Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization

Eivind Bøhn[1], Erlend M. Coates[2,3], Signe Moe[1,3], Tor Arne Johansen[2,3]

***Abstract*—Contemporary autopilot systems for unmanned aerial vehicles (UAVs) are far more limited in their flight envelope as compared to experienced human pilots, thereby restricting the conditions UAVs can operate in and the types of missions they can accomplish autonomously. This paper proposes a deep reinforcement learning (DRL) controller to handle the nonlinear attitude control problem, enabling extended flight envelopes for fixed-wing UAVs. A proof-of-concept controller using the proximal policy optimization (PPO) algorithm is developed, and is shown to be capable of stabilizing a fixed-wing UAV from a large set of initial conditions to reference roll, pitch and airspeed values. The training process is outlined and key factors for its progression rate are considered, with the most important factor found to be limiting the number of variables in the observation vector, and including values for several previous time steps for these variables. The trained reinforcement learning (RL) controller is compared to a proportional-integral-derivative (PID) controller, and is found to converge in more cases than the PID controller, with comparable performance. Furthermore, the RL controller is shown to generalize well to unseen disturbances in the form of wind and turbulence, even in severe disturbance conditions.**

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are employed extensively to increase safety and efficiency in a plethora of tasks such as infrastructure inspection, forest monitoring, and search and rescue missions. Many tasks can however not be accomplished fully autonomously, due to several limitations of autopilot systems. Low-level stabilization of the UAV's attitude provided by the inner control loops is increasingly difficult, due to various nonlinearities, as the attitude and airspeed deviates from stable, level conditions. The outer control layers providing path planning and guidance has to account for this, and settle for non-agile and safe plans. Equipping the autopilot with the stabilization skills of an experienced pilot would allow fully autonomous operation in turbulent or otherwise troublesome environments, such as search and rescue missions in extreme weather conditions, as well as increasing the usefulness of the UAV by for instance allowing the UAV to fly closer to its targets for inspection purposes.

[1]E. Bøhn and S. Moe are with the Department of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway

[2]E. M. Coates and T. A. Johansen are with the Centre of Autonomous Marine Operations and Systems (NTNU AMOS)

[3]E. M. Coates, S. Moe and T. A. Johansen are with the Department of Engineering Cybernetics, at the Norwegian University of Science and Technology, Trondheim, Norway

Corresponding Author: eivind.bohn@sintef.no

Autopilots for fixed-wing UAVs, as illustrated in Figure 1, are typically designed using cascaded single-variable loops under assumptions of decoupled longitudinal and lateral motion, using classical linear control theory [1]. The dynamics of fixed-wing aircraft including UAVs are however strongly coupled and nonlinear. Nonlinear terms in the equations of motion include kinematic nonlinearities (rotations and coriolis effects), actuator saturation and aerodynamic nonlinearities, which are also uncertain and difficult to model. The decoupled and linear designs are reliable and well-tested for nominal flight, but also requires conservative safety limits in the allowable range of flight conditions and maneuvers (flight envelope protection), because linear controllers applied to nonlinear systems typically result in a limited region of attraction [2]. This motivates the use of state-of-the-art nonlinear control algorithms.

Fig. 1: Skywalker X8 Fixed-Wing UAV

Examples of nonlinear control methods applied to UAVs include gain scheduling [3], linear parameter varying (LPV) control [4], dynamic inversion (feedback linearization) [5], adaptive backstepping [6], sliding mode control [7], nonlinear model predictive control [8], nonlinear H-infinity control [9], dynamic inversion combined with mu-synthesis [10], model reference adaptive control [11] and L1 adaptive control [12]. Automated agile and aerobatic maneuvering is treated in [13] and [14]. Several of these methods require a more or less accurate aerodynamic model of the UAV. A model-free method based on fuzzy logic can be found in [15]. Fuzzy control falls under the category of intelligent control systems, which also includes the use of neural networks. An adaptive backstepping controller using a neural network to compensate for aerodynamic uncertainties is given in [16], while a genetic neuro-fuzzy approach for attitude control is taken in [17]. The state of the art in intelligent flight control of small UAVs is discussed in [18].

Control of small UAVs requires making very fast con-

trol decisions with limited computational power available. Sufficiently sophisticated models incorporating aerodynamic nonlinearities and uncertainties with the necessary accuracy to enable robust real-time control may not be viable under these constraints. Biology suggests that a bottom-up approach to control design might be a more feasible option. Birds perform elegant and marvelous maneuvers and are able to land abruptly with pinpoint accuracy utilizing stall effects. Insects can hover and zip around with astonishing efficiency, in part due to exploiting unsteady, turbulent aerodynamic flow effects [1]. These creatures have developed the skills not through careful consideration and modeling, but through an evolutionary trial-and-error process driven by randomness, with mother nature as a ruthless arbiter of control design proficiency. In similar bottom-up fashion, machine learning (ML) methods have shown great promise in uncovering intricate models from data and representing complex nonlinear relations from its inputs to its outputs. ML can offer an additional class of designs through learning that are not easily accessible through first principles modeling, exhibiting antifragile properties where unexpected events and stressors provide data to learn and improve from, instead of invalidating the design. It can harbor powerful predictive powers allowing proactive behaviour, while meeting the strict computation time budget in fast control systems.

Reinforcement learning (RL) [19] is a subfield of ML concerned with how agents should act in order to maximize some measure of utility, and how they can learn this behaviour from interacting with their environment. Control has historically been viewed as a difficult application of RL due to the continuous nature of these problems' state and action spaces. Furthermore, the task has to be sufficiently nonlinear and complex for RL to be an appropriate consideration over conventional control methods in the first place. To apply tabular methods one would have to discretize and thus suffer from the consequences of the curse of dimensionality from a discretization-resolution appropriate to achieve acceptable control. The alternative to tabular methods require function approximation, which has to be sophisticated enough to handle the dynamics of the task, while having a sufficiently stable and tractable training process to offer convergence. Neural networks (NNs) are one of few models which satisfy these criteria: they can certainly be made expressively powerful enough for many tasks, but achieving a stable training phase can be a great challenge. Advances in computation capability and algorithmic progress in RL, reducing the variance in parameter updates, have made deep neural networks (DNNs) applicable to RL algorithms, spawning the field of deep reinforcement learning (DRL). DNNs in RL algorithms provide end-to-end learning of appropriate representations and features for the task at hand, allowing algorithms to solve classes of problems previously deemed unfit for RL. DRL has been applied to complex control tasks such as motion control of robots [20] as well as other tasks where formalizing a strategy with other means is difficult, e.g. game playing [21].

A central challenge with RL approaches to control is the low sample efficiency of these methods, meaning they need a large amount of data before they can become proficient. Allowing the algorithm full control to learn from its mistakes is often not a viable option due to operational constraints such as safety, and simulations are therefore usually the preferred option. The simulation is merely an approximation of the true environment. The model errors, i.e. the differences between the simulator and the real world, is called the reality gap. If the reality gap is small, then the low sample efficiency of these methods is not as paramount, and the agent might exhibit great skill the first time it is applied to the real world.

