Journal of Intelligent & Robotic Systems (2021) 102: 77
https://doi.org/10.1007/s10846-021-01412-3

REGULAR PAPER

Reinforcement Learning versus Conventional Control for Controlling a Planar Bi-rotor Platform with Tail Appendage

Halil Ibrahim Ugurlu[1] · **Sinan Kalkan**[2,3] · **Afsar Saranli**[4]

Received: 15 September 2020 / Accepted: 3 May 2021 / Published online: 8 July 2021

Abstract

In this paper, we study the conventional and learning-based control approaches for multi-rotor platforms, with and without the presence of an actuated "tail" appendage. A comprehensive experimental comparison between the proven control-theoretic approaches and more recent learning-based ones is one of the contributions. Furthermore, an actuated tail appendage is considered as a deviation from the typical multi-rotor morphology, complicating the control problem but promising some useful applications. Our study also explores, as another contribution, the impact of such an actuated tail on the overall position control for both the conventional as well as learning-based controllers. For the conventional control part, we used a multi-loop architecture where the inner loop regulates the attitude while the outer loop controls the position of the platform. For the learning controller, a multi-layer neural network architecture is used to learn a nonlinear state-feedback controller. To improve the learning and generalization performance of this controller, we adopted a curricular learning approach which gradually increases the difficulty of training samples. For the experiments, a planar bi-rotor platform is modeled in a 2D simulation environment. The planar model avoids mathematical complications while preserving the main attributes of the problem making the results more useful. We observe that both types of controllers achieve reasonable control performance and can solve the position control task. However, neither one shows a clear advantage over the other. The learning-based controller is not intuitive and the system suffers from long training times. The architecture of the multi-loop controller is handcrafted (not required for the learning-based controller) but provides a guaranteed stable behavior.

Keywords Flight control · Tail appendage · Conventional control · Deep reinforcement learning

1 Introduction

Multi-rotor unmanned aerial vehicles (UAVs) have found vast commercial applications in recent years and hence are also the focus of intense study [34, 41]. Existing platforms range from transportation size [28] to micro size [6] and operate using multiple propellers generating thrust to overcome gravity to hover. They are considered *non-linear* (due to gravity) and *under-actuated* systems since the number of actuators is often less than the platform's degree-of-freedom (six in 3D space). Because of these properties, automatic control problem for a multi-rotor system is challenging.

In this paper, we consider the control approaches for multi-rotor systems in two broad categories: 'conventional' control-theoretic approaches and learning-based approaches. In conventional control-theoretic approaches [8, 27, 40], the task is often divided into a number of sub-tasks such as, attitude, altitude and position control. Linear and non-linear control techniques are employed to design different controllers for these sub-tasks with carefully crafted hierarchy between them. The advantage is that the structure of the system and the control problem is well exposed within this hierarchy and failure modes can be traced and analyzed. However, this approach necessitates a careful breakdown of the system followed by system-identification and controller parameter tuning often requiring optimization-based approaches.

In learning-based control approaches, the controller is a nonlinear functional mapping (called a *policy*) from the platform states to its actuators. The form of this mapping is often a multi-layer network structure whose parameters are learned by applying machine learning methods. Thus the

✉ Halil Ibrahim Ugurlu
halil@ece.au.dk

Extended author information available on the last page of the article.

learned policy is the controller. For example, recent work [3, 22, 33] propose 'deep learning'-based control policies that implement a particular neural network structure. Following recent advances in Deep Reinforcement Learning methods starting from Deep Q-Networks [32], these structures are also employed to control dynamical robotic systems, in particular dynamic legged robots [21] as well as aerial robotic systems [33].

The aim of the present study is to comprehensively analyze the advantages and disadvantages of both conventional control-theoretic approaches as well as the more recent learning-based control approaches within a common simulation-based framework. To the best of our knowledge, this is the first study that performs such an analysis. Similar to the simplification handled by previous studies [5, 13, 35], instead of considering the full 6-DOF multi-rotor in 3D space, we consider a planar version of the problem: a planar *bi-rotor* platform in two-dimensional space. This choice considerably simplifies the mathematical complexity but retains the essential properties of the problem: non-linearity and under-actuated nature of the problem. As test scenarios, we consider positional control of the bi-rotor in both conventional and learning-based control cases. Position control of 3D multi-rotor, which is moving the multi-rotor between two equilibrium hovering states, is identical with control of 2D bi-rotor since it is manipulated on a vertical plane that includes starting and goal position. Apart from the simple bi-rotor system, we also evaluate the presence of a tail-appendage. Such a tail, often present and used in nature to improve animal motion dexterity, increases the complexity of the bi-rotor system and makes the control problem more difficult. Whether a torque actuated tail can be controlled with both approaches and how much performance is gained/lost by this addition are interesting investigative directions of our work.

1.1 Related Work

Controlling non-linear, under-actuated, agile and high-speed platforms such as quad-rotor UAVs is challenging, posing problems such as high state-space dimensions, concerns of stability and robustness as well as considerations of response time [7, 40]. Hand-crafted, low-level controllers have been proposed for the control of multi-rotor systems. These have recently been challenged by the emergence of learning-based approaches in the literature as an alternative to classical control theory [3, 7, 12, 29, 36]. A recent literature review [40] compares these different approaches in terms of ideas and implementation, however does not provide a fair performance and behavioral comparison within a common controlled environment. In a number of studies, different applications are considered: For example, in [29], a deep network is used to make a UAV follow a hand-drawn trajectory while [12, 36] trains a UAV to fly using its vision sensors through cluttered outdoor environments. Also, there are differences in solution approaches: [12, 36] uses learning by imitation while [7] uses an iterative method for function fitting which receives supervision from a feedback-based controller. A reference controller is used in [3] to train the deep network to demonstrate its trajectory generalization capability. None of these various studies attempts a comparison with a baseline conventional control-theoretic controller structure.

The review study by Tang and Kumar [40] presents a detailed overview of various control-theoretic approaches to our problem. In this study, we adopt a conventional control-theoretic controller (detailed in Section 3) that adopts the common structural properties described in this survey, such as imposing a position-attitude hierarchy.

Advances in deep learning and novel neural network architectures made it possible to apply them to dynamic system control problems through *deep reinforcement learning* (DRL) and several methods have been proposed for DRL. Starting from "Deep Q Networks" (DQN) [32] trained for playing Atari games, numerous other methods have been proposed for DRL. With promising results on using DRL for robot control problems, they have also been applied to quad-rotor UAV control. Sadeghi et al. [37] has used a DQN to teach a UAV to avoid obstacles in indoor environments from a monocular camera. Zhang et al. [42] has combined model-predictive control with DRL to improve performance. Hwangbo et al. [22] has used an actor-critic network to teach a UAV to go to a nearby way-point. Koch et al. [26] has used DRL to learn inner loop attitude controller. A more recent study [33] employs Proximal policy optimization (PPO) [38], which is also used in this study with a similar setting, to learn a control policy for multiple quad-rotors with different sizes.

The tail of many animals is an evolutionary feature that helps the animal with efficiency and balance during locomotion. Tail appendage for bio-inspired robots is often used with similar motivation, in order to improve the dynamic capabilities of the robotic platform [9, 30, 43]. Differently from these studies and from our focus in the present paper, some studies also consider such a tail not as a desirable actuator but as a source of unwanted disturbance to be compensated for using the existing actuators [1, 23].

As an example to the active usage of tail in a multi-rotor system, Demir et al. [14] proposed to use the below-hanging battery of the system as an actuated tail-assistance for stabilizing platform attitude during hover. In the present work, we extend the idea to use the tail for attitude control

during active flight as well including transition to forward flight and transition back to hover. We also consider the learning based control of the tail for this purpose.

The way the data is provided to the network has an impact on performance in neural networks. Bengio et al. observed that if the training data is presented to the network randomly during supervised learning, it takes longer for the network parameters to converge [4]. Moreover, it is a well-known problem that deep learning is a non-convex optimization problem and hence suffers from the presence of many local minima [24, 25]. Bengio et al. showed that starting the training procedure with simpler examples, and introducing gradually more complex ones as the network learns the easier subtasks improves the convergence speed and the quality of the converged minima. Curriculum based methods are not limited to supervised learning and there are studies [16, 18, 19, 31] that combine curriculum learning with DRL algorithms.

1.2 Contributions and Outline

The main contributions of our work are as follows:

- To the best of our knowledge, this is the first attempt to present a comparative analysis of conventional and learning-based controllers for multi-rotor systems.
- We extend the multi-rotor system with a tail-appendage as in [14]. Differently, we also investigate its benefit in improving both position and attitude control performance during flight for both learning-based and conventional control cases.
- We introduce a curricular learning strategy which gradually increases the difficulty of training examples to further improve the performance of the learning based controller.

