Volume 65, Issue 3, 2021

Journal of Scientific Research

Institute of Science,

Banaras Hindu University, Varanasi, India.

# Improving Energy Efficiency in UAV Attitude Control using Deep Reinforcement Learning

Varun Agarwal*[1], Rajiv Ranjan Tewari[2]

*[1]Centre of Computer Education, IPS, University of Allahabad, Prayagraj, India, varunag18@gmail.com
[2]Department of Electronics and Communication, University of Allahabad, Prayagraj, India, tewari.rr@gmail.com

***Abstract*—Attitude control refers to controlling the rotational motion of an Unmanned Aerial Vehicle (UAV) about its axes. Any movement requires energy consumption and UAV batteries store limited energy. We propose the use of reinforcement learning to optimise energy usage in UAVs. We use Proximal Policy Optimization (PPO) algorithm to train the model, modifying the existing algorithm to incorporate sigmoid activation function. We have introduced Ornstein Uhlenbeck noise to the policy function with the intention of adding the unpredictability found in real world environments. We have designed the reward function of our algorithm such that the quadcopter aims to change its existing angular velocity to achieve the target angular velocity. While this attitude control takes place, energy is spent due to motor/propeller revolutions. Our reward function minimizes the rapid change in motor speeds which causes fast battery depletion, thus saving energy and enhancing UAV flight time.**

***Index Terms*—attitude control, unmanned aerial vehicles, reinforcement learning, quadcopters, energy efficiency, proximal policy optimization**

## I. Introduction

Unmanned Aerial Vehicles (UAVs), also known as drones, are getting popular day by day. Till some time back, UAVs used to be a field attracting man's imagination and in the hands of the chosen few, who used it for hobby or passion. Continuous research in the academia and the industry has given great impetus to this device's use in a plethora of tasks. Today, we find UAVs being used in surveillance, photography, deliveries, disaster relief, exploration of mineral ores, building construction, recreational uses, etc. UAVs can be broadly classified into quadrotors or quadcopters and fixed wing. Quadcopters have a number of advantages like hovering capability, no take-off and landing space required, etc. However, they have a lower flight duration capability as compared to fixed wing UAVs owing to lesser battery carrying capacities. Thus, there is a need to optimally utilize their available energy. Since, the utility of quadcopters is known to be much greater than that of fixed wing UAVs, our study will also be focussed on quadcopters. A UAV has six degrees of freedom, three rotational (which define the three axis about which a UAV can rotate) and three translational (defining the motion of the UAV in the three dimensional space from one point to another). The UAV's movement is the most important source of its power consumption. The UAV movement can be either dictated by a human operator or it can be preprogrammed. There is another option where we use the power of artificial intelligence to allow the UAV decide on its own, its movements. This is particularly lucrative given that it is not always possible to consider the impact of external factors while pre-programming the UAV's path and orientation. Also, having a human operator is an expensive option.
Reinforcement learning, a branch of artificial intelligence, allows objects to learn from experience. There is ongoing research dedicated to finding areas where reinforcement learning can be used. We propose the application of reinforcement learning to minimize the energy consumption in the attitude control of quadcopter UAVs.

### A. What is attitude control in UAVs

Attitude is the angle at which the UAV flies, relative to the ground. It defines the UAVs orientation to the horizontal plane. The UAV has three axes about which it can rotate - roll, pitch and yaw. Roll is the UAV's rotation about the axis passing through its longitude. This axis runs from the head to tail of the UAV. Pitch implies the UAV's rotation about the lateral axis. It leads to the nose up and down movement. The yaw rotation is the UAV's rotation clockwise or anticlockwise while it remains level to the ground (Refer to Fig. 1). Attitude control refers to controlling this rotational motion by regulating the rotations about the roll, pitch, yaw axes.
Four armed quadcopters may have a + orientation or an X orientation Fig. 2. However, the X orientation has been found to be more useful as the quadcopter arm does not come in the way of the camera. For our study, we shall be taking up the X oriented quadcopter UAV. A UAV's rotational motion is governed by the difference in rotational speed of its motors.

*DOI:* 10.37398/JSR.2021.650325

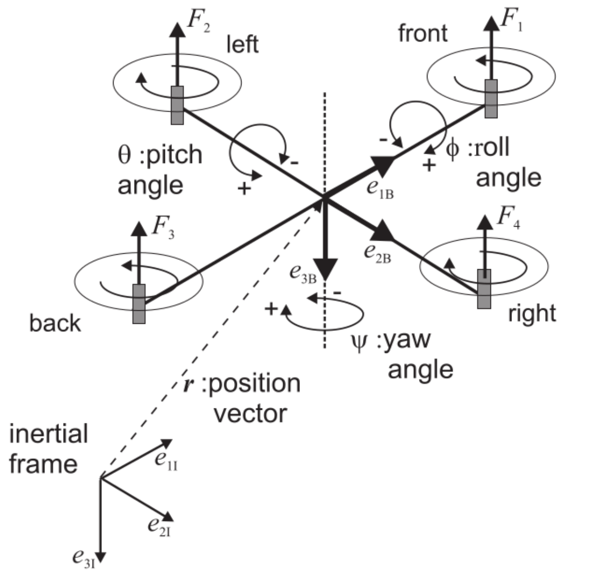

Fig. 1. Roll, Pitch, Yaw in a Quadcopter (Bou-Ammar, Voos, & Ertel, 2010, pp. 2130-2135)

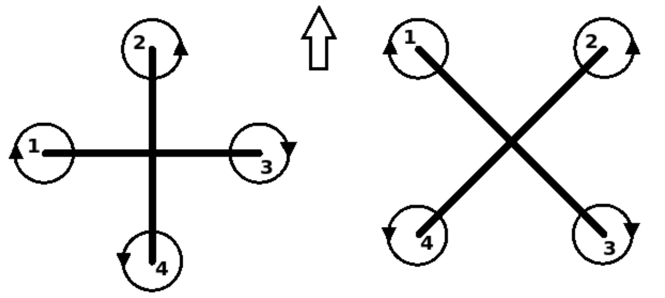

Fig. 2. Quadcopter orientations - Left (+ orientation) and Right (X orientation). Arrow indicates direction of flight

Performing attitude changes simply requires increasing or decreasing the speed of rotations of the motors in a set pattern, which are as follows:
*Pitch Down (Forward flight):* Increase speed of motors 3 and 4 while reduce speed of motors 1 and 2. Fig. 3(a)
*Pitch Up (Backward flight):* Increase speed of motors 1 and 2 while reduce speed of motors 3 and 4. Fig. 3(b)
*Roll Left (Turn left flight):* Increase speed of motors 2 and 3 while reduce speed of motors 1 and 4. Fig. 3(c)
*Roll Right (Turn right flight):* Increase speed of motors 1 and 4 while reduce speed of motors 2 and 3. Fig. 3(d)
*Yaw Left (Turn left):* Increase speed of motors 2 and 4 while reduce speed of motors 1 and 3. Fig. 3(e)
*Yaw Right (Turn right):* Increase speed of motors 1 and 3 while reduce speed of motors 2 and 4. Fig. 3(f)