The current state-of-the-art RL algorithms in continuous state and action spaces, notably deep deterministic policy gradient (DDPG) [22], trust region policy optimization (TRPO) [23], proximal policy optimization (PPO) [24] and soft actor-critic (SAC) [25], are generally policy-gradient methods, where some parameterization of the policy is iteratively optimized through estimating the gradients. They are model-free, meaning they make no attempt at estimating the state-transition function. Thus they are very general and can be applied to many problems with little effort, at the cost of lower sample efficiency. These methods generally follow the actor-critic architecture, wherein the actor module, i.e. the policy, chooses actions for the agent and the critic module evaluates how good these actions are, i.e. it estimates the expected long term reward, which reduces variance of the gradient estimates.

The premise of this research was to explore the application of RL methods to low-level control of fixed-wing UAVs, in the hopes of producing a proof-of-concept RL controller capable of stabilizing the attitude of the UAV to a given attitude reference. To this end, an OpenAI Gym environment [26] with a flight simulator tailored to the Skywalker X8 flying wing was implemented, in which the RL controller is tasked with controlling the attitude (the roll and pitch angles) as well as the airspeed of the aircraft. Aerodynamic coefficients for the X8 are given in [27]. The flight simulator was designed with the goal of being valid for a wide array of flight conditions, and therefore includes additional nonlinear effects in the aerodynamic model. The software has been made openly available [28, 29]. Key factors impacting the final performance of the controller as well as the rate of progression during training were identified. To the best of the authors' knowledge, this is the first reported work to use DRL for attitude control of fixed-wing UAVs.

The rest of the paper is organized as follows. First, previous applications of RL algorithms to UAVs are presented in Section II, and the aerodynamic model of the Skywalker X8 fixed-wing UAV is then introduced in Section III. Section IV outlines the approach taken to develop the RL controller, presenting the configuration of the RL algorithm and the key design decisions taken, and finally how the controller is evaluated. In Section V, the training process and its major aspects are presented and discussed, and the controller is evaluated in light of the approach described in the preceding section. Finally, Section VI offers some final remarks and suggestions for further work.

II. RELATED WORK

In general, the application of RL to UAV platforms has been limited compared to other robotics applications, due to data collection with UAV systems carrying significant risk of fatal damage to the aircraft. RL have been proposed as a solution to many high level tasks for UAVs such as the higher level path planning and guidance tasks, alongside tried and tested traditional controllers providing low-level stabilization. In the work of Gandhi et al. [30] a UAV is trained to navigate in an indoor environment by gathering a sizable dataset consisting of crashes, giving the UAV ample experience of how NOT to fly. In [31], the authors tackle the data collection problem by constructing a pseudo flight environment in which a fixed-wing UAV and the surrounding area is fitted with magnets, allowing for adjustable magnetic forces and moments in each degree of freedom (DOF). In this way, the UAV can be propped up as one would do when teaching a baby to walk, and thereby experiment without fear of crashing in a setting more realistic than simulations.

Imanberdiyev et al. [32] developed a model-based RL algorithm called TEXPLORE to efficiently plan trajectories in unknown environments subject to constraints such as battery life. In [33], the authors use a model predictive controller (MPC) to generate training data for an RL controller, thereby guiding the policy search and avoiding the potentially catastrophic early phase before an effective policy is found. Their controller generalizes to avoid multiple obstacles, compared to the singular obstacle avoided by the MPC in training, does not require full state information like the MPC does, and is computed at a fraction of the time. With the advent of DRL, it has also been used for more advanced tasks such as enabling intelligent cooperation between multiple UAVs [34], and for specific control problems such as landing [35]. RL algorithms have also been proposed for attitude control of other autonomous vehicles, including satellites [36] and underwater vehicles. Carlucho et al. [37] applies an actor-critic DRL algorithm to low-level attitude control of an autonomous underwater vehicle (AUV) — similar to the proposed method in this paper — and find that the derived control law transfers well from simulation to real world experiments.

Of work addressing problems more similar in nature to the one in this paper, i.e. low-level attitude control of UAVs, one can trace the application of RL methods back to the works of Bagnell and Schneider [38] and Ng et al. [39], both focusing on helicopter UAVs. Both employed methods based on offline learning from data gathered by an experienced pilot, as opposed to the online self-learning approach proposed in this paper. The former focuses on control of a subset of the controllable states while keeping the others fixed, whereas the latter work extends the control to all six degrees of freedom. In both cases, the controllers exhibit control performance exceeding that of the original pilot when tested on real UAVs. In [40], the latter work was further extended to include difficult aerobatic maneuvers such as forward flips and sideways rolls, significantly improving upon the state-of-the-art. Cory and Tedrake [41] presents experimental data of a fixed-wing UAV perching maneuver using an approximate optimal control solution. The control is calculated using a value iteration algorithm on a model obtained using nonlinear function approximators and unsteady system identification based on motion capture data. Bou-Ammar et al. [42] compared an RL algorithm using fitted value iteration (FVI) for approximation of the value function, to a non-linear controller based on feedback linearization, on their proficiency in stabilizing a quadcopter UAV after an input disturbance. They find the feedback-linearized controller to have superior performance. Recently, Koch et al. [43] applied three state-of-the-art RL algorithms to control the angular rates of a quadcopter UAV. They found PPO to perform the best of the RL algorithms, outperforming the proportional-integral-derivative (PID) controller on nearly every metric.

III. UAV MODEL

Following [1], the UAV is modeled as a rigid body of mass m with inertia tensor $\boldsymbol{I}$ and a body frame $\{b\}$ rigidly attached to its center of mass, moving relative to a north-east-down (NED) frame assumed to be inertial $\{n\}$. To allow for arbitrary attitude maneuvers during simulation, the attitude is represented using unit quaternions $\boldsymbol{q} = [\eta \ \epsilon_1 \ \epsilon_2 \ \epsilon_3]^T$ where $\boldsymbol{q}^T\boldsymbol{q} = 1$. The time evolution of the position $\boldsymbol{p} = [x \ y \ z]^T$ and attitude $\boldsymbol{q}$ of the UAV is governed by the kinematic equations

$$\dot{\boldsymbol{p}} = \boldsymbol{R}_b^n(\boldsymbol{q})\boldsymbol{v} \tag{1}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -\boldsymbol{S}(\boldsymbol{\omega}) \end{bmatrix}\boldsymbol{q} \tag{2}$$

where $\boldsymbol{v} = [u \ v \ w]^T$ and $\boldsymbol{\omega} = [p \ q \ r]^T$ are the linear and angular velocities, respectively, and $\boldsymbol{S}(a)$ is the skew-symmetric matrix

$$\boldsymbol{S}(a) = -\boldsymbol{S}^T(a) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{3}$$

The attitude can also be represented using Euler angles $\boldsymbol{\Theta} = [\phi \ \theta \ \psi]^T$, where ϕ, θ, ψ are the roll, pitch and yaw angles respectively. Euler angles will be used for plotting purposes in later sections, and also as inputs to the controllers. Algorithms to convert between unit quaternions and Euler angles can be found in [1].