Our comparative study is conducted with a planar bi-rotor system. We believe the planar model preserves the essential properties of a multi-rotor system for a comparative study between conventional and learning based control approaches and has been used in the literature for other difficult robot models [2].

The paper is organized as follows: In Section 2, the motion dynamics for our bi-rotor model is derived including the derivation for a tail appendage. Section 3 continues with detailing the conventional control-theoretic method for the bi-rotor system. Section 4 details the Deep Reinforcement Learning based control of the platform. Experimental setup and comparative results are presented in Section 5. Finally, Section 6 concludes by summarising our main results and possible directions for future research.

2 Dynamics of the Bi-rotor System

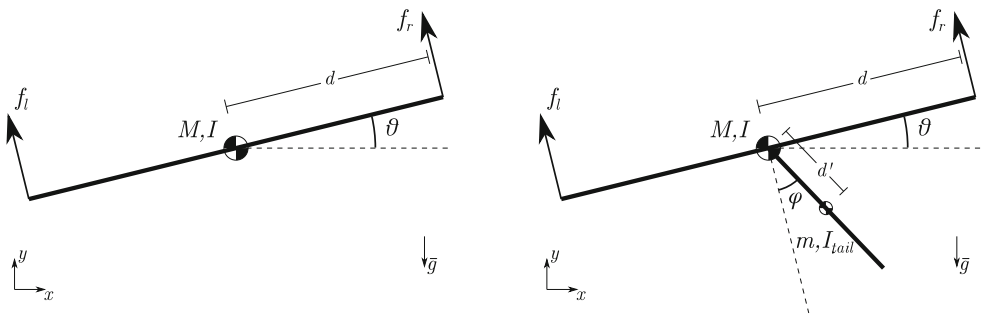
In this section, we first present the dynamics of a bi-rotor with and without tail appendage. We define the state, $\mathbf{x}$, and the input, $\mathbf{u}$, of the bi-rotor system as:

$$\mathbf{x} = \left[p_x, p_y, \vartheta, v_x, v_y, \omega\right]^T , \tag{1}$$

and,

$$\mathbf{u} = [f_l, f_r]^T , \tag{2}$$

where p_x, p_y, ϑ are respectively the position of bi-rotor in Cartesian coordinates as shown in Fig. 1a and the orientation with respect to the horizontal axis; v_x, v_y, ω are velocities; and, f_l, f_r are respectively thrust inputs to left and right rotors.



(a) The bi-rotor model without tail

(b) The bi-rotor model with tail

Fig. 1 The bi-rotor model considered in this paper, with and without tail appendage. x and y are dimensions in 2D Cartesian plane. $\vec{g}$ is gravitational acceleration. f_r and f_l are force inputs to right and left rotors. M and m are mass of bi-rotor body and tail appendage. I and I_{tail} are inertia of bi-rotor body and tail appendage. d and d' are half length of bi-rotor body and tail appendage. ϑ is orientation of bi-rotor body with respect to world coordinates. φ is orientation of tail appendage with respect to bi-rotor body

Then we can derive the dynamics of the bi-rotor system as follows in a non-linear state-space form:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\sin(\vartheta)/M & -\sin(\vartheta)/M \\ \cos(\vartheta)/M & \cos(\vartheta)/M \\ -d/I & d/I \end{bmatrix} \mathbf{u} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -g \\ 0 \end{bmatrix}, \tag{3}$$

where d is half length of the bi-rotor body; g is the magnitude of gravitational acceleration in negative y-direction; M and $I = \frac{1}{3}Md^2$ are mass and moment of inertia of the bi-rotor body (see Fig. 1a). The output of the system is considered as the positions:

$$\mathbf{y} = \begin{bmatrix} p_x, p_y \end{bmatrix}^T. \tag{4}$$

Additionally, we consider the dynamics of the propellers. Instead of supplying commanded thrust forces instantaneously, a simple mathematical motor model is implemented. This is not an actual motor model, however, it implements a first-order discrete characteristics for the motor speed to reach the desired point with a delay. This is a similar model with a previous work [33] that studies it on a quad-rotor UAV, as propeller speed can be modeled in first-order dynamics [8]. The thrust forces of propellers according to their motor speeds are defined as,

$$f_l = f_{max}\omega_{m,l}^2, \tag{5}$$

and,

$$f_r = f_{max}\omega_{m,r}^2, \tag{6}$$

where $\omega_{m,l}, \omega_{m,r} \in [0, 1]$ represent the normalized motor velocities. Then, the motor velocity commands are derived from the normalized actions as,

$$\omega'_{m,l} = \sqrt{\frac{f'_l + 1}{2}}, \tag{7}$$

and,

$$\omega'_{m,r} = \sqrt{\frac{f'_r + 1}{2}}, \tag{8}$$

where $\omega'_{m,l}, \omega'_{m,r} \in [0, 1]$ are motor commands. Finally, motor velocities are updated through a discrete-time low-pass filter as:

$$\omega_{m,t} = \frac{4\Delta t}{t_s}(\omega'_{m,t} - \omega_{m,t-1}) + \omega_{m,t-1}, \tag{9}$$

where $\omega_{m,t-1}$ is motor speed at time $t-1$, $\omega'_{m,t}$ is the motor command, $\omega_{m,t}$ is the motor speed at time t and t_s is the %2 settling time of low pass filter.

Tail Appendage As illustrated in Fig. 1b, we consider the tail as an additional rigid body connected to the center of bi-rotor body with an actuated revolute joint. For this, the state vector is now extended with orientation and angular velocity of tail with respect to the bi-rotor body as,

$$\mathbf{x}^{\sim} = \begin{bmatrix} p_x, p_y, \vartheta, v_x, v_y, \omega, \varphi, \dot{\varphi} \end{bmatrix}^T, \tag{10}$$

where φ represents the tail angle with respect to main body as seen in Fig. 1b. The input vector is extended with torque input of the revolute joint as,

$$\mathbf{u}^{\sim} = [f_l, f_r, \tau_{tail}]^T, \tag{11}$$

where τ_{tail} represents the torque applied to the revolute joint.

The dynamics of tail assisted bi-rotor system can be derived as,

$$\mathbf{M}\ddot{\mathbf{q}} = \begin{bmatrix} -(f_l + f_r)\sin(\vartheta) + md'(\dot{\vartheta} + \dot{\varphi})^2 \sin(\vartheta + \varphi) \\ (f_l + f_r)\cos(\vartheta) - md'(\dot{\vartheta} + \dot{\varphi})^2 \cos(\vartheta + \varphi) \\ d(f_r - f_l) - mgd' \sin(\vartheta + \varphi) \\ \tau_{tail} - mgd' \sin(\vartheta + \varphi) \end{bmatrix}, \tag{12}$$

where d' and m are the half length and mass of tail appendage, $\mathbf{q} = [p_x, p_y, \vartheta, \varphi]^T$ is the generalized coordinates of the system, and,

$$\mathbf{M} = \begin{bmatrix} M+m & 0 & md'\cos(\psi) & md'\cos(\psi) \\ 0 & M+m & md'\sin(\psi) & md'\sin(\psi) \\ md'\cos(\psi) & md'\sin(\psi) & I + I_{tail} + md'^2 & I_{tail} + md'^2 \\ md'\cos(\psi) & md'\sin(\psi) & I_{tail} + md'^2 & I_{tail} + md'^2 \end{bmatrix}, \tag{13}$$

is the inertia matrix with $\psi = \vartheta + \varphi$. $I_{tail} = \frac{1}{3}md'^2$ is the moment of inertia of the tail appendage. Note that the determinant of the inertia matrix is given by,

$$\det(\mathbf{M}) = I(M+m)(I_{tail}(M+m) + Mmd'^2), \tag{14}$$

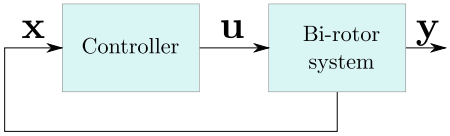
and is strictly positive. Therefore, the inertia matrix $\mathbf{M}$ is invertible.