A quadcopter mainly consists of a flight controller, battery, radio receiver and a set of Electronic Speed Controllers (ESCs), motors and propellers for each of its four arms. The flight controller includes an Inertial Measurement Unit (IMU), GPS, etc. The IMU includes the Gyroscope, Accelerometer,

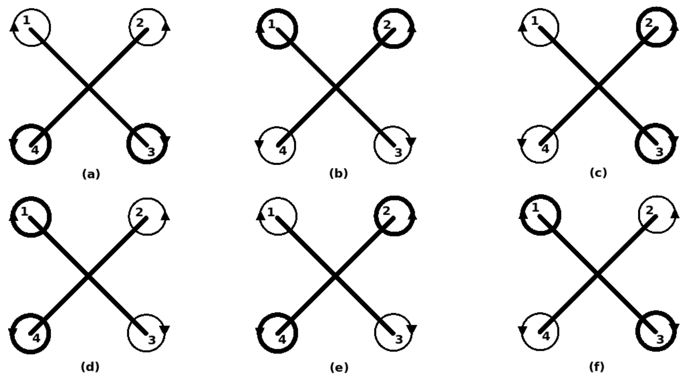

Fig. 3. Changing the speed of motor rotates the quadcopter about the desired axis. (Dark circles indicate increased speed of motor)

Magnetometer, etc. Our study will be focussing on the Gyroscope only as it detects the rotational attributes roll, pitch and yaw. The Electronic Speed Controllers (ESCs) are used to control the UAV's attitude by varying the speed of the propellers.

## B. Energy efficiency in UAV attitude control

The only moving objects in a quadcopter are the motors and propellers. The Electronic Speed Controllers (ESCs) pass instructions received from the flight controller to drive the motors which make the propellers work. Thus, in order to make the UAV's attitude control energy efficient, we need to manage the instructions passed from the ESCs to their respective motors. If the motors are driven at high revolutions per minute or RPMs, the battery drainage would be fast. Also, if there are high variations in the RPM values, energy gets drawn from the batteries faster leading to faster drainage. The ultimate aim is to increase UAV flying time, which can be best obtained if the rate of battery depletion is reduced. Since, the movable parts of the UAV are its major energy guzzlers, we need to find strategies to minimize their energy consumption. Hence, we propose to use reinforcement learning as the technique to improve energy efficiency in UAV's attitude control.

## C. Deep Reinforcement learning

Reinforcement learning is a branch where we have a problem and we have a number of possible solutions or ways to handle that problem (Sutton & Barto, 2018). The purpose is to select an action from the available options with the aim to achieve the best result. The aim of reinforcement learning is to create a mapping between situations and actions in order to get the best reward. Deep reinforcement learning marries the power of deep neural networks to understand the world with the ability to act on that understanding. What differentiates reinforcement learning from the other forms of machine learning is that the system is not taught what output to give, instead the system learns from experience. There is no human supervision involved. It is a closed loop process, i.e. the output of one step forms the basis of input of the next step. Reinforcement learning finds application in problems where the entire problem is spread over multiple stages or steps and sequential decision making is needed.

### D. Our Contribution

Our contribution, in this research, can be broadly stated as:
1. We have modified the Proximal Policy Optimization (PPO) algorithm taken from Open AI Baselines and added sigmoid function as the activation function.
2. In order to introduce unpredictability in the environment, we have added Ornstein Uhlenbeck noise into the system. This provides the necessary unpredictability which is expected in the real world.
3. With the aim to minimize energy depletion by way of change of motor/propeller speeds, we have designed the reward function of our PPO algorithm to minimize this change.
4. We train our system to achieve the target angular velocity from the existing angular velocity while minimizing change in ESC instructions passed to the motors.

## II. Related Work

Attitude control has been handled differently in different papers. Some have taken attitude control as an isolated problem while some have clubbed it with navigation control. We shall be first discussing some of the papers which target the attitude control problem as a standalone issue.
Aggregated Multi Reinforcement Learning Systems (Jiang & Kamel, 2007, pp. 41-46) test pitch control of quadcopter on multiple reinforcement learning algorithms. Weight learning has been treated separately, with Weighted Borda Count Aggregation giving the best performance. The impact of wind and other external factors are considered in Constrained Finite Time Optimal Control Scheme (Alexis, Nikolakopoulos & Tzes, 2010, pp. 4451-4455). They assume a rigid and symmetrical quadcopter structure. It is also assumed that drag and forces of thrust have proportional impact on the propeller speed. Pioneering work has been done by Hwangbo et al. (2017, pp. 2096-2103) where raw sensory data was mapped to the motor velocity. They have implemented their approach in the real world. Using the deterministic on-policy approach, they have used two networks - a value network and a policy network.Training was done model free. They have incorporated waypoint tracking, recovery tests and improved on the tracking error. Deep Model Based Reinforcement Learning in (Lambert, Drew, Yaconelli, Levine, Calandra & Pister, 2019, pp. 4224-4230) proposes to counter system based and dynamic limitations using techniques like firmware changes, system design and model adaptations. Their design is portable to different types of quadcopters.
We also came across multiple works of literature where attitude control had been treated along with navigation control of quadcopters. Attitude control has been presented as the inner loop while navigation control forms the outer loop of the entire problem. Abbeel et al. (2006, pp. 1-8) propose a hybrid algorithm where the model is approximate. They reason that as it is, continuous control tasks need model free algorithms. Their approach is to check the policy evaluations on real world scenarios along with using the approximation model to fine tune the local changes. In (Bou-Ammar, Voos & Ertel, 2010, pp. 2130-2135), feedback linearization is used along with decoupling to achieve attitude control. Santos et al. (2012, pp. 1-16) propose Finite Action-set Learning Automata algorithm which has two hosts. One host is used to perform control loops while the other performs the robot model. The paper by (Lou & Guo, 2016) creates a link between attitude control and linear acceleration with the help of Proportional Derivative equation. The outer loop navigation controller tracks attitude control using backstepping technique that is command filtered. TEXPLORE, by (Imanberdiyev, Fu, Kayacan & Chen, 2016, pp. 1-6) uses target exploration. Decision trees are used to aid in learning and multiple learned models are called forests. In Vision-Based Autonomous Multirotor Landing (Rodriguez-Ramos, Sampedro, Bavle, Moreno & Campoy, 2018, pp. 1010-1017), the authors use RGB cameras along with Inertial Measurement Units and sensors, to achieve attitude control. It uses a reinforcement learning algorithm Deep Deterministic Policy Gradients (DDPG). A feedback controller, that uses a quarternion, to describe euler angles and attitude errors has been used in the integrated reinforcement learning algorithm in (Li, Durdevic & Yang, 2019, pp. 55-60) A neural network called Cerebellar Model Articulation Controller is the basis of work done in (Nie, Zheng & Zhu, 2019). The pitch angle is expressed as a function of the velocity of the UAV. Shin et al. (2019, p. 5571), suggest using two joint convolutional neural networks, thus creating a duelling architecture. They have tested their work on AirSim's Woodland package. Twin Delayed Deep Deterministic Policy Gradients (TD3) in (Koning, 2020) adopts a two step approach where the quadcopter's inertial matrix is calculated using multiple point masses and the weight of the centre. Thereafter, a physical model is recreated using Matlab Simulink. The authors claim a 94 % success in achieving attitude control. Another interesting proposal by Bekal et al. in (2020, p. 0898) suggests applying DDPG algorithm for pitch control while PID controllers are used for roll and yaw control. Bøhn et al. (2019, pp. 523-533) use Proximal Policy Optimization (PPO) on fixed wing UAV's attitude control. This paper treats yaw rate as a function of the roll angle. The research work in (Zhou, Yin, Wang & Wang, 2014) performs a mathematical analysis of attitude control in fixed wing UAVs by doing feedback linearization and dividing the model into three separate channels for roll, pitch and yaw. The following sections are organised as follows: In section 3, we discuss the preliminaries. Section 4 talks about our proposed method. We discuss our experimental evaluation in Section 5. In Section 6, we discuss our research along with the scope of future work while Section 7 contains the conclusion.