The rotation matrix $\boldsymbol{R}_b^n$ transforms vectors from $\{b\}$ to $\{n\}$ and can be calculated from $\boldsymbol{q}$ using [44]

$$\boldsymbol{R}_b^n(\boldsymbol{q}) = \boldsymbol{I}_{3\times 3} + 2\eta\boldsymbol{S}(\boldsymbol{\epsilon}) + 2\boldsymbol{S}^2(\boldsymbol{\epsilon}) \tag{4}$$

where $\boldsymbol{I}_{3\times 3}$ is the 3 by 3 identity matrix and $\boldsymbol{\epsilon} = [\epsilon_1 \ \epsilon_2 \ \epsilon_3]^T$.

The rates of change of the velocities $\boldsymbol{v}$ and $\boldsymbol{\omega}$ are given by the Newton-Euler equations of motion:

$$m\dot{\boldsymbol{v}} + \boldsymbol{\omega} \times m\boldsymbol{v} = \boldsymbol{R}_b^n(\boldsymbol{q})^T m\boldsymbol{g}^n + \boldsymbol{F}_{prop} + \boldsymbol{F}_{aero} \tag{5}$$

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{M}_{prop} + \boldsymbol{M}_{aero} \tag{6}$$

where $\boldsymbol{g}^n = [0 \ 0 \ g]^T$ and g is the acceleration of gravity. Apart from gravity, the UAV is affected by forces and moments due to aerodynamics and propulsion, which are explained in the next sections. All velocities, forces and moments are represented in the body frame.

A. Aerodynamic Forces and Moments

The UAV is flying in a wind field decomposed into a steady part $\boldsymbol{v}_{w_s}^n$ and a stochastic part $\boldsymbol{v}_{w_g}^b$ representing gusts and turbulence. The steady part is represented in $\{n\}$, while the stochastic part is represented in $\{b\}$. Similarly, rotational disturbances are modeled through the wind angular velocity $\boldsymbol{\omega}_w$. The relative (to the surrounding air mass) velocities of the UAV is then defined as:

$$\boldsymbol{v}_r = \boldsymbol{v} - \boldsymbol{R}_b^n(\boldsymbol{q})^T \boldsymbol{v}_{w_s} - \boldsymbol{v}_{w_g} = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} \tag{7}$$

$$\boldsymbol{\omega}_r = \boldsymbol{\omega} - \boldsymbol{\omega}_w = \begin{bmatrix} p_r \\ q_r \\ r_r \end{bmatrix} \tag{8}$$

From the relative velocity we can calculate the airspeed V_a, angle of attack α and sideslip angle β:

$$V_a = \sqrt{u_r^2 + v_r^2 + w_r^2} \tag{9}$$

$$\alpha = \tan^{-1}\left(\frac{u_r}{w_r}\right) \tag{10}$$

$$\beta = \sin^{-1}\left(\frac{v_r}{V_a}\right) \tag{11}$$

The stochastic components of the wind, given by $\boldsymbol{v}_{w_g} = [u_{w_g} \ v_{w_g} \ w_{w_g}]^T$ and $\boldsymbol{\omega}_w = [p_w \ q_w \ r_w]^T$ are generated by passing white noise through shaping filters given by the Dryden velocity spectra [45][46].

The aerodynamic forces and moments are formulated in terms of aerodynamic coefficients $C_{(*)}$ that are, in general, nonlinear functions of α, β and $\boldsymbol{\omega}_r$, as well as control surface deflections. Aerodynamic coefficients are taken from [27], based on wind tunnel experiments of the Skywalker X8 flying wing as well as a Computational Fluid Dynamics (CFD) code. The X8 is equipped with right and left elevon control surfaces. Note that there is no tail or rudder. Even though the vehicle under consideration has elevons, in [27] the model is parameterized in terms of "virtual" aileron and elevator deflections δ_a and δ_e. These are related to elevon deflections through the transformation

$$\begin{bmatrix} \delta_a \\ \delta_e \end{bmatrix} = \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} \delta_{e,r} \\ \delta_{e,l} \end{bmatrix} \tag{12}$$

where $\delta_{e,r}$ and $\delta_{e,l}$ are right and left elevon deflections, respectively.

The aerodynamic forces are described by

$$\boldsymbol{F}_{aero} = \boldsymbol{R}_w^b(\alpha, \beta) \begin{bmatrix} -D \\ Y \\ -L \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} D \\ Y \\ L \end{bmatrix} = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} C_D(\alpha, \beta, q_r, \delta_e) \\ C_Y(\beta, p_r, r_r, \delta_a) \\ C_L(\alpha, q_r, \delta_e) \end{bmatrix} \tag{14}$$

$$\boldsymbol{M}_{aero} = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} bC_l(\beta, p_r, r_r, \delta_a) \\ cC_m(\alpha, q_r, \delta_e) \\ bC_n(\beta, p_r, r_r, \delta_a) \end{bmatrix} \tag{15}$$

where ρ is the density of air, S is the wing planform area, c is the aerodynamic chord, and b the wingspan of the UAV. The rotation matrix transforming the drag force D, side force Y and lift force L from the wind frame to the body frame is given by:

$$\boldsymbol{R}_w^b(\alpha, \beta) = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta) & -\sin(\alpha) \\ -\sin(\beta) & \cos(\beta) & 0 \\ \cos(\beta)\sin(\alpha) & \sin(\alpha)\sin(\beta) & \cos(\alpha) \end{bmatrix} \tag{16}$$

The model in [27] has similar structure to the linear coefficients in [1], but has added quadratic terms in α and β to the drag coefficient C_D. In addition, C_D is quadratic in the elevator deflection δ_e. In this paper, as an attempt to extend the range of validity of the model, the lift, drag and pitch moment coefficients in [27] are extended using nonlinear Newtonian flat plate theory from [1] and [47]. This makes the lift, drag and pitch coefficients nonlinear in angle of attack by blending between the linear models which are valid for small angles, and the flat plate models which are only valid for large angles. While the linear models are based on physical wind-tunnel experiments and CFD, the nonlinear models have not been validated experimentally.

B. Propulsion Forces and Moments

Assuming the propeller thrust is aligned with the x-axis of $\{b\}$, we can write

$$\boldsymbol{F}_{prop} = \begin{bmatrix} T_p \\ 0 \\ 0 \end{bmatrix} \tag{17}$$

The propeller thrust T_p is given by [48] as presented in [49]:

$$V_d = V_a + \delta_t(k_m - V_a) \tag{18}$$

$$T_p = \frac{1}{2}\rho S_p C_p V_d \left(V_d - V_a\right) \tag{19}$$

where V_d is the discharge velocity of air from the propeller, k_m is a motor constant, S_p is the propeller disc area, C_p is an efficiency factor, and $\delta_t \in [0, 1]$ is the throttle. The parameters in (18) and (19) for a typical X8 motor/propeller configuration are given in [50], which are based on wind tunnel experiments.