In Eq. 12, multiplying both sides by $\mathbf{M}^{-1}$, we get the third row as,

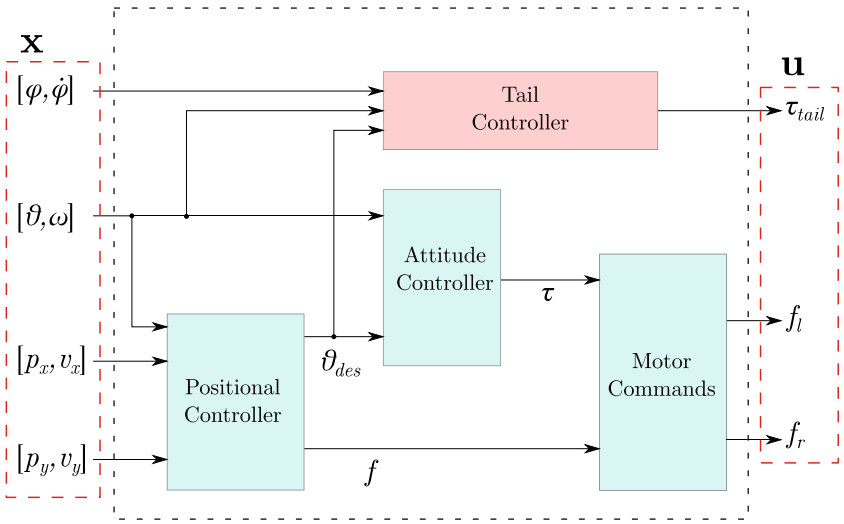
$$\ddot{\vartheta} = \frac{d(f_r - f_l) - \tau_{tail}}{I}, \tag{15}$$

which gives the angular dynamics of the bi-rotor body. This equation formulates that the torque applied to the tail is linearly related with the angular position of the bi-rotor. Therefore, the tail appendage should help the angular positioning of the bi-rotor. This property is used while designing the conventional controller (see Section 3).

We consider state feedback control for the bi-rotor system as shown in Fig. 2a. The controller takes the state information of the bi-rotor and provides required inputs for the system. The conventional controller implemented in this block is explained in Section 3. In the learning-based control, this block is implemented by the actor/policy neural network which is explained in Section 4.



(a) The block diagram of a state feedback controller for the bi-rotor system. The controller is implemented either as the conventional controller or as the policy neural network.



(b) The block diagram of conventional controller. The red block, *tail controller*, is used in case of tail appendage

Fig. 2 **a** The block diagrams of overall state feedback controller scheme used in this research. **b** Internal controller block for the conventional controller

3 Conventional Control of the Bi-rotor System

In Fig. 2b, the block diagram of the conventional controller is given. The complete diagram takes the terms of state vector as inputs and provides the signals for controlling the bi-rotor. This controller structure, separating attitude control and positional control, is inspired from the multi-rotor control literature [8, 40], where the asymptotic stability of the method is also shown.

There are four main blocks in Fig. 2b. The red block is responsible for tail appendage and will be explained at the end. The remaining three blocks are for generating the left and right rotor thrust commands. Those thrust forces can be expressed as an equivalent torque, τ, and force f as,

$$\tau = (f_r - f_l) \cdot d, \tag{16}$$

$$f = f_r + f_l. \tag{17}$$

Combining Eqs. 16 and 17, we get,

$$\begin{aligned} f_r &= \frac{f}{2} + \frac{\tau}{2d}, \\ f_l &= \frac{f}{2} - \frac{\tau}{2d}. \end{aligned} \tag{18}$$

Hence, we can divide the control task into position and attitude control parts by expressing input propeller thrust forces as a torque as well as a thrust force. The *Motor Commands* block simply implements this conversion using Eq. 18.

In our work, two Proportional-Derivative (PD) controllers are used for position and attitude control. In the *Position Controller* block, the desired horizontal and vertical forces to control the cartesian position are calculated using PD-control as,

$$f_{x,des} = -(K_{p,x} p_x + K_{d,x} v_x), \tag{19}$$

$$f_{y,des} = Mg - (K_{p,y} p_y + K_{d,y} v_y), \tag{20}$$

where $K_{p,x}$, $K_{d,x}$, $K_{p,y}$ and $K_{d,y}$ are controller parameters for the two controllers. The PD-controller expressions convert the positions errors into required force components in both directions. Those force components cannot be satisfied simultaneously because the system is under-actuated. Therefore, the thrust force f is determined such that its vertical component corresponds to the desired vertical force effecting platform altitude, namely,

$$f = f_{y,des} \frac{1}{\cos(\vartheta)}. \tag{21}$$

The horizontal force vector for position control is achieved through regulating the platform attitude. With the appropriate attitude, the same thrust force also generates a horizontal component serving also x-axis position control. However while determining the desired attitude, we should also be aware of its negative effect on the y-axis lifting component. In practice large x-axis thrust component (through a large attitude angle) is not feasible because the motor thrusts may not be sufficient to maintain the required vertical thrust component.

Incorporating these constraints, the desired attitude to be tracked by the attitude controller is therefore given by,

$$\vartheta_{des} = \text{clip}\left(\text{atan2}(f_{x,des}, f_{y,des}), \{\vartheta_{min}, \vartheta_{max}\}\right), \tag{22}$$

where atan2$(\cdot,\cdot)$ is the two-argument arc-tangent function while the clip$(\cdot,\cdot)$ function is defined by:

$$\text{clip}\left(\vartheta, \{\vartheta_{min}, \vartheta_{max}\}\right) = \begin{cases} \vartheta_{min}, & \text{if } \vartheta \le \vartheta_{min}, \\ \vartheta, & \text{if } \vartheta_{min} < \vartheta < \vartheta_{max}, \\ \vartheta_{max}, & \text{if } \vartheta_{max} \le \vartheta. \end{cases} \tag{23}$$

The clipping interval is a design parameter determined by considering the thrust-to-weight ratio of the bi-rotor so that the bi-rotor can supply enough thrust to at least keep itself hovering on y-axis at these limiting attitudes. It also prevents Eq. 21 to be singular by assuring the attitude not to be right angle. In general, we have $\vartheta_{min} = -\vartheta_{max}$.

The *Attitude Controller* block is also a PD-controller that regulates the attitude of the bi-rotor by producing a torque component expressed by

$$\tau = (K_{p,\vartheta}(\vartheta_{des} - \vartheta) + K_{d,\vartheta}(-\omega)), \tag{24}$$

which is then transformed back into propeller forces (Eq. 18). Here, $K_{p,\vartheta}$ and $K_{d,\vartheta}$ are again the PD-controller parameters.

3.1 Tail-assisted Attitude Control

The tail actuator is used to generate additional torque to help with attitude control. Equation 15 illustrates that the torque applied to the tail motor can be directly incorporated into the angular dynamics of the bi-rotor. However, the tail, besides generating an assist torque, also needs to stay mostly in a vertical position so as not to further interact with the attitude control problem. Hence the tail controller (shaded red in Fig. 2b) is composed of two PD-control terms, one for bi-rotor attitude control and a second one for compensating tail angle with the vertical. These terms are given by

$$\tau_{tail1} = -(K_{p,\vartheta,tail}(\vartheta_{des} - \vartheta) + K_{d,\vartheta,tail}(-\omega)), \tag{25}$$

$$\tau_{tail2} = (K_{p,\varphi}(-\vartheta_{des} - \varphi) + K_{d,\varphi}(-\dot{\varphi})), \tag{26}$$

where K_p and K_d parameters are the corresponding controller parameters for each term. The combined torque given by,

$$\tau_{tail} = \tau_{tail1} + \tau_{tail2}, \tag{27}$$

is supplied to the tail motor which is assumed to be torque (current) controlled.

4 Deep Reinforcement Learning Based Control of the Bi-rotor System

As the learning-based control approach, we adopted Deep Reinforcement Learning (DRL), a paradigm suitable especially for learning from interactions with an environment [39]. In this section, we first provide a short background on DRL and then describe how we adapted DRL for bi-rotor control.

4.1 Background on Deep Reinforcement Learning

In RL, the environment is considered as a Markov Decision Process (MDP) that can be characterized by a five-element tuple: $(\mathcal{S}, \mathcal{A}, P, R, \rho_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the sets of all possible states and actions; P, R and ρ_0 are the transition, the reward and the initial state distribution functions. The state transitions are shaped by a transition function denoted P:

$$P(s'|s,a) = Pr\{s_{t+1} = s'|s_t = s, a_t = a\}, \tag{28}$$

where $s, s' \in \mathcal{S}$ are the current and next states, $a \in \mathcal{A}$ is the action, and t denotes the discrete time-step. This equation incorporates the *Markovian Property* of MDPs: The next state depends on only the current state, and not on the past ones. That is the dual of state-space representation of a system in control theory where the state vector can represent the future dynamics of a system at an instant.