## III. Preliminaries

In this section we provide an overview of all the ideas used in our research as well as the explanation of the terms used.

### A. Terms used in Deep Reinforcement Learning

*Agent (Fig. 4):* It consists of a neural network having one input layer, one output layer and multiple hidden layers. Each layer has multiple nodes called neurons and an activation

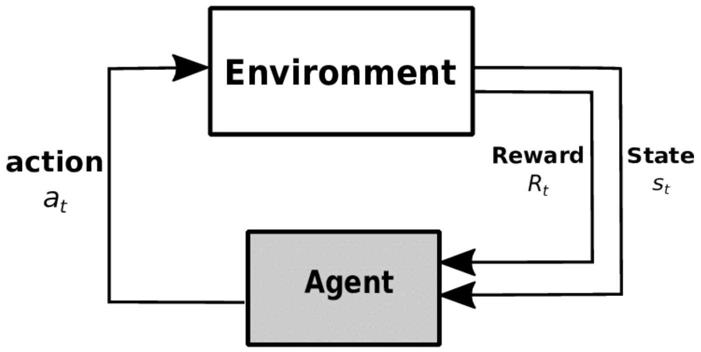

Fig. 4. Agent environment interaction in Reinforcement learning (Amiri, Mehrpouyan, Fridman, Mallik, Nallanathan & Matolak, 2018, pp. 1-7)

function. The agent works based on a reinforcement learning algorithm (Sutton & Barto, 2018).
*Action:* The output of the agent defines the action and is denoted by $A_t \in \mathcal{A}(S_t)$ where $(S_t)$ is one of the states of the environment(discussed next). The action is either an action value function $q_\pi(s, a)$ or a state value function $v_\pi(s)$. The value function is an estimate of how good an action is or how good a state is. At each time step, the agent maps states to the probabilities of selecting an action, known as the *policy*.
*Policy:* It is represented by $\pi_t$ and $\pi_t(a|s)$ means that $A_t = a$ if $S_t = s$. A policy may be:

- *Deterministic:* Used in environments where there is no uncertainty, a deterministic policy maps states to actions. $\pi_t(s) = a_t$
- *Stochastic:* A stochastic policy gives a probability distribution over actions. It is used when the environment is uncertain. $\pi_t(a|s) = \mathcal{P}(a_t|s_t)$

*Environment:* It is the world with which the agent interacts. The action is sent by the agent to the environment. The environment receives this and responds to it. The response is sent back to the agent. In our case, the UAV is the environment.
*State:* The condition of the environment is its state. The environment returns its state to the agent.
*Reward:* The reward, by far the most critical part of any reinforcement learning algorithm's design, defines its goal. It is a single number sent by the environment at each time step. The agent aims to maximize this reward at the end of the whole process. This reward depends on the agent's action and the environment's current state.
The agent creates a model based on its representation of the state and the reward. Reinforcement learning algorithms can be divided into model based and model free algorithms.
*Model based:* As the agent learns the model of the world, it constructs a model or an estimate of the model about how the environment works. Once this is done, the agent is able to plan into the future by predicting how the actions will change the world. E.g. Chess, Go. They are extremely sample efficient.
*Model free:* The agent doesn't create a model of the world. Model free approach is of two kinds.

- *Value based:* Value based agents constantly update how good is an action taken at a given state and use that model to take the optimal action. They don't learn a policy, they learn how good it is to be in a state and use that information to pick the best one. These methods are applicable for deterministic policies only.
- *Policy based:* They directly learn a policy function which maps states to actions, i.e. we select the action without using a value function. They can work on stochastic policies also.

### B. Discrete and continuous action spaces

Reinforcement learning has been found to deliver state of the art results in problems like computer games, Go and chess (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare & Petersen, 2015, pp. 529-533) (Mnih, Kavukcuoglu, Silver, Graves, Antonoglou, Wierstra & Riedmiller, 2013) and (Silver, Huang, Maddison, Guez, Sifre, Driessche & Dieleman, 2016, pp. 484-489). The Deep Q Network (DQN) algorithm used for such tasks performed in low dimensional and discrete action spaces. However, we need algorithms that can work in the high dimensional and continuous action spaces like the attitude control problem. (Lillicrap, Hunt, Pritzel, Heess, Erez, Tassa & Wierstra, 2015) gave a landmark reinforcement learning algorithm using policy gradient approach to solve continuous tasks. Since then, many other reinforcement algorithms have come up for solving complex tasks in continuous action spaces.

### C. Policy gradient

Policy gradient is used in on-policy approaches. It directly optimizes the policy space such that we take those actions that have a greater probability of giving better rewards. At the same time, we decrease the chances of selecting a policy if it gives a high negative reward. The policy network forms a representation of the environment space and its output presents a stochastic estimate of the probability of different actions. The expected reward $J(\theta)$ is the sum of probability of the trajectory $\tau$ multiplied by the corresponding rewards.

$$J(\theta) = \sum_\tau \mathcal{P}(\tau; \theta) R(\tau) \tag{1}$$

where $\theta$ is the policy used to create the trajectory $\tau$ where

$$\tau = (s_1, a_1, s_2, a_2, ..., s_t, a_t) \tag{2}$$

We need to find a $\theta$ that gives the maximum reward.

$$\max_\theta J(\theta) = \max_\theta \sum_\tau \mathcal{P}(\tau; \theta) R(\tau) \tag{3}$$

If the environment is too complex that value based methods like DQN cannot learn, policy gradient methods will still work and learn a good policy because they learn the policy directly. Policy gradient algorithms have faster convergence. They are capable of learning stochastic policies which value based methods cannot. They can also deal with continuous action spaces in a much easier way. However, they are sample inefficient and can become highly unstable. They have poor credit assignment for delayed rewards. As we progress beyond Vanilla Policy Gradient techniques, we have reached on policy optimization processes like Trust Region Policy Optimization(TRPO) and Proximal Policy Optimization(PPO). The action one takes, influences the rest of the optimization process. So, the action has to be taken very carefully and specially avoid taking bad actions.