The propeller moments are given by

$$\boldsymbol{M}_{prop} = \begin{bmatrix} -k_Q(k_\Omega \delta_t)^2 \\ 0 \\ 0 \end{bmatrix} \tag{20}$$

where $k_\Omega = 797.1268$ and $k_Q = 1.1871\text{e}{-6}$, which are based on the same experimental data used in [50]. Gyroscopic moments are assumed negligible.

C. Actuator Dynamics and Constraints

Denoting commands with superscript c, the elevon control surface dynamics are modeled by rate limited and saturated second-order integrators similar to [51]:

$$\frac{\delta_{e,i}(s)}{\delta_{e,i}^c(s)} = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{21}$$

for $i = r, l$, where $\omega_0 = 100$ and $\zeta = \frac{1}{\sqrt{2}}$. The angular deflections and rates are constrained to ± 30 degrees and ± 200 degrees per second, respectively.

The throttle dynamics are given by the first order transfer function [47]

$$\frac{\delta_t(s)}{\delta_t^c(s)} = \frac{1}{Ts+1} \tag{22}$$

where $T = 0.2$.

IV. METHOD

PPO was the chosen RL algorithm for the attitude controller for several reasons: first, PPO was found to be the best performing algorithm for attitude control of quadcopters in [43], and secondly, PPO's hyperparameters are robust for a large variety of tasks, and it has high performance and low computational complexity. It is therefore the default choice of algorithm in OpenAIs projects.

The objective is to control the UAV's attitude, so a natural choice of controlled variables are the roll, pitch and yaw angles. However, the yaw angle of the aircraft is typically not controlled directly, but through the yaw-rate that depends on the roll angle. In addition, it is desirable to stay close to some nominal airspeed to ensure energy efficient flight, to avoid stall, and to maintain control surface effectiveness which is proportional to airspeed squared. The RL controller is therefore tasked with controlling the roll and pitch angles, ϕ and θ, and the airspeed V_a to desired reference values. At each time step the controller receives an immediate reward, and it aims at developing a control law that maximizes the sum of future discounted rewards.

The action space of the controller is three dimensional, consisting of commanded virtual elevator and aileron angles as well as the throttle. Elevator and aileron commands are mapped to commanded elevon deflections using the inverse of the transformation given by (12).

The observation vector (i.e. the input to the RL algorithm) contains information obtained directly from state feedback of states typically measured by standard sensor suites. No sensor noise is added. To promote smooth actions it also includes a moving average of previous actuator setpoints. Moreover, since the policy network is a feed-forward network with no memory, the observation vector at each time step consists of these values for several previous time steps to facilitate learning of the dynamics.

A. The Proximal Policy Optimization Algorithm

PPO is a model-free, on-policy, actor-critic, policy-gradient method. It aims to retain the reliable performance of TRPO algorithms, which guarantee monotonic improvements by considering the Kullback–Leibler (KL) divergence of policy updates, while only using first-order optimization. In this section, π is the policy network (that is, the control law) which is optimized wrt. its parameterization θ,[1] in this case the NN weights. The policy network takes the state, s, as its input, i.e. the observation vector, and outputs an action, a, consisting of the elevator, aileron and throttle setpoints. For continuous action spaces, the policy network is tasked with outputting the moments of a probability distribution, in this case the means and variances of a multivariate Gaussian, from which actions are drawn. During training, actions are randomly sampled from this distribution to increase exploration, while the mean is taken as the action when training is completed.

Policy gradient algorithms work by estimating the policy gradient, and then applying a gradient ascent algorithm to the gradient estimate. The gradients are estimated in a Monte Carlo (MC) fashion by running the policy in the environment to obtain samples of the policy loss $J(\theta)$ and its gradient [19]:[2]

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_t R(s_t, a_t)\right] = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[R(\tau)] \tag{23}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right) R(\tau)\right] \tag{24}$$

In practice, these gradients are obtained with automatic differentiation software on a surrogate loss objective, whose gradients are the same as (24), and are then backpropagated through the NN to update θ.

The central challenge in policy gradient methods lie in reducing the variance of the gradient estimates, such that consistent progress towards better policies can be made. The actor-critic architecture makes a significant impact in this regard, by reformulating the reward signals in terms of advantage:

$$Q^\pi(s,a) = \sum_t \mathbb{E}_{\pi_\theta}[R(s_t,a_t) \mid s, a] \tag{25}$$

$$V^\pi(s) = \sum_t \mathbb{E}_{\pi_\theta}[R(s_t,a_t) \mid s] \tag{26}$$

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{27}$$

The advantage function (27) measures how good an action is compared to the other actions available in the state, such that good actions have positive rewards, and bad actions have negative rewards. One thus has to be able to estimate the average reward of the state, i.e. the value function $V(s)$.[3] This is the job of the critic network, a separate NN trained in a supervised manner to predict the value function with ground truth from the reward values in the gathered samples. Several improvements such as generalized advantage estimate (GAE) are further employed to reduce variance of the advantage estimates. PPO also makes use of several actors simultaneously gathering samples with the policy, to increase the sample batch size.

PPO maximizes the surrogate objective function

[1] θ is used in this section as it is the established nomenclature in the machine learning field, but will in the rest of the article refer to the pitch angle.

[2] τ represents trajectories of the form $(s_1, a_1, s_2, a_2, \ldots, s_T, a_T)$

[3] The value function $V^\pi(s)$ is the expected long term reward of being in state s and then following policy π, as opposed to the $Q^\pi(s,a)$-function which focuses on the long term reward of taking a specific action in the state, and then following the policy.

$$L(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{28}$$

in which $\hat{A}$ and $\hat{\mathbb{E}}$ denotes the empirically obtained estimates of the advantage function and expectation, respectively, and $r_t(\theta)$ is the probability ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)} \tag{29}$$

Vanilla policy gradients require samples from the policy being optimized, which after a single optimization step are no longer usable for the improved policy. For increased sample efficiency, PPO uses importance sampling to obtain the expectation of samples gathered from an old policy $\pi_{\theta_{old}}$ under the new policy we want to refine π_θ. In this way, each sample can be used for several gradient ascent steps. As the new policy is refined, the two policies will diverge, increasing variance of the estimation, and the old policy is therefore periodically updated to match the new policy. For this approach to be valid, the state transition function must be similar between the two policies, which is ensured by clipping the probability ratio (29) to the region $[1-\epsilon,\ 1+\epsilon]$.[4] This also gives a first-order approach to trust region optimization: The algorithm is not too greedy in favoring actions with positive advantage, and not too quick to avoid actions with a negative advantage function from a small set of samples. The minimum operator ensures that the surrogate objective function is a lower bound on the unclipped objective, and eliminates increased preference for actions with negative advantage function. PPO is outlined in Algorithm 1.