The reward function $R(s,a)$ provides a scalar reward depending on the current state of the MDP. In some environments, the reward signal might be *sparse*, and therefore, *reward shaping* might be required to provide a reward for intermediate states to guide the agent better towards goal states. The objective of the agent is to *find* a policy that maximizes the rewards collected in that environment.

4.2 Our Deep RL Approach for Bi-rotor Control

In order to formulate our bi-rotor dynamical system as an MDP, we use the state vector, $\mathbf{x}$, and the input vector, $\mathbf{u}$, of the bi-rotor system as the states and the actions of the MDP. Here, the state and action sets are in continuous real number

domain that yields infinite number of states and actions. That is an extension to the classical MDPs that are considered finite. The state transitions of MDP happen according to the dynamics given in Eq. 3 with time discretization, noting that a state-space dynamics equation is the dual of *Markovian Property* in the field of control theory.

In this work, we adopt the Proximal Policy Optimization (PPO) method [38] since, as reviewed in Section 1.1, PPO provides state-of-the-art results in various control problems. We use the implementation based on *stable-baselines* [20] package. In PPO, we have a neural network that is employed as a policy and a value function approximator. The policy network takes the current state of MDP as input and outputs the action. The value network has the same network structure of policy network except it outputs single scalar value approximation of current state.

As the policy network, we employ a Multilayer Perceptron (MLP) illustrated in Fig. 3 together with the critic network. The input layer has size $n_s = 6$ or $n_s = 8$ according to tail usage. There are two hidden layers of H neurons with *tanh* non-linearity for both networks. Finally, the output layer of the policy network provides $n_a = 2$ or $n_a = 3$ dimensional action vector with *tanh* non-linearity. We use *tanh* in order to get normalized actions between $[-1, +1]$. In addition to that a similar network architecture was employed for quad-rotor control task at a previous work [33] we examine the effect of different hidden neuron numbers also.

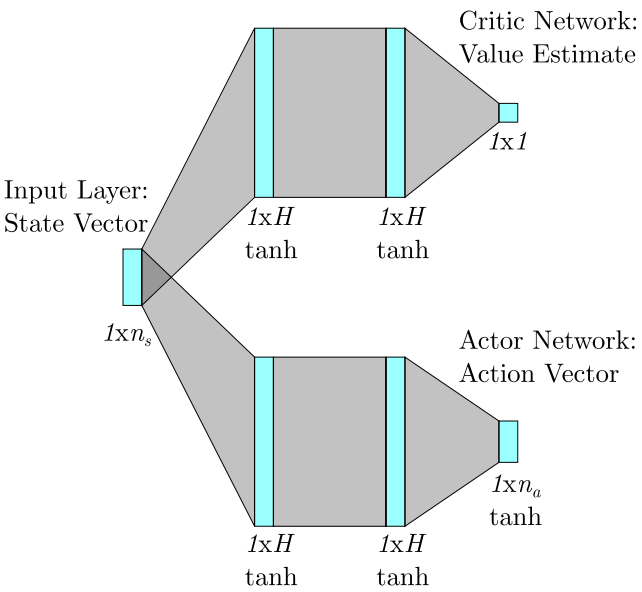


Fig. 3 Actor-critic network structure. **Critic network:** A fully connected neural network, input is state vector, two hidden layers with tanh activated H neurons and output is state-value estimation. **Actor (Policy) network:** A fully connected neural network, input is state vector, two hidden layers with tanh activated H neurons and output is action

4.3 Reward Shaping

In many environments, rewards are rather sparse. Reward shaping is the mechanism of propagating sparse reward to the state space. In our work, we use a shaped reward similar to previous studies on quad-rotors [22, 33]. Reward at a time step is calculated for the bi-rotor without tail-appendage as:

$$r_t = 25 - \left(k_p \|p_t\| + k_v \|v_t\| + k_\vartheta \|\vartheta_t\| + k_\omega \|\omega_t\| + k_a \|a_t - a_h\|\right), \tag{29}$$

where r_t denotes the reward at time step t; p_t is the position vector $[p_x \;\; p_y]^T$; v_t is the velocity vector $[v_x \;\; v_y]^T$; a_t denotes the action vector; a_h is hovering action; and k's are constant parameters. $||\cdot||$ denotes $L1$ norm.

With this definition, the agent can receive a maximum reward of 25, and the rewards decreases by the distance from the goal state-action configuration. Similarly, reward for the bi-rotor with tail-appendage is defined as:

$$\tilde{r_t} = r_t - (k_\varphi \|\varphi_t\| + k_{\dot{\varphi}} \|\dot{\varphi}_t\|), \tag{30}$$

which punishes tail angle and velocity in addition to penalties of the non-tail case in Eq. 29.

Parameters of reward formulation are given in Table 1c. Here, main contribution to reward is coming from positional errors, since the main task is position control. The costs on linear velocities are similar to a derivative term of PID controller; it becomes significant after position becomes small and prevents the agent to oscillate around goal point. It can be noted that angular position or non-sense actions are also penalized in order to prevent the bi-rotor making angular oscillations around goal point without changing its position.

4.4 Training Strategies

We evaluated two different training strategies based on initial state distribution of the episodes: vanilla training and curricular training. In vanilla training, the whole state space is randomly explored throughout training, similar to e.g. [22, 33], whereas, in curricular training, the state space is gradually expanded to start learning first from "easy" (close-by) states and then from "hard" (far-away) states. As it will be detailed under Section 5.1, we initiate episodes randomly by a uniform distribution over position variables, where we control lower and higher boundaries of distance in both directions.

Vanilla Training Formally, we assign the boundaries of the random initialization as,

$$p_x^h = p_y^h = p_{max}, \tag{31}$$

Table 1 System parameter values in the paper

Parameters	Bi-rotor	Bi-rotor with tail
(a) Parameters of the simulation environment (Fig. 1, Eq. 9)		
g	9.81	9.81
M	0.8	0.4
m	0	0.4
d	0.5	0.5
d'	–	0.25
Δt	0.01	0.01
t_s	0.15	0.15
(b) Optimized controller parameters of our conventional controller (Eqs. 19, 20, 24, 25 and 26)		
$K_{p,x}$	3.23	3.42
$K_{d,x}$	2.57	2.78
$K_{p,y}$	4.45	5.18
$K_{d,y}$	2.89	3.18
$K_{p,\vartheta}$	11.79	4.40
$K_{d,\vartheta}$	2.40	1.11
$K_{p,\vartheta,tail}$	–	3.43
$K_{d,\vartheta,tail}$	–	0.93
$K_{p,\varphi}$	–	−0.80
$K_{d,\varphi}$	–	1.00
(c) Parameters of the shaped reward for the bi-rotor with and without tail appendage (Eqs. 29 and 30)		
k_p	20	20
k_v	2	2
k_θ	1	4
k_ω	0	0
k_a	1	1
k_φ	–	2
$k_{\dot{\varphi}}$	–	0.2

where p_x^h and p_y^h are the high boundaries of the uniform distribution; p_{max} is a parameter controlling the extent of the space. Here, the distribution covers a square box centered at origin. The algorithm runs 10^7 time-steps and saves the best network parameters during training based on total reward collected for a number of previous episodes. Additionally, the same reward is used to compare the behaviour of both controllers.

As an additional case, the task is simplified to fixed horizontal length called fixed-point-to-point training. By sampling only single starting point initially, the agent becomes more successful when initialized from that point but does not give guarantee to converge from other positions. Now, the constraints are assigned as,

$$(p_x, p_y) = (\pm p_{p2p}, 0), \tag{32}$$

where p_{p2p} is the fixed length that the agent learns to overcome. Here, the distribution is restricted to two points at a fixed horizontal location from the left and right of the goal point $(0, 0)$.

Curriculum Training As an alternative training strategy we use a curriculum that is increasing the hardness of task gradually. We control p_{max} parameter in Eq. 31 for adjusting hardness of task during training. At first the bi-rotor is trained to hover for 5×10^6 time-steps. Formally, we define the initial position limit of *stage 0* as,

$$p_{max,0} = 0. \tag{33}$$

Then the best network parameter is found similar with vanilla case and is transferred to the next stage. At each stage initial position limit value is incremented as,

$$p_{max,n} = 0.5 + 0.5 \times n, \tag{34}$$

for *stage* n. 1.25×10^6 time-steps are given for training of each stage.

5 Experiments and Results

In this section, we describe our experimental setup and present the results of our extensive experiments where

we compare the conventional control and reinforcement learning based control for a bi-rotor platform with and without a tail appendage. We first explain the design of the simulation environment including the used tools and parameter selection. Next, we provide the optimization details of both controllers used in this study. Then, we conduct three experiments. In the first experiment, we analyze the average behavior of the controllers as well as the contribution of a tail appendage by deploying them from different initial states. The second experiment digs deeper and presents the results for a single horizontal movement of platforms. In the last experiment, we focus on the learning-based controller and propose a curriculum strategy for training.