## D. Role of IMU and ESC

In order to implement our proposal to solve the attitude control problem of quadcopters using deep reinforcement learning, we have used GymFC (Koch, 2018), a flight controller environment. Built upon the Open AI Gym environment's API, GymFC has an action space and an observation space. The observation space defines the bounds of the environment's observations while the action space states the bounds of the action input. Both, the action space and the observation space belong to the continuous domain. Within the Inertial Measurement Unit (IMU), there is a component called the gyroscope which computes how fast the quadcopter is rotating about the roll, pitch and yaw axis. This value, the angular velocity of the quadrotor, is returned by the environment to the agent and forms a part of its input. It is represented by $\Omega(t) = [\Omega_\phi(t), \Omega_\theta(t), \Omega_\psi(t)]$. $\Omega_\phi(t), \Omega_\theta(t)$ and $\Omega_\psi(t)$ denote the angular velocities about the roll, pitch and yaw axes respectively, of the quadcopter. The other component returned by the environment is the velocity of each of the four motors computed and sent by their respective Electronic Speed Controllers (ESCs), denoted by $\omega(t) = [\omega_0(t), \omega_1(t), \omega_2(t), \omega_3(t)]$. $\omega_0(t), \omega_1(t), \omega_2(t)$ and $\omega_3(t)$ is the velocity of each of the four motors of the quadcopter measured in Revolutions per Minute (RPM). The agent receives this value which forms the environment's current state, applies the reinforcement learning algorithm and sends its response (action) in the form of control signals to the aircraft's actuators. The control signals are in the form of Pulse Width Modulation (PWM) signals $a(t) = [a_0(t), a_1(t), a_2(t), a_3(t)]$ for each of the four motors sent to the ESCs.

## E. Neural Networks

The agent's function of mapping a state to an action can simply be expressed as a table where there are states on the y axis and actions on the x axis. Each cell of the table denotes the reward received. As we use the reinforcement learning algorithm, we keep updating this table. However, any real life problem cannot be represented in a single table as that would result in an exceedingly large number of rows and columns in the table. Our attitude control problem is spread across the three dimensions as discussed in the previous section and hence, the agent's action cannot be represented in a table - enter deep reinforcement learning. The deep part of the reinforcement learning is the neural network. It consists of an input layer, output layer and hidden layers, the constitution of which depends on the reinforcement algorithm used. Neural networks aggregate all the information received in the past to the useful information that is pertinent to the task at hand. The aim of reinforcement learning is to train the agent to act in the real world using this information.

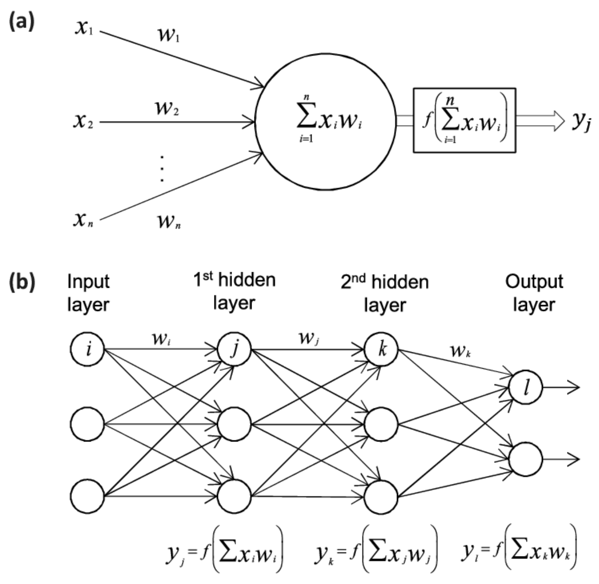

Fig. 5. (a) A neuron - the building block of a neural network (b) A multilayer neural network (Vieira, Pinaya & Mechelli, 2017)

## F. Layers and activation function

A neural network consists of an input layer, an output layer and at least one layer in between called the hidden layer (Fig. 5). Each layer consists of nodes which is the location where computations take place. The node is designed on the concept of the neurons of the human brain which trigger an action when sufficient stimulus is received. The node multiplies input data $x_i$ with a weight $w_i$, that either increases or decreases that input, thus allocating significance to each input depending on the problem the algorithm is aiming to learn. The sum of the products of input and weights $\sum_{i=1}^{n} x_i w_i$ is further subjected to an activation function $f(\sum_{i=1}^{n} x_i w_i)$. The activation function decides whether a signal is to pass through the network and influence the final outcome. There are multiple activation functions used in neural networks, for example Tanh, Sigmoid, ReLU, etc.

## G. Proximal Policy Optimization

As discussed earlier, working in the continuous action domain requires policy based approaches. Algorithms like Deep Deterministic Policy Gradients (DDPG) (Lillicrap, Hunt, Pritzel, Heess, Erez, Tassa & Wierstra, 2015), Asynchronous Advantage Actor Critic (A3C) (Mnih, Badia, Mirza, Graves, Lillicrap, Harley & Kavukcuoglu, 2016, pp. 1928-1937), Trust Region Policy Optimization(TRPO) (Schulman, Levine, Abbeel, Jordan & Moritz, 2015, pp. 1889-1897), Proximal Policy Optimization(PPO) (Schulman, Wolski, Dhariwal, Radford & Klimov, 2017), etc. have been proposed and shown promising results. We surfed through a great deal of literature (Such, Madhavan, Liu, Wang, Castro, Li & Lehman, 2018), (Silver, Lever, Heess, Degris, Wierstra & Riedmiller, 2014), (Henderson, Islam, Bachman, Pineau, Precup & Meger, 2018), (Islam, Henderson, Gomrokchi & Precup, 2017) to zero down on PPO as our preferred algorithm for handling attitude control. The reason for selecting PPO for our problem is mainly because PPO has been reported to offer the best convergence figures amongst all (Wang, Cai, Yang & Wang, 2019). A3C requires high computational power (Gazori, Rahbari & Nickray, 2019). DDPG is an off-policy technique, i.e. it figures out the optimal policy irrespective of the agent's actions. Its exploration strategy is time consuming (Xu, Cao, Chen & Li, 2018), (Le, Vo, Kieu, Hwang, Rho & Baik, 2020) . TRPO's policy update process takes the policy too far from the current policy, which PPO improves by restricting the policy update (Schulman, Wolski, Dhariwal, Radford & Klimov, 2017). Also, PPO's objective function is very simple, which is critical for our problem of improving energy efficiency in our process. PPO algorithm (Schulman, Wolski, Dhariwal, Radford & Klimov, 2017) defines policy gradient loss as the expectation over the log of policy actions times the advantage function.

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{4}$$

The advantage function is the estimate of the relative value of the selected action. It is a measure of the action quality relative to the average action. In other words, it denotes how much better the current policy is. To calculate the advantage function, we need the discounted rewards which provide the baseline estimate.

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{5}$$

$$where\ \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

The discounted reward is the cumulative reward after the completion of the episode sequence. The baseline estimate or value function gives an estimate of the discounted return from this point onwards. Thus, the advantage function provides an estimate of how much better was the action taken based on the expectation of what would normally happen depending on the state the agent was in. When the advantage function is a positive value, we increase the likelihood of taking that action when that particular state is observed. If the advantage function is negative, we reduce the probability of taking that action when the similar state is observed. The problem is that if we keep running gradient descent on a single batch of collective experiences, we end up updating the parameters so far out of the range, that we go far away from our initial estimate and destroy our policy. The solution is to never move too far away from the old policy. In order to make sure the updated policy doesn't move too far from the existing policy, TRPO adds a Kullback-Leibler(KL) constraint. Thus, we stay close to the region where we know everything works fine. However, KL constraint adds an additional overhead to the optimization process and leads to undesirable training behaviour. Hence, there is a need to add this constraint directly to the optimization objective. PPO does it as below:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1 - \epsilon, \tag{6}$$

PPO uses actor critic approach. There are two neural networks, an actor which is policy based and a critic which is value based. The actor samples an action from a policy and the critic measures how good the chosen action is. Thus, the algorithm adds a value function error term to the policy surrogate. Also, an entropy bonus is added to provide sufficient exploration.