Algorithm 1: PPO

```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy π_θold in environment for T time steps
        Compute advantage estimates Â_t for
          t = 1, 2, ..., T
    end
    Optimize surrogate L wrt. θ.
    θ_old ← θ
end
```

B. Action Space

A known issue in optimal control is that while continually switching between maximum and minimum input is often optimal in the sense of maximizing the objective function, in practice it wears unnecessarily on the actuators. Since PPO during training samples its outputs from a Gaussian distribution, a high variance will generate highly fluctuating actions. This is not much of a problem in a simulator environment but could be an issue if trained online on a real aircraft. PPO's hyperparameters are tuned wrt. a symmetric action space with a small range (e.g. -1 to 1). Adhering to this design also has the benefit of increased generality, training the controller to output actions as a fraction of maximal and minimal setpoints. The actions produced by the controller are therefore clipped to this range, and subsequently scaled to fit the actuator ranges as described in Section III.

[4]The clip operator saturates the variable in the first argument between the values supplied by the two following arguments.

TABLE I: Constraints and ranges for initial conditions and target setpoints used during training of controller.

Variable	Initial Condition	Target
ϕ	$\pm 150°$	$\pm 60°$
θ	$\pm 45°$	$\pm 30°$
ψ	$\pm 60°$	-
ω	$\pm 60\,°/\text{s}$	-
α	$\pm 26°$	-
β	$\pm 26°$	-
V_a	$12 - 30\,\text{m/s}$	$12 - 30\,\text{m/s}$

C. Training of Controller

The PPO RL controller was initialized with the default hyperparameters in the OpenAI Baselines implementation [52], and ran with 6 parallel actors. The controller policy is an extended version of the default two hidden layer, 64 nodes multi layer perceptron (MLP) policy: The observation vector is first processed in a convolutional layer with three filters spanning the time dimension for each component, before being fed to the default policy. This allows the policy to construct functions on the time evolution of the observation vector, while scaling more favorably in parameter count with increasing observation vector size compared to a fully connected input layer.

The controller is trained in an episodic manner to assume control of an aircraft in motion and orient it towards some new reference attitude. Although the task at hand is not truly episodic in the sense of having natural terminal states, episodic training allows one to adjust episode conditions to suit the agents proficiency, and also admits greater control of the agents exploration of the state space. The initial state and reference setpoints for the aircraft are randomized in the ranges shown in Table I. Episode conditions are progressively made more difficult as the controller improves, beginning close to target setpoints and in stable conditions, until finally spanning the entirety of Table I. The chosen ranges allow the RL controller to demonstrate that it is capable of attitude control, and facilitates comparison with the PID controller as it is expected to perform well in this region. According to [1], a typical sampling frequency for autopilots is 100 Hertz, and the simulator therefore advances 0.01 seconds at each time step. Each episode is terminated after a maximum of 2000 time steps, corresponding to 20 seconds of flight time. No wind or turbulence forces are enabled during training of the controller.

In accordance with traditional control theory, where one usually considers cost to be minimized rather than rewards to be maximized, the immediate reward returns to the RL

controller are all negative rewards in the normalized range of -1 to 0:

$$
\begin{aligned}
R_\phi &= \text{clip}\left(\frac{|\phi-\phi^d|}{\zeta_1}, 0, \gamma_1\right)\\
R_\theta &= \text{clip}\left(\frac{|(\theta-\theta^d)|}{\zeta_2}, 0, \gamma_2\right)\\
R_{V_a} &= \text{clip}\left(\frac{|V_a - V_a^d|}{\zeta_3}, 0, \gamma_3\right)\\
R_{\delta^c} &= \text{clip}\left(\frac{\sum_{j\in[a,e,t]}\sum_{i=0}^{4}|\delta^c_{j_{t-i}} - \delta^c_{j_{t-1-i}}|}{\zeta_4}, 0, \gamma_4\right)\\
R_t &= -(R_\phi + R_\theta + R_{V_a} + R_{\delta^c}) \qquad (30)
\end{aligned}
$$

$$
\begin{aligned}
\zeta_1 = 3.3,\ \zeta_2 = 2.25,\ \zeta_3 = 25,\ \zeta_4 = 60\\
\gamma_1 = 0.3,\ \gamma_2 = 0.3,\ \ \gamma_3 = 0.3, \gamma_4 = 0.1
\end{aligned}
$$

In this reward function, L_1 was chosen as the distance metric between the current state and the desired state (denoted with superscript d).[5] Furthermore, a cost is attached to changing the actuator setpoints to address oscillatory control behaviour. Commanded control setpoint of actuator j at time step t is denoted $\delta^c_{j_t}$, where $j \in [a, e, t]$. The importance of each component of the reward function is weighted through the γ factors. To balance the disparate scales of the different components, the values are divided by the variables approximate dynamic range, represented by the ζ factors.

The components of the observation vector are expressed in different units and also have differing dynamic ranges. NNs are known to converge faster when the input features share a common scale, such that the network does not need to learn this scaling itself. The observation vector should therefore be normalized. This is accomplished with the VecNormalize class of [52], which estimates a running mean and variance of each observation component and normalizes based on these estimates.

D. Evaluation

Representing the state-of-the-art in model free control, fixed-gain PID controllers for roll, pitch and airspeed were implemented to provide a baseline comparison for the RL controller:

$$\delta^c_t = -k_{p_V}(V_a - V_a^d) - k_{i_V}\int_0^t (V_a - V_a^d)d\tau \qquad (31)$$

$$\delta^c_a = -k_{p_\phi}(\phi - \phi^d) - k_{i_\phi}\int_0^t (\phi - \phi^d)d\tau - k_{d_\phi}p \qquad (32)$$

$$\delta^c_e = -k_{p_\theta}(\theta - \theta^d) - k_{i_\theta}\int_0^t (\theta - \theta^d)d\tau - k_{d_\theta}q \qquad (33)$$

The throttle is used to control airspeed, while virtual aileron and elevator commands are calculated to control roll and pitch, respectively. The PID controllers were manually tuned using a trial-and-error approach until achieving acceptable transient responses and low steady-state errors for a range of initial conditions and setpoints. Wind was turned off in the simulator during tuning. The integral terms in (31)-(33) are implemented numerically using forward Euler. Controller gains are given in Table II.

TABLE II: PID controller parameters.