5.1 The Simulation Environment

We designed a simulation environment to solve the bi-rotor dynamics with and without a tail as given in Eqs. 3 and 12. The simulation environment receives an action (control) signal a_t at time step t, integrates the bi-rotor dynamics for a small time duration Δt, calculates the next state s_{t+1} and the reward signal r_{t+1}. The state-transition dynamics from s_t to s_{t+1} under action a_t are as detailed in Section 2. The reward signal is determined according to the current state distance from the goal point as explained in Section 4.2.

The simulation environment is implemented in Python using the Box2D physics engine [11] to simulate the dynamics of the bi-rotor. Box2D is an open-source engine that solves the differential equations of rigid multi-body systems in a planar environment. We also used OpenAI's "Gym Toolkit" [10] to integrate Deep Reinforcement Learning algorithms with the simulation environment. The relevant parameters of this simulation environment are given in Table 1a.

Random Initial State Distribution. For training and testing the system, unless otherwise stated, the initial position of each episode is sampled from a uniform distribution as follows:

$$p_x \sim \mathcal{U}(-p_x^h, p_x^h), \tag{35}$$

$$p_y \sim \mathcal{U}(-p_y^h, p_y^h), \tag{36}$$

where $p_x^h, p_y^h \in [0, p_{max})$, such that $p_{max} < 6$, are parameters representing lower and higher bounds of the uniform distribution. Additionally, the velocity of the platform is also initialized uniformly in a small interval for all cases as follows:

$$v_x \sim \mathcal{U}(-0.2, 0.2), \tag{37}$$

$$v_y \sim \mathcal{U}(-0.2, 0.2). \tag{38}$$

The other state information, such as orientation and angular velocity of body or tail, are initiated as zero.

Termination Criteria. An episode is terminated in two situations: The first is when the bi-rotor leaves a large bounding region, namely a $10m \times 10m$ square, around the goal location, i.e. when:

$$\max(p_x, p_y) > 5m. \tag{39}$$

Note that this bounding region is larger than the simulation region ($p_{max} \times p_{max}$). This criterion allows the agent to ignore irrelevant regions -more than 5m distant from the goal position- in order to speed up training by reducing simulation time and eliminating irrelevant data samples.

The second termination criterion pertains to the time spent on an episode: We set a time limit of 4 seconds (400 time steps). After 4 seconds, a new episode starts with the bi-rotor being initialized from another state, which encourages exploration of other states in region.

5.2 Training and Parameter Optimization

Conventional controller. Parameters of the conventional controller are optimized using both global and local optimization techniques for the cases with and without tail. Particle Swarm Optimization (PSO) [15] is employed as the global optimizer, and in order to fine-tune the local minimum, the Nelder-Mead method [17] is used. We use the reward functions in Eqs. 29 and 30 in the optimization cost function to have a fair comparison with the Deep Reinforcement Learning solution. Specifically, the cost function is defined as the total reward collected through 20 random starting positions. The resulting optimized parameter values are listed in Table 1a. Note that the proportional term of the tail stabilizer ($K_{p,\varphi}$) in Eq. 26 turns out to be negative, which, however, does not cause instability thanks to the derivative term.

Details of the learning-based controller. The network architecture for the learning-based controller is optimized for the number of hidden neurons (H in Fig. 3) in the two hidden layers. This optimization is conducted with six different values of $H \in [16, 128]$. For each H value, five training runs each with 1.5×10^7 time steps is conducted. In these training runs, the agent is trained under the vanilla learning within a $6m \times 6m$ simulation arena box as in Section 5.3. The average of the acquired maximum rewards in consecutive 20 episodes during training is computed and tabulated in Table 2a. The table suggests that there is a dramatic increase in performance up to $H = 64$ after which the performance is no longer improved. Therefore, we use $H = 64$ for the rest of our experiments.

Table 2 An analysis of the effect of the number of hidden neurons (**a**) and hidden layers (**b**) on the performance. Average of maximum reward collected during five training-runs is reported

(a) Effect of the number hidden neurons (H) per hidden layer

# neurons (H)	16	32	48	64	80	128
Avg. Max Reward	-6703	-3317	-162	4487	4330	-1467

(b) Effect of the different number hidden layers (L)

# layers (L)	2	3
Avg. Max Reward	4487	4054

Another design parameter we have experimented with is the number of hidden layers, L, in the architecture. Assuming $H = 64$, changing L from two resulted in performance drop as presented in Table 2b. Hence, we have proceeded with $L = 2$ hidden layers for the rest of our experiments.

5.3 Experiment 1: Comparison through 50 Random Initial States

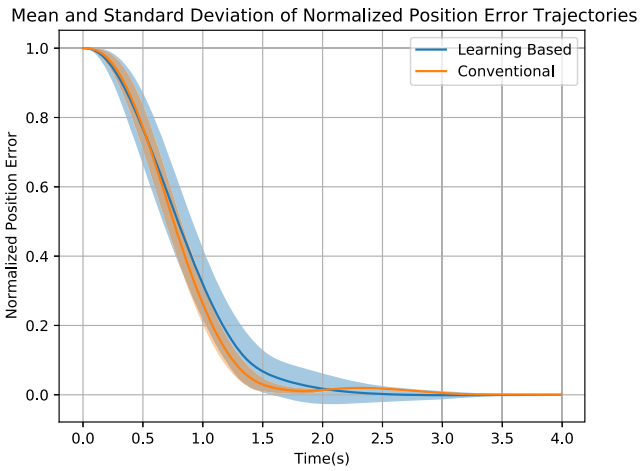
In this experiment, the conventional and learning-based controllers are compared based on their average performance over 50 scenario episodes. Here, the learning-based controllers are trained with vanilla training with $p_{max} = 3$ (Section 4.4). All controllers with and without tail are started from 50 randomly determined initial state points drawn from a uniform distribution. Positional errors are analyzed through those 50 episodes. To be able to compare episodes with close and far starting points, we normalize positional error for each episode to be within the [0,1] range, such that the error at $t = 0$ is scaled to one and the error at final is scaled to zero. The evolution of the average error as well as the standard deviation bands as a function of the simulation time is shown in Fig. 4. These ensemble average curves are the result of 50 simulation runs.

In Fig. 4a, normalized position errors are plotted for bi-rotor without tail under both controllers. We observe that both approaches converge to the goal under two seconds. This result can also be considered an experimental proof of stability and convergence of the learning-based controller, although mathematical analysis is very hard on a neural network policy. We notice that the learning-based controller's mean behaviour stabilizes at the target later than the conventional controller (see also Table 3) and it has higher standard deviation.

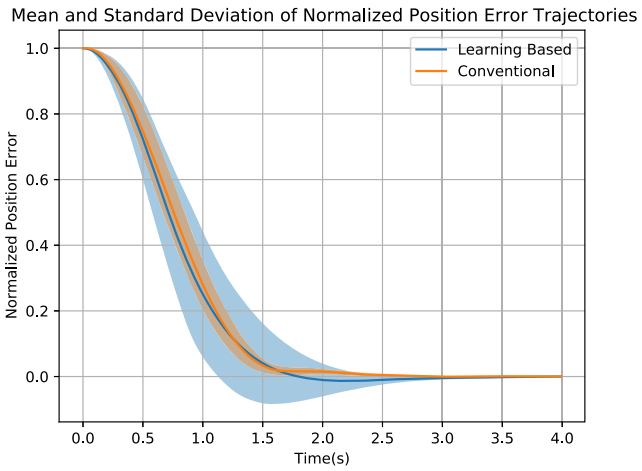
A similar analysis for the tail appendage is given in Fig. 4b. Here we observe that the mean behavior of the learning-based controller approximates the conventional controller better and they have identical settling times (see also Table 3). However, the standard deviation for the learning-based controller is even higher.

5.4 Experiment 2: Comparison from a fixed-state

In this experiment, sample runs of bi-rotor with and without tail are analyzed in detail. In each case, the bi-rotor is initiated from $(p_x, p_y) = (2, 0)$ and observed for 4 seconds. The results are plotted in Fig. 5 providing a visual



(a) Normalized position errors without tail.



(b) Normalized position errors with tail.