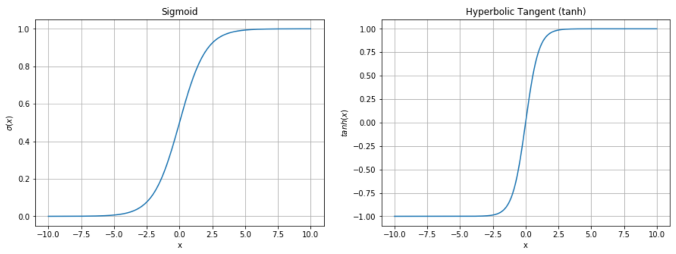

Fig. 6. Plots of Sigmoid and Tanh activation functions (Bonaccorso, 2020)

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{7}$$

The whole aim of PPO to improve TRPO is to restrict the policy update to be not too far from the current policy. PPO has a very simple objective function unlike TRPO.

### *H. Noise*

In order to apply reinforcement learning agents to real life problems, it is important that the policies should generalize in newer environments. We have to avoid overfitting of an agent's policy to only certain training environments. Regularization techniques like injecting noise into the system are required (Igl, Ciosek, Li, Tschiatschek, Zhang, Devlin & Hofmann, 2019, pp. 13978-13990). Noise induces a level of unpredictability into the system that is expected from real life scenarios. As minimizing energy consumption is our ultimate criteria, we need to replicate the real world behaviour in the simulation environment and then create a reward function to optimize energy consumption and test if our algorithm returns a better than baseline performance.

## IV. Proposed PPO based method

In order to improve the application of reinforcement learning so that it is applicable to real life problems, there are two approaches:
1. Improve the algorithms so that they can create policies that are transferable across all types of domains, including the real world, i.e. train in simulation and then transfer that training to the real world.
2. Improve the simulation in such a way that the gap between simulation and the real world reduces to the extent that things learnt in simulation are directly transferable to the real world. We group our contribution into the following four sections:

### *A. Change in activation function*

PPO uses *tanh* or hyperbolic tangent activation function in its layers. Tanh returns values in the range (-1,1). However, the ESCs of the quadcopter accept values only in the range (0,1). A workaround is to write a function that converts the obtained value of the output layer of the neural network to the action value range (0,1) Fig. 6. We have made use of *sigmoid* activation function, also known as inverse logit function, ($(\sigma(x))$) in PPO instead of tanh, which returns the values in the range (0,1), thus outputting a value in the range expected by the ESCs.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{8}$$

Infact, tanh is a rescaling of the sigmoid activation function, thus being computationally expensive. There is horizontal stretching in tanh, as well.

### B. Add Ornstein Uhlenbeck noise

In order to add uncertainty, we add Ornstein Uhlenbeck noise (Uhlenbeck & Ornstein, 1930) to our action space.

$$\pi'(s_t) = \pi(s_t|\theta_t^{\pi}) + \mathcal{N} \tag{9}$$

$\pi'(s_t)$ is the exploration policy which is constructed by adding noise from the noise process $\mathcal{N}$ to the policy $\pi(s_t|\theta_t^{\pi})$. We have used temporally correlated noise such that better exploration in physical environments is possible. Our Ornstein-Uhlenbeck noise has $\theta = 0.15$ and $\sigma = 0.3$. The Ornstein-Uhlenbeck process considers the velocity of a Brownian particle along with friction and results in temporally correlated noise with 0 as the mean (Lillicrap, Hunt, Pritzel, Heess, Erez, Tassa & Wierstra, 2015). With the introduction of Ornstein-Uhlenbeck noise, this research work can be later extended to numerous other kinds of quadcopter models as well.

### C. Minimize change in rotational velocities of each motor

With the aim of achieving attitude control while minimizing energy consumption, we have identified the key areas of energy usage in a quadcopter. It is but natural for the movable parts of a device to consume the maximum power. The motors and the propellers connected to them, are the only moving parts in a quadcopter. A large variation in revolutions per minute of the motors depletes the battery very fast. Thus, we have designed our algorithm to reduce the change in rotational velocities of the 4 motors.

### D. Proposed algorithm for Energy Efficient Attitude Control

Our attitude control algorithm aims to learn the process to achieve a given angular velocity $\Omega^*$ while minimising the energy consumption to achieve it. In each episode, we initialize this angular velocity to a random value and the PPO algorithm aims to make the angular velocity of the UAV approach this target angular velocity while maximizing the reward. In our case, minimizing the energy consumed along with achieving the target angular velocity, is the reward. We initialize the target angular velocity $\Omega^*$ to $\Omega_r^*, \Omega_p^*, \Omega_y^*$ corresponding to the roll, pitch and yaw axes with values chosen randomly. Next, we run a loop with the aim to maximize the reward, or in our case, minimize the negative reward such that reward $r_t \to 0$. Within the loop, we initialize the action $a_t$, i.e. the output of our reinforcement learning algorithm with the PWM values $y_i$ (where $i \to 0$ *to* 3) for each of the four motors. To this value $a_t$, we add the Ornstein Uhlenbeck noise $\mathcal{N}$. We send this action to the environment and receive the state $s_{t+1}$. The state includes the current angular velocity of the quadrotor, which is again a 3 element value about the roll, pitch and yaw axes along with the motor velocities (RPM) of the quadcopter model returned by the environment. We store the absolute value of difference of the motor velocity values received from the environment between two epochs and take the average of the four values $\Delta\omega$. Next, we take the angular velocity values received in state $s_{t+1}$ and find the difference between them and the target angular velocity. We square the values of each of the 3 axes, add the result and store the square root $\Delta\Omega$. We treat the differences in the motor RPM values and the angular velocity values as penalties that need to be minimized, thus optimizing energy consumption. We take the product of the motor velocity difference value $\Delta\omega$ and a constant $\alpha$ and add it to the angular velocity difference value $\Delta\Omega$ and assign a negative sign to the sum, to denote that it is a penalty that needs to be minimized. The current values of motor RPM values are copied into the previously stored motor RPM values for use in the next epoch.