Parameter	Value	Parameter	Value
k_{p_V}	0.5	k_{d_ϕ}	0.5
k_{i_V}	0.1	k_{p_θ}	-4
k_{p_ϕ}	1	k_{i_θ}	-0.75
k_{i_ϕ}	0	k_{d_θ}	-0.1

The same aerodynamic model that is used for training is also used for evaluation purposes, with the addition of disturbances in the form of wind to test generalization capabilities. The controllers are compared in four distinct wind and turbulence regions: light, moderate, severe and no turbulence. Each setting consists of a steady wind component, with randomized orientation and a magnitude of 7 m/s, 15 m/s, 23 m/s and 0 m/s respectively, and additive turbulence given by the Dryden turbulence model [45]. Note that a wind speed of 23 m/s is a substantial disturbance, especially when considering the Skywalker X8's nominal airspeed of 18 m/s. For each wind setting, 100 sets of initial conditions and target setpoints are generated, spanning the ranges shown in Table I. The reference setpoints are set to 20-30 degrees and 3-4 m/s deviation from the initial state for the angle variables and airspeed, respectively. Each evaluation scenario is run for a maximum of 1500 time steps, corresponding to 15 seconds of flight time, which should be sufficient time to allow the controller to regulate to the setpoint.

The reward function is not merely measuring the proficiency of the RL controller, but is also designed to facilitate learning. To compare, rank and evaluate different controllers, one needs to define additional evaluation criteria. To this end, the controllers are evaluated on the following criteria: **Success/failure**, whether the controller is successful in controlling the state to within some bound of the setpoint. The state must remain within the bounds for at least 100 consecutive time steps to be counted as a success. The bound was chosen to be $\pm 5^\circ$ for the roll and pitch angles, and $\pm$2m/s for the airspeed. **Rise time**, the time it takes the controller to reduce the initial error from 90 % to 10 %. As these control scenarios are not just simple step responses and may cross these thresholds several times during the episode, the rise time is calculated from the first time it crosses the lower threshold until the first time it reaches the upper threshold. **Settling time**, the time it takes the controller to settle within the success setpoint bounds, and never leave this bound again. **Overshoot**, the peak value reached on the opposing side of the setpoint wrt. the initial error, expressed as a percentage of the initial error. **Control variation**, the average change in actuator commands per second, where the average is taken over time steps and actuators. Rise time, settling time, overshoot and control variation are only measured

[5]The L_1 distance has the advantage of punishing small errors harsher than the L_2 distance, and therefore encourages eliminating small steady-state errors.

when the episode is counted as a success. When comparing controllers, the success criterion is the most important, as it is indicative of stability as well as achieving the control objective. Secondly, low control variation is important to avoid unnecessary wear and tear on the actuators. While success or failure is a binary variable, rise time, settling time and overshoot give additional quantitative information on the average performance of the successful scenarios.

V. RESULTS AND DISCUSSION

The controller was trained on a desktop computer with an i7-9700k CPU and an RTX 2070 GPU. The model converges after around two million time steps of training, which on this hardware takes about an hour. This is relatively little compared to other applications of DRL, and suggests that the RL controller has additional capacity to master more difficult tasks. Inference with the trained model takes 800 microseconds on this hardware, meaning that the RL controller could reasonably be expected to be able to operate at the assumed autopilot sampling frequency of 100 Hertz in flight.

A. Key Factors Impacting Training

The choice of observation vector supplied to the RL controller proved to be significant for its rate of improvement during training and its final performance. It was found that reducing the observation vector to only the essential components, i.e. the current airspeed and roll and pitch angles, the current angular velocities of the UAV, and the state errors, helped the RL controller improve significantly faster than other, larger observation vectors.[6] Including values for several previous time steps (five was found to a good choice) further accelerated training, as this makes learning the dynamics easier for the memoryless feed-forward policy.

The reward function is one of the major ways the designer can influence and direct the behaviour of the agent. One of the more popular alternatives to L_1 norm and clipping to achieve saturated rewards are the class of exponential reward functions, and notably the Gaussian reward function as in [37]. Analyzing different choices of the reward function was not given much focus as the original choice gave satisfying results.

B. Evaluation of Controller

The RL controller generalizes well to situations and tasks not encountered during training. Even though the controller is trained with a single setpoint for each episode, Figure 2 shows that the controller is perfectly capable of adapting to new setpoints during flight. This result was also found by Koch et al. [43] for quadcopters. The generalization capability also holds true for unmodeled wind and turbulence forces. The controller is trained with no wind estimates present in the observation vector, and no wind forces enabled in the simulator, but as Table III shows it is still able to achieve tracking to the setpoint when steady wind and turbulence is enabled in the test environment. Table III should be read as a quantitative analysis of performance in conditions similar to normal operating conditions, while Figure 2 and 3 qualitatively shows the capabilities of the controllers on significantly more challenging tasks.

Table III shows that both controllers are generally capable of achieving convergence to the target for the evaluation tasks, with neither controller clearly outperforming the other. The RL controller has an advantage over the PID controller on the success criterion, and seems to be more robust to the turbulence disturbance. It is able to achieve convergence in the attitude states in all situations, unlike the PID controller, and is also notably more successful in moderate and severe turbulence conditions. The PID controller has considerably lower control variation for the simple settings with little or no wind, but its control variation grows fast with increasing disturbance. At severe turbulence the RL controller has the least control variation.

The two controllers perform similarly wrt. settling time and rise time, each having the edge in different states under various conditions, while the PID controller performs favorably when measured on overshoot. All in all, this is an encouraging result for the RL controller, as it is able to perform similarly as the established PID controller in its preferred domain, while the RL controller is expected to make its greatest contribution in the more nonlinear regions of the state space.

A comparison of the two controllers is shown in Figure 3 on a scenario involving fairly aggressive maneuvers, which both are able to execute. Figure 2 and 3 illustrate an interesting result, the RL controller is able to eliminate steady state errors. While the PID controller has integral action to mitigate steady-state errors, the control law of the RL controller is only a function of the last few states and references. This might suggest that the RL controller has learned some feed-forward action, including nominal inputs in each equilibrium state, thus removing steady-state errors in most cases. Another possibility is that steady-state errors are greatly reduced through the use of high-gain feedback, but the low control variation shown for severe turbulence in Table III indicates that the gain is not excessive. Future work should include integral error states in the observations and evaluate the implications on training and flight performance.

VI. CONCLUSIONS

The ease with which the proof of concept RL controller learns to control the UAV for the tasks presented in this paper, and its ability to generalize to turbulent wind conditions, suggests that DRL is a good candidate for nonlinear flight control design. A central unanswered question here is the severity of the reality gap, or in other words how transferable the strategies learned in simulations are to real world flight. Future work should evaluate the controller's robustness to parametric and structural aerodynamic uncertainties; this is essential to do before undertaking any real life flight experiments. For more advanced maneuvers, e.g. aerobatic flight

[6]Essential here referring to the factors' impact on performance for this specific control task. One would for instance expect α and β to be essential factors when operating in the more extreme and nonlinear regions of the state space.

TABLE III: Performance metrics for the RL controller and the baseline PID controller on the evaluation scenarios. Both controllers exhibit strengths in different aspects — the best value in each circumstance is shown in bold.

