byron Boots. Fast policy learning through imitation and reinforcement. CoRR, abs/1805.10413, 2018.

[29] Jordan Ash and Ryan P Adams. On warm-starting neural network training. Advances in Neural Information Processing Systems, 33:3884–3894, 2020.

[30] Andrew Silva and Matthew Gombolay. Encoding human domain knowledge to warm start reinforcement learning. Proceedings of the AAAI Conference on Artificial Intelligence, 35(6):5042–5050, May 2021.

[31] Randal W. Beard and Timothy W. McLain. Small unmanned aircraft: Theory and practice. Princeton University Press, 2012.

[32] Brian L. Stevens, Frank L. Lewis, and Eric N. Johnson. Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems. Wiley, 2016.

[33] Richard Bellman. A markovian decision process. Journal of Mathematics and Mechanics, 6(5):679–684, 1957.

[34] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. cite arxiv:1606.01540.

[35] William Koch, Renato Mancuso, Richard West, and Azer Bestavros. Reinforcement learning for UAV attitude control. CoRR, abs/1804.04154, 2018.

[36] Openai spinning up documentation, 2018. spinningup.openai.com.

[37] Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. Ppo-cma: Proximal policy optimization with covariance matrix adaptation. In 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), pages 1–6. IEEE, 2020.

[38] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2019.