**Algorithm 1** Energy efficient attitude control (EEAC)

```
1: for episode:=1,....,M do
2:     Initialize $a_t$, $\Omega$, $\omega$
3:     Initialize the environment and receive initial state $s_1$
4:     Initialize $\Omega^* = \{\Omega_r^*, \Omega_p^*, \Omega_y^*\}$
5:     for epoch t:=1,....,T do
6:         Generate an action $a_t = (y_0, y_1, y_2, y_3)$
7:         $a_t \leftarrow a_t + \mathcal{N}$ where $\mathcal{N}$ is Ornstein Uhlenbeck noise
8:         Send action $a_t$ to the environment and receive state $s_{t+1}$
9:         $s_{t+1} = \{(\Omega_r, \Omega_p, \Omega_y), (\omega_0', \omega_1', \omega_2', \omega_3')\}$
10:        $\Omega = (\Omega_r, \Omega_p, \Omega_y)$
11:        $\omega' = (\omega_0', \omega_1', \omega_2', \omega_3')$
12:        $\delta\omega = (|\omega_0' - \omega_0|, |\omega_1' - \omega_1|, |\omega_2' - \omega_2|, |\omega_3' - \omega_3|)$
13:        $\Delta\omega = \frac{1}{4}\sum_{n=0}^{3} \delta\omega_n$
14:        $\delta\Omega = ((\Omega_r^* - \Omega_r)^2, (\Omega_p^* - \Omega_p)^2, (\Omega_y^* - \Omega_y)^2)$
15:        $\Delta\Omega = \frac{1}{3}\sum_{i=0}^{2} \delta\Omega_i$
16:        $\Delta\Omega \leftarrow \sqrt{\Delta\Omega}$
17:        $r_t = -(\alpha * \Delta\omega + \Delta\Omega)$
18:        Set $(\omega_0, \omega_1, \omega_2, \omega_3) \leftarrow (\omega_0', \omega_1', \omega_2', \omega_3')$
19:        return $r_t$
20:    end for
21: end for
```

## V. Experimental Evaluation

We conducted simulations to test the effectiveness of our proposed approach. In the following sections, we have described the simulation settings and discuss the results.

### A. Simulation Settings

In order to simulate, we have used GymFC , an open source flight controller environment based on Open AI Gym. GymFC runs on Ubuntu 18.04. We have installed Gazebo v10.1.0 as the simulator for GymFC. Gazebo is the backend simulator for designing robots, in our case the quadcopter. The quadcopter is modelled as a *.sdf* file that is accessed by

TABLE I
HYPERPARAMETERS OF THE PPO AGENT

| Hyperparameter | Value |
|---|---|
| Horizon(T) | 500 |
| Adam Stepsize | $1 \times 10^{-4} \times \rho$ |
| Num. Epochs | 5 |
| Minibatch Size | 32 |
| Discount Factor ($\gamma$) | 0.99 |
| GAE Parameter ($\lambda$) | 0.95 |

Value of $\rho$ reduces gradually from 1 to 0 during training

the environment. We have installed Dynamic Animation and Robotics Toolkit (DART) version 6.7.0. DART is a physics engine which provides the algorithms and data structures for computing the dynamics of motion. GymFC integrates with Open AI Gym, which is an open source toolkit for writing and comparing reinforcement learning algorithms. We have forked from the baselines available on the GitHub page of Open AI Gym to use and manipulate the *PPO1* algorithm. The neural network we have created consists of an input layer having six nodes, an output layer with four nodes and two hidden layers with 32 nodes each. The output of the neural network is the Gaussian distribution mean with a varying standard deviation, as has been defined in the PPO paper (Schulman, Wolski, Dhariwal, Radford & Klimov, 2017) for continuous domain problems like ours. The hyperparameters used by the neural network are as stated in Table I. The Ornstein Uhlenbeck Noise parameters are taken as follows: $\mu$=0,$\theta$=0.15 and $\sigma$=0.3. We trained our model for a total of 10 million steps on a Ubuntu 18.04 system having 4 core i5-8250U CPU and the training process took 16 hours. We created checkpoints at every 100,000 steps. Thus, we had 100 checkpoints to test and evaluate our algorithm's performance. We have compared the results of our algorithm with two baselines, one is the PID baseline, where each motor of the quadcopter is connected to a Proportional Integral Derivative (PID) controller and the aim is to achieve a motor velocity target which is the product of control signal and the maximum motor velocity constant. The other baseline is the default PPO controller of GymFC. We used TensorFlow 1.14 for compilation of graphs. We have evaluated our results on the following metrics:
*Reward* Our reward function, as stated previously, is a function of the difference in angular velocity and the change in motor RPM values. After running the training process, we evaluated and compared the performance of our algorithm with the GymFC baseline on the following four parameters:

- *Mean Absolute Error (MAE)* This is the difference between the roll, pitch and yaw values of the quadcopter (angular velocities) and the set point roll, pitch and yaw values (target angular velocities).

- *Average Motor Velocities (RPM values)* Average of the velocities of the four motors.
- *Average Change in Motor Velocities* Average of the absolute values of the difference of the motor velocities of the current time step and the previous time step.
- *Average Reward* Output of our algorithm at each time step.

*Energy efficiency* To calculate the energy consumed, we use the formula (Oosedo, Konno, Matumoto, Go, Masuko, Abiko & Uchiyama, 2010, pp. 254-259):

$$EnergyConsumption(E) = C_P \rho n^3 d^5 \quad (10)$$

where $C_P$ is the co-efficient of energy consumption, $\rho$ is atmospheric density, n is the change in propeller speed and d is propeller diameter. $C_P = 2\pi C_Q$ where $C_Q$ is the co-efficient of torque and $C_Q = 1.38 \times 10^{-3}$ in GymFC. $\rho = 1.275 kg/m^3$ at sea level and at zero celsius temperature (Jakubowski & Foster, 1999), (Kidder, 1921). The motor model in this simulation is derived from PX4 Gazebo simulation plugins where the model has a propeller diameter $(d) = 10cms$. We plot the energy consumed at each of the 100 checkpoints and compare our results with the PPO baseline.

## B. Results and analysis

In this section, we have displayed the results of our simulations in the form of graphs obtained. As stated previously, we have compared our proposed Energy Efficient Attitude Control (EEAC) approach with the PPO baseline and compared the results on four parameters. Fig. 7(a) depicts the Mean Absolute Error (MEA), which is the difference between the target angular velocities and the angular velocities of the quadcopter about the roll, pitch and yaw axes. The baseline performs better in this metric throughout the lifetime of the 10 million training process. From the very start, the MAE values using our approach are higher than found in the baseline. At around 1.5 million steps, this gap widens only to narrow at 5 million steps. Thereafter, there is little difference in the MAE values of our approach and the baseline. In Fig. 7(b), we compare the average of the velocities of the four motors. We find the average velocities in both the approaches to be more or less the same. At about 6 million timesteps, the average velocities of the motors of the baseline seem to reduce as compared to our approach. The reason for this is that the model starts to converge at this point. In Fig. 7(c), we compare the change in the motor velocities. We find our approach performing considerably better than the baseline, in this metric. Right through the training setup, the graph depicts frequent spurts in average motor velocities of the baseline approach. Our method shows a much smoother change in RPM values, thereby implying that we fare better than the baseline in this particular parameter. One thing that draws our attention is the visible spurts in the graph towards the end of the simulation, which is due to the algorithm starting to converge and taking into consideration the other parameters of the reward function. In Fig. 7(d), we depict the average reward at each timestep. A symmetrical log scale is used for the y-axis. The reward