Setting	Controller	Success (%)				Rise time (s)			Settling time (s)			Overshoot (%)			Control variation (s^{-1})
		ϕ	θ	V_a	All	ϕ	θ	V_a	ϕ	θ	V_a	ϕ	θ	V_a	
No turbulence	RL	100	100	**100**	**100**	**0.265**	0.661	**0.825**	**1.584**	1.663	2.798	21	24	**31**	0.517
	PID	100	100	98	98	1.344	**0.228**	0.962	2.050	**1.364**	**2.198**	**4**	**17**	35	**0.199**
Light turbulence	RL	100	100	**100**	**100**	**0.210**	0.773	**0.744**	**1.676**	1.806	2.738	28	33	**36**	0.748
	PID	100	100	99	99	1.081	**0.294**	0.863	2.057	**1.638**	**2.369**	**6**	**20**	43	**0.457**
Moderate turbulence	RL	**100**	**100**	**98**	**98**	**0.192**	1.474	0.934	**2.167**	**2.438**	4.085	54	54	74	0.913
	PID	100	93	90	87	0.793	**0.525**	**0.864**	2.764	2.563	**3.460**	**34**	**35**	**70**	**0.781**
Severe turbulence	RL	**100**	**100**	**92**	**92**	**0.166**	1.792	1.585	**2.903**	**3.280**	7.049	122	93	156	**1.083**
	PID	99	96	87	86	0.630	**0.945**	**1.343**	3.576	5.256	**5.470**	**92**	**80**	**122**	1.117

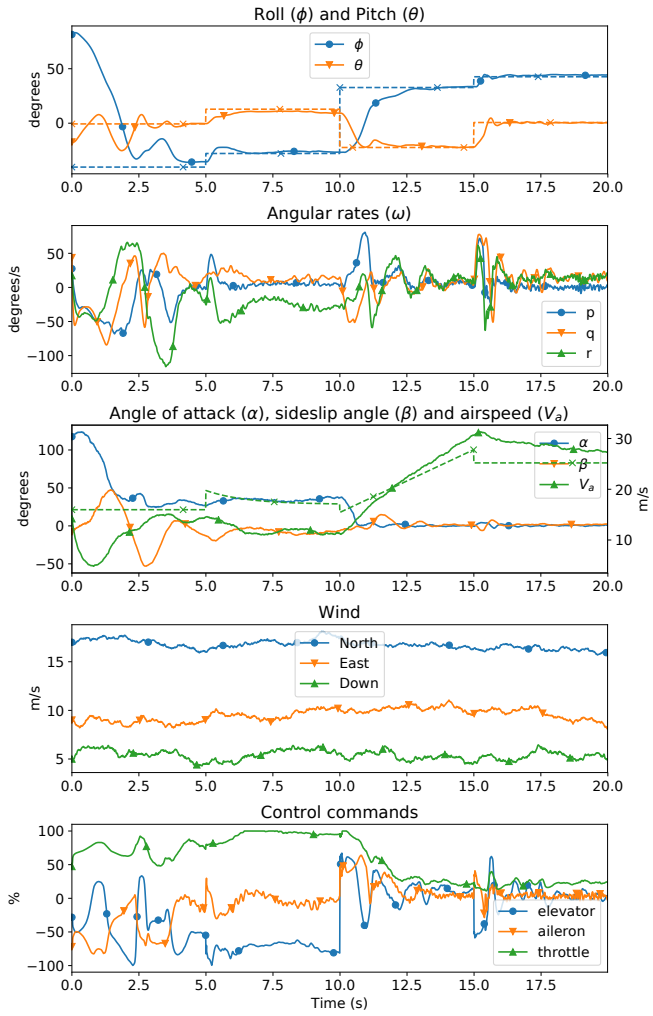

Fig. 2: The RL controller trained episodically with a single setpoint and no wind or turbulence generalizes well to many wind conditions and continuous tracking of setpoints (shown with dashed lines marked by crosses). Here subjected to severe wind and turbulence disturbances with a magnitude of 20 m/s.

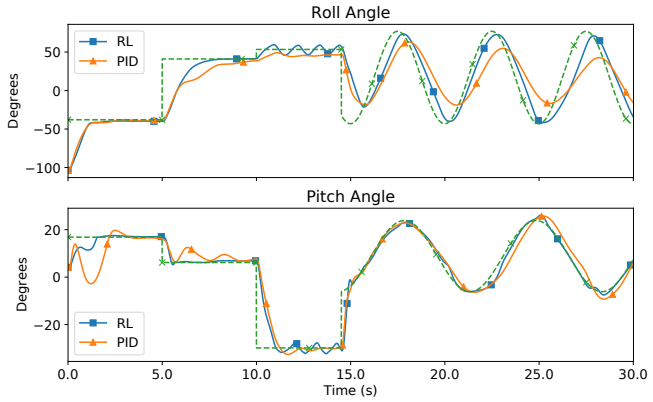

Fig. 3: Comparison of the PID and RL controllers tasked with tracking the dashed green line.

or recovering from extreme situations, the controller should be given more freedom in adjusting the airspeed, possibly through having it as an uncontrolled state.

There is still much potential left to harness for this class of controllers. The policy network used to represent the control law is small and simple; more complex architectures such as long short term memory (LSTM) could be used to make a dynamic RL controller. Training, experiments and reward structures can be designed to facilitate learning of more advanced behavior, tighter control or better robustness. Should the reality gap prove to be a major obstacle for the success of the RL controller in the real world, one should look to the class of off-policy algorithms such as SAC. These algorithms are able to learn offline from gathered data, and thus might be more suited for UAV applications.

Acknowledgments

The first author is financed by "PhD Scholarships at SINTEF" from the Research Council of Norway (grant no. 272402). The second and fourth authors were partially supported by the Research Council of Norway at the Norwegian University of Science and Technology (grants no. 223254 NTNU AMOS and no. 261791 AutoFly).

References

[1] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft : Theory and Practice*. Princeton University Press, 2012.

[2] H. K. Khalil, *Nonlinear Systems (3rd Edition)*. Pearson, 2001.

[3] C. V. Girish, F. Emilio, H. P. Jonathan, and L. Hugh, "Nonlinear flight control techniques for unmanned aerial vehicles," in *Handbook of Unmanned Aerial Vehicles*. Springer Netherlands, aug 2014, pp. 577–612.

[4] D. Rotondo, A. Cristofaro, K. Gryte, and T. A. Johansen, "LPV model reference control for fixed-wing UAVs," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 559–11 564, jul 2017.

[5] Y. Kawakami and K. Uchiyama, "Nonlinear controller design for transition flight of a fixed-wing UAV with input constraints," in *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, jan 2017.

[6] W. Ren and E. Atkins, "Nonlinear trajectory tracking for fixed wing UAVs via backstepping and parameter adaptation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, aug 2005.

[7] H. Castañeda, O. S. Salas-Peña, and J. de León-Morales, "Extended observer based on adaptive second order sliding mode control for a fixed wing UAV," *ISA Transactions*, vol. 66, pp. 226–232, jan 2017.