Fig. 4 Mean and standard deviation plots of normalized position errors of 50 episodes under conventional and learning-based controller with and without tail

Table 3 Mean settling time of conventional and learning based controllers with and without tail

Controller	2% Settling time(s)
Conventional without tail	1.59
Learning based without tail	1.95
Conventional with tail	1.60
Learning based with tail	1.61

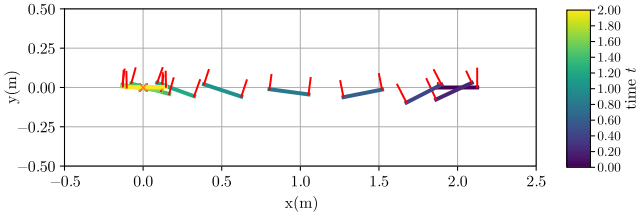
illustration of the overall movement for first 2 seconds as snapshots along the trajectory and in Fig. 6 showing the transition of all state and action terms. In this subsection, we give a comparison of conventional and learning-based control for the bi-rotor without and with tail individually. When the tail usage is compared, we observe that the use of tail decreases the torque demand from rotors as expected.

The learning curves with and without tail cases are plotted in Fig. 7. Although these learning curves depict how the policy learns to collect higher rewards, they do not directly show how the policy will run or what will be the settling time. Nevertheless, the training of the bi-rotor has a similar curve with or without tail which shows that the rise in robot complexity will not increase the complexity in the configuration of the learning-based controller, unlike the conventional controller.

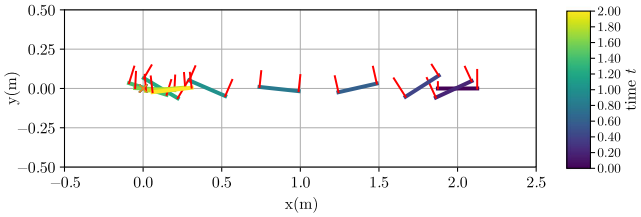
Bi-rotor without Tail. Trajectories obtained from the bi-rotor without tail are shown at the top two rows of Fig. 6. For conventional controller, the p_x is settled in 1.42 seconds considering 2% settling time, and a slight change along y is observed due to the attitude (θ) change of the bi-rotor as can be also seen in Fig. 5. All states asymptotically converged to zero (the goal point). The learning-based controller shows a similar converge speed except for p_x which converges slightly faster. However, there remains nonzero steady-state error around zero. These steady-state errors are tabulated in Table 4.

Attitude change of the bi-rotor is given at the center column, in Fig. 6b and e. In both cases, the bi-rotor first tilts in the positive θ direction to accelerate in the negative x-direction and later the negative θ direction to slow down as shown in Fig. 5. However, the changes in angular velocity is more aggressive than the conventional case. We speculate that the neural network can encode a higher-order non-linearity and the policy can make more radical changes at each time-step.

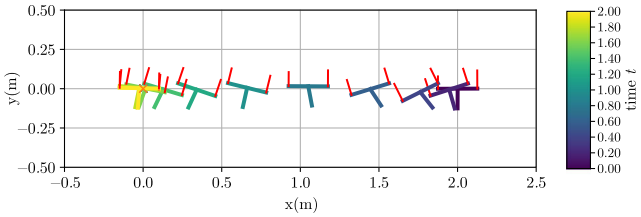
Finally, force commands are plotted in Fig. 6c and f. Initially, the rotors behave to obtain an attitude change followed by convergence to hovering behavior. Notably, changes in motor commands for the learning-based controller are also more aggressive than for the conventional controller. However, in our scenario, this seems like an unnecessary effort in completing the job.



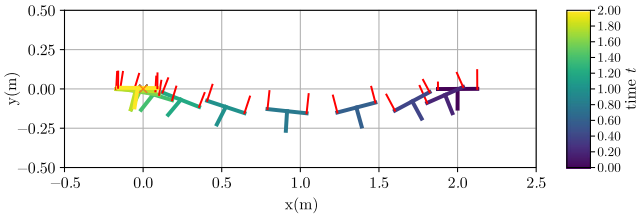
(a) Conventional controller without tail



(b) Learning-based controller without tail



(c) Conventional controller with tail



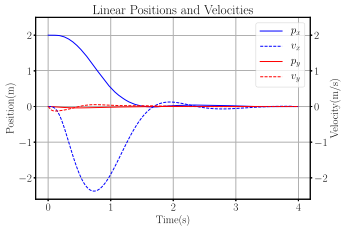
(d) Learning-based controller with tail

Fig. 5 Comparative visual illustration of the four studied scenarios as snapshots along the trajectory. Platform and tail poses and thrust force vectors are shown. Color-scale at the right represents evolution of time. Red lines perpendicular to the bi-rotor body in each case illustrate the relative magnitudes of the rotor thrusts. Bi-rotor size is scaled down to improve the clarity of the motion. As the time axis illustrates the motion is from right to left

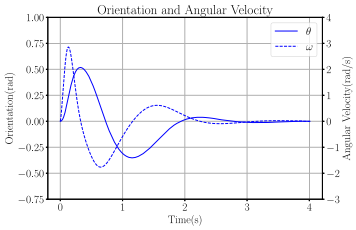
Bi-rotor with Tail. Trajectories obtained from the bi-rotor without tail are shown at the bottom two rows of Fig. 6. In Fig. 6g and j, positions and linear velocities are plotted. For the conventional controller, the settling time of p_x is 1.55 seconds, slightly worse than the no-tail case, and although v_x starts decreasing sharper than in the no-tail case, it finally shows a similar performance. On the other hand, the learning-based controller has a rapid initial acceleration in p_x compared to both no-tail and tail

Bi-rotor without tail

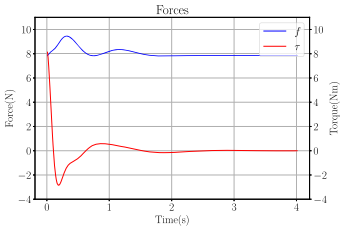
Conventional controller



(a) Linear position and velocity.

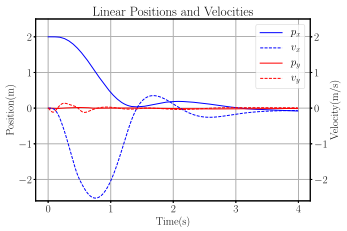


(b) Orientation and angular velocity.

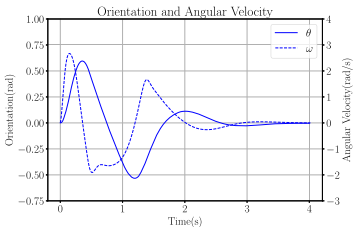


(c) Total thrust force and effective torque.

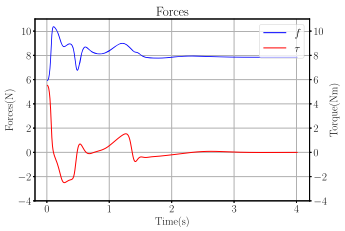
Learning-based controller



(d) Linear position and velocity.



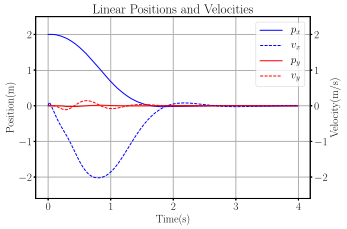
(e) Orientation and angular velocity.



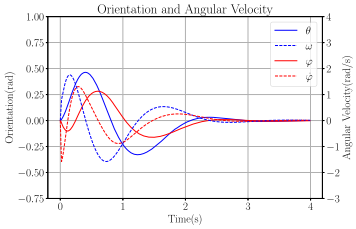
(f) Total thrust force and effective torque.

Bi-rotor with tail

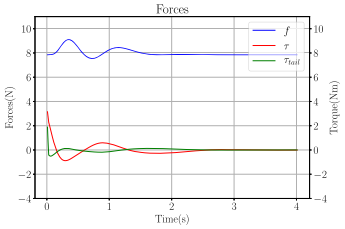
Conventional controller



(g) Linear position and velocity.

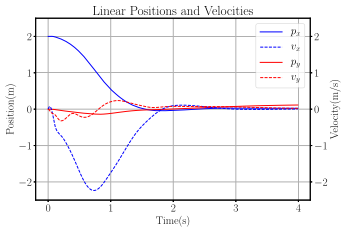


(h) Orientation and angular velocity.

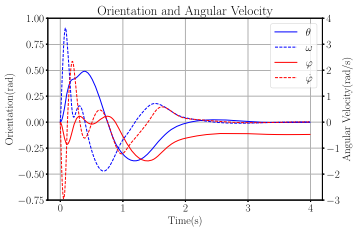


(i) Total thrust force, effective torque and tail torque.