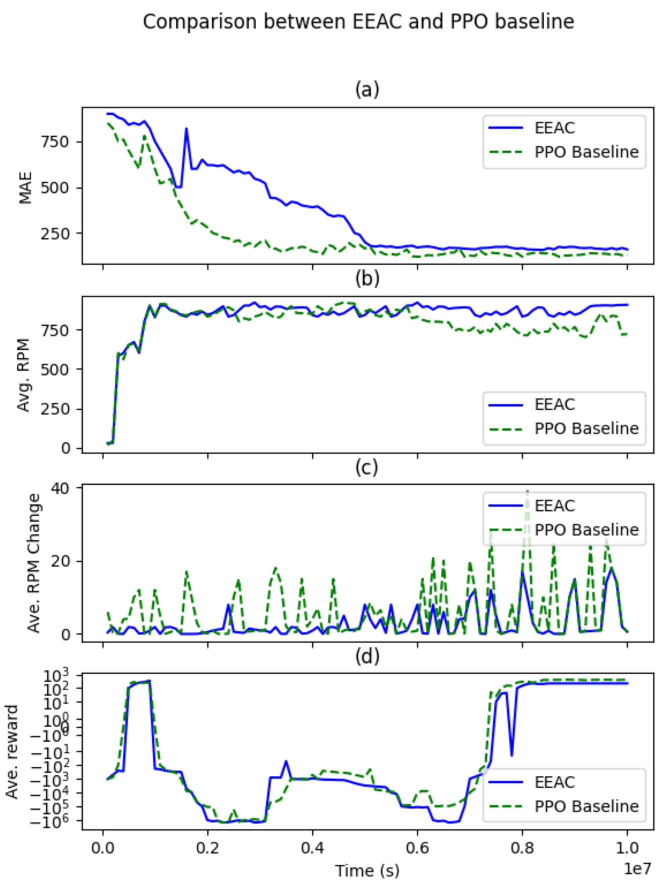

Fig. 7. Reward comparison between EEAC and PPO baseline

values increase during initial training phase, corresponding to the reduction in MAE values. However, when the algorithm tries to minimize energy consumption, the reward function values go into negative territory. At about 7.5 million steps, the model starts to converge. If we need to select any particular checkpoint that can be used for attitude control in actual quadcopters, we can safely use the results of any checkpoint after 8 million steps for the purpose. (Later, we disclose the comparison graph of one such checkpoint). The reward function returns more or less similar results for EEAC and the baseline, all through out the 10 million step run, thereby implying that for the same performance of the two processes, we our able to achieve better energy efficiency. This is depicted in Fig. 8. This graph plots the energy consumed at each of the 100 checkpoints created during the training process. We find the baseline process consuming considerably higher energy with major crests and troughs during the first 30 checkpoints. Thereafter, the variation in energy consumption stays at around 2.55e8. Our EEAC approach's energy consumption hovers around 2.475e8, a major reduction in value.
Since our model converges after 8 million steps, we have selected a random step after that point and tested the quadcopter's angular velocity values of our approach and compared the same with the PID baseline Fig. 9. The target values of roll, pitch and yaw axes are drawn in black and the PID controller

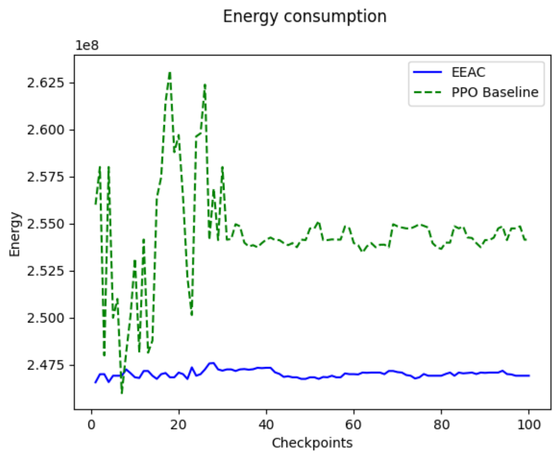

Fig. 8. Comparison of energy efficiency between EEAC and PPO baseline

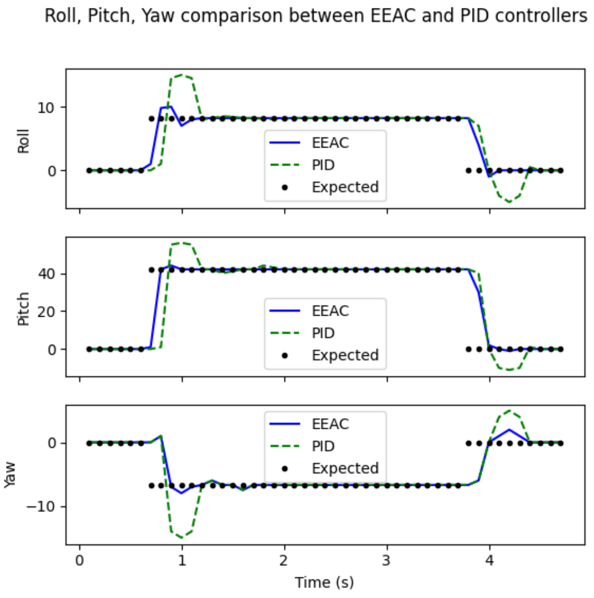

Fig. 9. Roll, pitch, yaw comparison between EEAC and PID Controllers

and EEAC controller try to achieve the expected value using respective techniques. Although both the approaches achieve the target values, as can be seen from the graphs, there is wide fluctuation in the case of PID controller. The target values change twice during the run, once at about .75 timestep and again at around 3.75 timestep. At both the time instances, we find the angular velocities using PID controller varying sharply from the previous values. This rapidly increases energy consumption. On the other hand, the EEAC controller achieves the target roll, pitch and yaw angular velocities very smoothly.

## VI. Conclusion

In this paper, we have implemented a way to reduce energy consumption in the quadcopter while achieving attitude control using deep reinforcement learning. We have modified the existing Proximal Policy Optimization algorithm and incorporated sigmoid activation function to provide output in the range (0,1) that can be used as PWM instructions to the quadcopter directly. We have modelled real world uncertainties by including Ornstein Uhlenbeck noise. The major source of energy consumption in a UAV is variation of its motor velocities. We have created our reinforcement learning algorithm's reward function in such a way that major weightage is given to minimizing variations in propeller revolutions. Our reward function aligns the angular velocity of the quadcopter to the target angular velocity gradually, such that there are no major changes in the motor RPM values.

Our current work can to be extended to incorporate the navigation control problem so that all the six degrees of freedom of the quadcopter can be managed using reinforcement learning. Also, we have disabled the effect of gravity in Gazebo, while running the simulations. In case the impact of gravity is incorporated, it will present interesting issues in the attitude control as well as the navigation control problems.

## References

[1] Aashmango4793 (2020)- Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=81688701 Accessed 29 June 2020

[2] Bonaccorso G., https://www.oreilly.com/library/view/mastering-machine-learning/9781788621113/913c98cf-b765-4840-92e1-d873df659c6a.xhtml Accessed 19 July 2020

[3] Vieira, Sandra & Pinaya, Walter & Mechelli, Andrea. (2017). Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications. Neuroscience & Biobehavioral Reviews. 74. 10.1016/j.neubiorev.2017.01.002.

[4] Abbeel, P., Quigley, M., & Ng, A. Y. (2006, June). Using inaccurate models in reinforcement learning. In Proceedings of the 23rd international conference on Machine learning (pp. 1-8).

[5] Alexis, K., Nikolakopoulos, G., & Tzes, A. (2010, June). Constrained optimal attitude control of a quadrotor helicopter subject to wind-gusts: experimental studies. In Proceedings of the 2010 American Control Conference (pp. 4451-4455). IEEE.