[8] S. H. Mathisen, K. Gryte, T. Johansen, and T. I. Fossen, "Non-linear model predictive control for longitudinal and lateral guidance of a small fixed-wing UAV in precision deep stall landing," in *AIAA Infotech @ Aerospace*. American Institute of Aeronautics and Astronautics, jan 2016.

[9] G. A. Garcia, S. Kashmiri, and D. Shukla, "Nonlinear control based on h-infinity theory for autonomous aerial vehicle," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2017.

[10] M. G. Michailidis, K. Kanistras, M. Agha, M. J. Rutherford, and K. P. Valavanis, "Robust nonlinear control of the longitudinal flight dynamics of a circulation control fixed wing UAV," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, dec 2017.

[11] E. Lavretsky and K. A. Wise, *Robust and Adaptive Control With Aerospace Applications*. Springer London, 2012.

[12] I. Kaminer, A. Pascoal, E. Xarga, N. Hovakimyan, C. Cao, and V. Dobrokhodov, "Path Following for Small Unmanned Aerial Vehicles Using L1 Adaptive Augmentation of Commercial Autopilots," *Journal of Guidance, Control, and Dynamics*, vol. 33, pp. 550–564, 2010.

[13] J. M. Levin, A. A. Paranjape, and M. Nahon, "Agile maneuvering with a small fixed-wing unmanned aerial vehicle," *Robotics and Autonomous Systems*, vol. 116, pp. 148–161, jun 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018304305

[14] E. Bulka and M. Nahon, "Automatic Control for Aerobatic Maneuvering of Agile Fixed-Wing UAVs," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1-2, pp. 85–100, feb 2019. [Online]. Available: http://link.springer.com/10.1007/s10846-018-0790-z

[15] S. Kurnaz, O. Cetin, and O. Kaynak, "Fuzzy logic based approach to design of flight control and navigation tasks for autonomous unmanned aerial vehicles," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1-3, pp. 229–244, oct 2008.

[16] T. Lee and Y. Kim, "Nonlinear adaptive flight control using backstepping and neural networks controller," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 4, pp. 675–682, jul 2001.

[17] H. A. de Oliveira and P. F. F. Rosa, "Genetic neuro-fuzzy approach for unmanned fixed wing attitude control," in *2017 International Conference on Military Technologies (ICMT)*. IEEE, may 2017.

[18] F. Santoso, M. A. Garratt, and S. G. Anavatti, "State-of-the-art intelligent flight control systems in unmanned aerial vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 613–627, apr 2018.

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

[20] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. I. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *CoRR*, vol. abs/1511.03791, 2015. [Online]. Available: http://arxiv.org/abs/1511.03791

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971 [cs, stat]*, Sep. 2015, arXiv: 1509.02971.

[23] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," *arXiv:1502.05477 [cs]*, Feb. 2015, arXiv: 1502.05477.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs]*, Jul. 2017, arXiv: 1707.06347.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *arXiv:1801.01290 [cs, stat]*, Jan. 2018, arXiv: 1801.01290. [Online]. Available: http://arxiv.org/abs/1801.01290

[26] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv:1606.01540 [cs]*, Jun. 2016, arXiv: 1606.01540.

[27] K. Gryte, R. Hann, M. Alam, J. Roháč, T. A. Johansen, and T. I. Fossen, "Aerodynamic modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018.

[28] E. Bøhn, "Pyfly," https://github.com/eivindeb/pyfly, 2019.

[29] ——, "Fixed-wing aircraft gym environment," https://github.com/eivindeb/fixed-wing-gym, 2019.

[30] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3948–3955.

[31] J.-H. Han, D.-K. Lee, J.-S. Lee, and S.-J. Chung, "Teaching micro air vehicles how to fly as we teach babies how to walk," *Journal of Intelligent Material Systems and Structures*, vol. 24, no. 8, pp. 936–944, 2013. [Online]. Available: https://doi.org/10.1177/1045389X13478270

[32] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, "Autonomous navigation of UAV by using real-time model-based reinforcement learning," in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, nov 2016, pp. 1–6.

[33] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," *CoRR*, vol. abs/1509.06791, 2015. [Online]. Available: http://arxiv.org/abs/1509.06791

[34] S.-M. Hung and S. N. Givigi, "A Q-Learning Approach to Flocking With UAVs in a Stochastic Environment," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 186–197, jan 2017.

[35] R. Polvara, M. Patacchiola, S. Sharma, J. Wan, A. Manning, R. Sutton, and A. Cangelosi, "Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2018, pp. 115–123.

[36] K. Xu, F. Wu, and J. Zhao, "Model-based deep reinforcement learning with heuristic search for satellite attitude control," *Industrial Robot: An International Journal*, pp. IR–05–2018–0086, oct 2018.

[37] I. Carlucho, M. De Paula, S. Wang, Y. Petillot, and G. G. Acosta, "Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 107, pp. 71–86, sep 2018.

[38] J. A. Bagnell and J. G. Schneider, "Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods," in *2001 IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[39] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *International Symposium on Experimental Robotics*, 2004.

[40] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 1–8.

[41] R. Cory and R. Tedrake, "Experiments in Fixed-Wing UAV Perching," in *AIAA Guidance, Navigation and Control Conference and Exhibit*. Reston, Virigina: American Institute of Aeronautics and Astronautics, aug 2008. [Online]. Available: http://arc.aiaa.org/doi/10.2514/6.2008-7256

[42] H. Bou-Ammar, H. Voos, and W. Ertel, "Controller design for quadrotor UAVs using reinforcement learning," in *2010 IEEE International Conference on Control Applications*. IEEE, sep 2010, pp. 2130–2135.

[43] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for UAV attitude control," *CoRR*, vol. abs/1804.04154, 2018. [Online]. Available: http://arxiv.org/abs/1804.04154

[44] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. Wiley, 2011.

[45] U.S. Department of Defense, "U.S. Military Specification MIL-F-8785C," *Washington, D.C.: U.S. Department of Defense*, 1980.

[46] MathWorks, "Dryden wind turbulence model (continuous)," https://se.mathworks.com/help/aeroblks/drydenwindturbulencemodelcontinuous.html, accessed: 2019-02-18.

[47] K. Gryte, "High Angle of Attack Landing of an Unmanned Aerial Vehicle," Master's thesis, Norwegian University of Science and Technology, 2015. [Online]. Available: https://brage.bibsys.no/xmlui/handle/11250/2352405

[48] P. Fitzpatrick, "Calculation of thrust in a ducted fan assembly for hovercraft," Hovercraft Club of Great Britain, Tech. Rep., 2003.

[49] R. W. Beard, "UAVBOOK Supplement. Additional thoughts on propeller thrust model," http://uavbook.byu.edu/, Princeton University Press, Tech. Rep., 2014.

[50] E. M. Coates, A. Wenz, K. Gryte, and T. A. Johansen, "Propulsion System Modeling for Small Fixed-Wing UAVs," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[51] B. prasad B. and S. Pradeep, "Automatic landing system design using feedback linearization method," in *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, 2007.

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