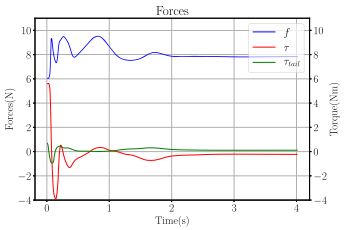
Learning-based controller



(j) Linear position and velocity.

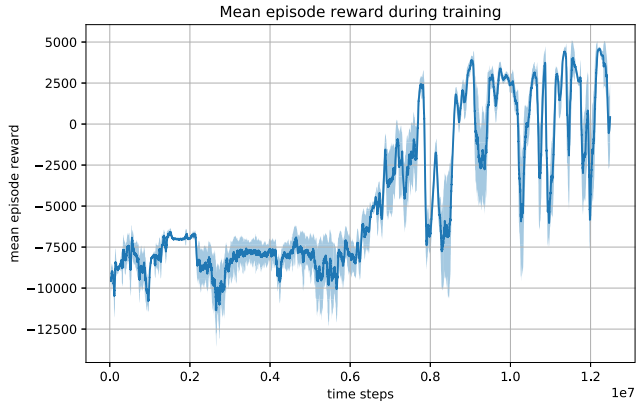


(k) Orientation and angular velocity.

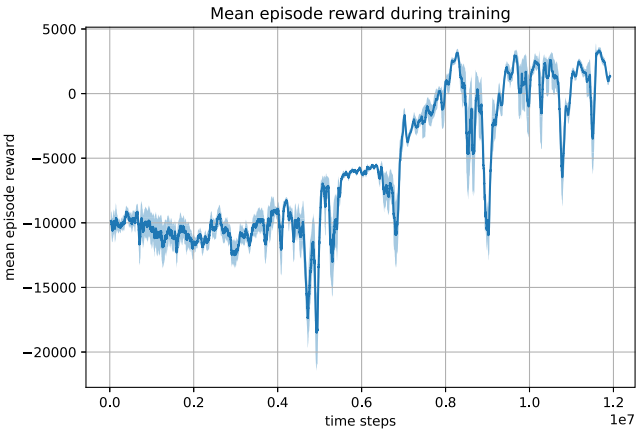


(l) Total thrust force, effective torque and tail torque.

Fig. 6 States and input vs time for the initial 4 seconds of the simulation. The bi-rotor behavior for both of the controllers and both with and without tail are illustrated. From top to bottom: conventional control without tail, learning-based control without tail, conventional control with tail and, learning-based control with tail



(a) The bi-rotor without tail.



(b) The bi-rotor with tail.

Fig. 7 Mean episode reward collected during point-to-point training of the bi-rotor with and without tail. Blue line represents average episode reward in last 200 episodes, and light blue regions are their standard deviation

with the conventional controller cases. 2% settling time is 1.47s and better than the conventional controller.

Orientation and angular velocities of the body and the tail are plotted in Fig. 6h and k. Body attitude and angular velocity exhibit a very similar pattern with the no-tail case. Tail moves as expected to help the body attitude change faster in both conventional and learning based controller cases. Note also that these behaviours, such as the negative peaks of tail angle observed at time 1.5s as can be also seen in Fig. 5, helps the robot to orient back in equilibrium by the tail's gravitational pull. In other words, the gravitational force of the tail is also useful to rotate the body since it can shift the center of gravity of the bi-rotor. The bi-rotor can gain angular acceleration with a shift in the center of gravity even if the rotor thrusts are kept constant. For the learning-based case, again rapid changes are similarly observed. Moreover, the tail orientation settles with a steady state error (Table 4).

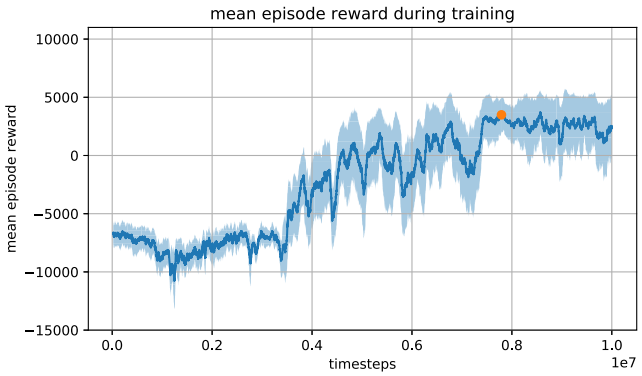
Table 4 Steady state errors of point-to-point trained control policies

Steady state error	Bi-rotor w/o tail	Bi-rotor with tail
p_x	0.146	0.032
p_y	0.010	0.143
ϑ	0	0
φ	–	0.113

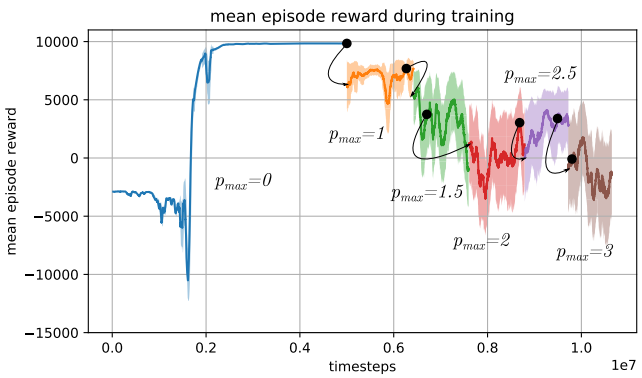
Finally, the command forces are drawn in Fig. 6i and l. Notably, the torque due to the rotors are nearly half way down compared to no-tail case with the help of the tail for the conventional case. Again, sudden unnecessary changes are observed in the learning-based controller.

5.5 Experiment 3: Curriculum Learning vs. Vanilla Learning

In this experiment we compare curriculum learning with the vanilla training. The agents are trained with a limit of $p_{max} = 3$ for the vanilla case. For curriculum training, six stages are used as explained in Section 4.4. The final stage

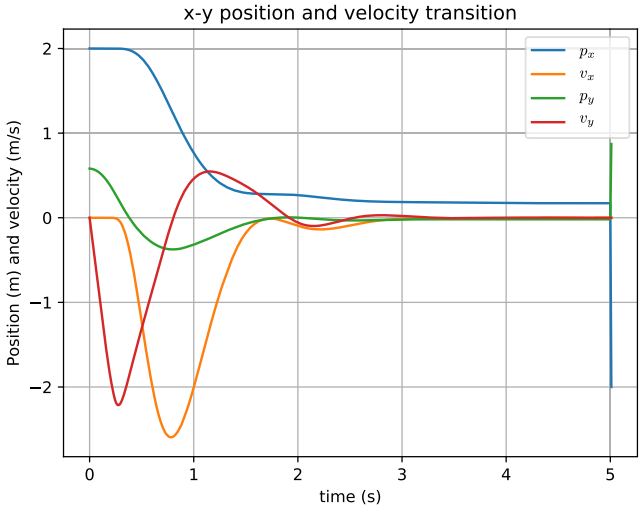


(a) Vanilla training.

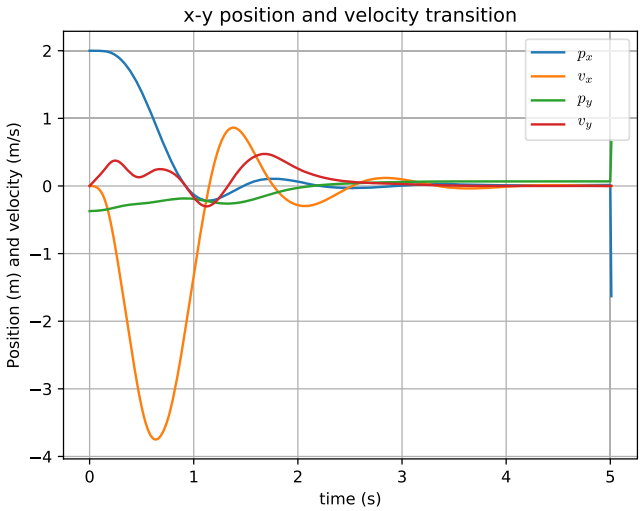


(b) Curriculum training.

Fig. 8 Mean reward collected by agent for both training strategies. In curriculum learning, each color represents different stage. Black dots in each stage are instance of best performing network weights, which are transferred to the next stage



(a) Vanilla training



(b) Curriculum training

Fig. 9 Test runs for policy learned with curriculum training and vanilla training

of the curriculum case covers the same area with the vanilla case.