[6] Amiri, R., Mehrpouyan, H., Fridman, L., Mallik, R. K., Nallanathan, A., & Matolak, D. (2018, May). A machine learning approach for power allocation in HetNets considering QoS. In 2018 IEEE International Conference on Communications (ICC) (pp. 1-7). IEEE.

[7] Bekar, C., Yuksek, B., & Inalhan, G. (2020). High Fidelity Progressive Reinforcement Learning for Agile Maneuvering UAVs. In AIAA Scitech 2020 Forum (p. 0898).

[8] Bøhn, E., Coates, E. M., Moe, S., & Johansen, T. A. (2019, June). Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In 2019 International Conference on Unmanned Aircraft Systems (ICUAS) (pp. 523-533). IEEE.

[9] Bou-Ammar, H., Voos, H., & Ertel, W. (2010, September). Controller design for quadrotor uavs using reinforcement learning. In 2010 IEEE International Conference on Control Applications (pp. 2130-2135). IEEE.

[10] dos Santos, S. R. B., Nascimento, C. L., & Givigi, S. N. (2012, March). Design of attitude and path tracking controllers for quad-rotor robots using reinforcement learning. In 2012 IEEE Aerospace Conference (pp. 1-16). IEEE.

[11] Gazori, P., Rahbari, D., & Nickray, M. (2019). Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach. Future Generation Computer Systems.

[12] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018, April). Deep reinforcement learning that matters. In Thirty-Second AAAI Conference on Artificial Intelligence.

[13] Huo, Y., Li, Y., & Feng, X. (2020, January). Tiltrotors Position Tracking Controller Design Using Deep Reinforcement Learning. In IOP Conference Series: Materials Science and Engineering (Vol. 751, No. 1, p. 012047). IOP Publishing.

[14] Hwangbo, J., Sa, I., Siegwart, R., & Hutter, M. (2017). Control of a quadrotor with reinforcement learning. IEEE Robotics and Automation Letters, 2(4), 2096-2103.

[15] Igl, M., Ciosek, K., Li, Y., Tschiatschek, S., Zhang, C., Devlin, S., & Hofmann, K. (2019). Generalization in reinforcement learning with selective noise injection and information bottleneck. In Advances in Neural Information Processing Systems (pp. 13978-13990).

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

[17] Islam, R., Henderson, P., Gomrokchi, M., & Precup, D. (2017). Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. arXiv preprint arXiv:1708.04133.

[18] Jakubowski Jr, T., & Foster, J. K. (1999). U.S. Patent No. 5,904,323. Washington, DC: U.S. Patent and Trademark Office.

[19] Jiang, J., & Kamel, M. S. (2007, August). Pitch Control of an Aircraft with Aggregated Reinforcement Learning Algorithms. In 2007 International Joint Conference on Neural Networks (pp. 41-46). IEEE.

[20] Kidder, F. E. (1921). The Architects' and Builders' Handbook. John Wiley & Sons, Incorporated.

[21] Koch, W., Mancuso, R., West, R., & Bestavros, A. (2019). Reinforcement learning for UAV attitude control. ACM Transactions on Cyber-Physical Systems, 3(2), 1-21.

[22] Koch, W., Mancuso, R., & Bestavros, A. (2019). Neuroflight: Next generation flight control firmware. arXiv preprint arXiv:1901.06553.

[23] Koch, W. (2018a). GymFC. https://github.com/wil3/gymfc

[24] Koning, T. (2020). Low level quadcopter control using Reinforcement Learning: Developing a self-learning drone.

[25] Lambert, N. O., Drew, D. S., Yaconelli, J., Levine, S., Calandra, R., & Pister, K. S. (2019). Low-level control of a quadrotor with deep model-based reinforcement learning. IEEE Robotics and Automation Letters, 4(4), 4224-4230.

[26] Le, T., Vo, M. T., Kieu, T., Hwang, E., Rho, S., & Baik, S. W. (2020). Multiple Electric Energy Consumption Forecasting Using a Cluster-Based Strategy for Transfer Learning in Smart Building. Sensors, 20(9), 2668.

[27] Li, S., Durdevic, P., & Yang, Z. (2019). Optimal Tracking Control Based on Integral Reinforcement Learning for An Underactuated Drone. IFAC-PapersOnLine, 52(8), 55-60.

[28] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[29] Lin, X., Yu, Y., & Sun, C. (2019). Supplementary reinforcement learning controller designed for quadrotor UAVs. IEEE Access, 7, 26422-26431.

[30] Liu, H., Zhao, W., Lewis, F. L., Jiang, Z. P., & Modares, H. (2019, July). Attitude Synchronization for Multiple Quadrotors using Reinforcement Learning. In 2019 Chinese Control Conference (CCC) (pp. 2480-2483). IEEE.

[31] Lou, W., & Guo, X. (2016). Adaptive trajectory tracking control using reinforcement learning for quadrotor. International Journal of Advanced Robotic Systems, 13(1), 38.

[32] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

[33] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[34] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016, June). Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928-1937).

[35] Nie, C., Zheng, Z., & Zhu, M. (2019). Three-dimensional path-following control of a robotic airship with reinforcement learning. International Journal of Aerospace Engineering, 2019.

[36] Oosedo, A., Konno, A., Matumoto, T., Go, K., Masuko, K., Abiko, S., & Uchiyama, M. (2010, December). Design and simulation of a quad rotor tail-sitter unmanned aerial vehicle. In 2010 IEEE/SICE International Symposium on System Integration (pp. 254-259). IEEE.

[37] Rodriguez-Ramos, A., Sampedro, C., Bavle, H., Moreno, I. G., & Campoy, P. (2018, October). A Deep Reinforcement Learning Technique for Vision-Based Autonomous Multirotor Landing on a Moving Platform. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1010-1017). IEEE.

[38] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, June). Trust region policy optimization. In International conference on machine learning (pp. 1889-1897).

[39] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[40] Shin, S. Y., Kang, Y. W., & Kim, Y. G. (2019). Obstacle Avoidance Drone by Deep Reinforcement Learning and Its Racing with Human Pilot. Applied Sciences, 9(24), 5571.

[41] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484-489.

[42] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014, June). Deterministic policy gradient algorithms.

[43] Such, F. P., Madhavan, V., Liu, R., Wang, R., Castro, P. S., Li, Y., ... & Lehman, J. (2018). An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents. arXiv preprint arXiv:1812.07069.

[44] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[45] Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the Brownian motion. Physical review, 36(5), 823.

[46] Wang, L., Cai, Q., Yang, Z., & Wang, Z. (2019). Neural policy gradient methods: Global optimality and rates of convergence. arXiv preprint arXiv:1909.01150.

[47] Xu, Z. X., Cao, L., Chen, X. L., & Li, C. X. (2018). Reward-Based Exploration: Adaptive Control for Deep Reinforcement Learning. IEICE Transactions on Information and Systems, 101(9), 2409-2412.

[48] Zhou, W., Yin, K., Wang, R., & Wang, Y. E. (2014). Design of attitude control system for UAV based on feedback linearization and adaptive control. Mathematical Problems in Engineering, 2014.