The mean rewards collected in the last 200 episodes during vanilla training and curriculum training are given in Fig. 8. In the curriculum case, different stages are represented with different colors. The best performing network weights in a stage are stored (marked with black dots in Fig. 8). The next stage starts with the best weights of the current stage. We observe that curriculum training strategy yields significantly more rewards being collected especially in the earlier stages since the initial states are closer to the target states. However, at higher stages, the mean reward collected by the agent sometimes decreases to a lower number when compared with the vanilla case due to exploration of agent. Also, we cannot observe significant improvements in mean reward graph (Fig. 8b) at stages with $p_{max} > 1$ due to the usage of transferred policy networks. This implies that the policy learned is generalized to a larger area than the specified extent. Eventually, the mean reward graph is not directly reflects the performance of learned policy.

To get a better insight, we analyze the trajectories of the vanilla training and curriculum training in Fig. 9. The bi-rotor starts 2m away along x-axis and its transition is observed through positions and velocities in horizontal and vertical directions for five seconds. A significant steady state error can be observed for the vanilla training. On the other hand, the curriculum method is more maneuverable (settles in less than 1s) and has much less steady state error. The reason for this improvement may be that the policy first learns to stabilize closer to the goal point and this behaviour is preserved at next stages.

6 Conclusion

In this paper, flight control of a planar bi-rotor platform is considered with two different control approaches, namely a conventional control approach and a learning-based approach. The conventional approach uses a cascaded controller structure from the literature while the learning-based approach uses Deep Reinforcement Learning to learn a non-linear control policy, a mapping from the platform state to the actuator commands. We also study, with both control approaches, the extension of the bi-rotor with a torque actuated tail appendage.

Our results suggest that the conventional and learning-based controllers navigate the bi-rotor in a similar manner even with the tail appendage for which there are multiple solutions. Although it is not directly observed in our analyses, the two controllers do have differences: A small change in the input state can lead to a significant change in the performed action in the learning-based approach. Therefore, the learning-based controller can be more maneuverable than the conventional controller and preferable for flight in tight spaces with obstacles. On the other hand, the conventional controller is mathematically proven to be stable, which is not possible with the learning-based controller. The convergence is also provable in a conventional controller while the learning-based policy convergence to the goal is demonstrated in our experiments with some remaining steady-state error.

If we compare the two approaches, we should note that the learning-based controller only necessitates the reward function but requires a significant computational effort for training. On the other hand, the conventional controller requires a model-based controller design as well as a cost function, which can be considered as the counterpart of the reward function, to optimize its parameters. Although

the computational load of the learning-based controller is relatively higher than the model-based controller, most of this occurs during the training phase (hence off-line) and it still can operate in real-time on specialized hardware [33] available today.

A curriculum learning strategy is also examined in our work by controlling the initial state distribution of the episodes. Our results suggest that training under a curriculum from simpler regions to harder ones helps the learning agent to find a better overall control policy.

We also investigate the use of the tail appendage, which promises an opportunity for a quickly changing attitude. It also facilitates controlling the robot's center of mass during flight. However, these benefits come with a complication in robot dynamics. Hence, it is hard to design and implement a conventional controller but a learning-based controller can learn those dynamics without significant modifications on the approach. Although our results could not substantiate a significant performance benefit of the tail-appendage in point-to-point moves, such an appendage can be a robotic arm that can serve other functional purposes. We believe demonstrating that platform motion can still be controlled with such an add-on is important for future work in mobile aerial manipulation.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Halil Ibrahim Ugurlu. The first draft of the manuscript was written by Halil Ibrahim Ugurlu and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This research is partially supported by METU-BAP with project no GAP-312-2018-2705. H. I. Ugurlu is supported under a scholarship grant from Scientific and Technological Research Council of Turkey (TÜBİTAK). S. Kalkan is supported by Scientific and Technological Research Council of Turkey (TÜBİTAK) through BIDEB 2219 International Postdoctoral Research Scholarship Program and the BAGEP Award of the Science Academy.

Availability of data and materials The code and data will be available at https://github.com/halil93ibrahim/gym-bi-rotor.git.

References

1. Acosta, J.Á.., Sanchez, M., Ollero, A.: Robust control of underactuated aerial manipulators via Ida-Pbc. In: 53Rd IEEE Conference on Decision and Control, pp. 673–678. IEEE (2014)
2. Ankaralı, M.M., Saranlı, U., Saranlı, A.: Control of underactuated planar hexapedal pronking through a dynamically embedded slip monopod. In: 2010 IEEE International Conference on Robotics and Automation, pp. 4721–4727. IEEE (2010)
3. Bansal, S., Akametalu, A.K., Jiang, F.J., Laine, F., Tomlin, C.J.: Learning quadrotor dynamics using neural network for flight control. In: 2016 IEEE 55Th Conference on Decision and Control (CDC), pp. 4653–4660. IEEE (2016)
4. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 41–48. ACM (2009)
5. Beul, M., Behnke, S.: Analytical time-optimal trajectory generation and control for multirotors. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 87–96. IEEE (2016)
6. Bitcraze, A.: Crazy ie 2.0 (2016)
7. Bou-Ammar, H., Voos, H., Ertel, W.: Controller design for quadrotor Uavs using reinforcement learning. In: IEEE International Conference on Control Applications (CCA), pp. 2130–2135. IEEE (2010)
8. Bouabdallah, S., Siegwart, R.: Full control of a quadrotor. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 153–158. Ieee (2007)
9. Briggs, R., Lee, J., Haberland, M., Kim, S.: Tails in biomimetic design: Analysis, simulation, and experiment. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1473–1480. IEEE (2012)
10. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
11. Catto, E.: Box2d: A 2d physics engine for games (2011)
12. Daftry, S., Bagnell, J.A., Hebert, M.: Learning transferable policies for monocular reactive Mav control. In: International Symposium on Experimental Robotics, pp. 3–11. Springer (2016)
13. De Simone, M.C., Russo, S., Ruggiero, A.: Influence of aerodynamics on quadrotor dynamics
14. Demir, A., Ankaralı, M.M., Dyhr, J.P., Morgansen, K.A., Daniel, T.L., Cowan, N.J.: Inertial redirection of thrust forces for flight stabilization. In: Adaptive Mobile Robotics, pp. 239–246. World Scientific (2012)
15. Eberhart, R., Kennedy, J.: Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, vol. 4, pp. 1942–1948. Citeseer (1995)
16. Florensa, C., Held, D., Wulfmeier, M., Zhang, M., Abbeel, P.: Reverse curriculum generation for reinforcement learning. arXiv preprint arXiv:1707.05300 (2017)
17. Gao, F., Han, L.: Implementing the nelder-mead simplex algorithm with adaptive parameters. Comput. Optim. Appl. **51**(1), 259–277 (2012)
18. Graves, A., Bellemare, M.G., Menick, J., Munos, R., Kavukcuoglu, K.: Automated curriculum learning for neural networks. in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1311–1320. JMLR. org (2017)
19. Held, D., Geng, X., Florensa, C., Abbeel, P.: Automatic goal generation for reinforcement learning agents. arXiv preprint arXiv:1705.06366 (2017)
20. Hill, A., Raffin, A., Ernestus, M., Gleave, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y.: Stable baselines https://github.com/hill-a/stable-baselines (2018)
21. Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., Hutter, M.: Learning agile and dynamic motor skills for legged robots, vol. 4 (2019)
22. Hwangbo, J., Sa, I., Siegwart, R., Hutter, M.: Control of a quadrotor with reinforcement learning. IEEE Robot. Autom. Lett. **2**(4), 2096–2103 (2017)
23. Imanberdiyev, N., Kayacan, E.: A fast learning control strategy for unmanned aerial manipulators. J. Intell. Robot. Syst. **94**(3-4), 805–824 (2019)
24. Kawaguchi, K.: Deep learning without poor local minima. In: Advances in Neural Information Processing Systems, pp. 586–594 (2016)
25. Kawaguchi, K., Kaelbling, L.P.: Elimination of all bad local minima in deep learning. arXiv preprint arXiv:1901.00279 (2019)

26. Koch, W., Mancuso, R., West, R., Bestavros, A.: Reinforcement learning for uav attitude control. ACM Trans Cyber-Phys Syst **3**(2), 1–21 (2019)
27. Lee, T., Leok, M., McClamroch, N.H.: Geometric tracking control of a quadrotor Uav on Se (3). In: 49Th IEEE Conference on Decision and Control (CDC), pp. 5420–5425. IEEE (2010)
28. Lepine, M.D.: Design of a personal aerial vehicle (2017)
29. Li, Q., Qian, J., Zhu, Z., Bao, X., Helwa, M.K., Schoellig, A.P.: Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 5183–5189. IEEE (2017)
30. Libby, T., Moore, T.Y., Chang-Siu, E., Li, D., Cohen, D.J., Jusufi, A., Full, R.J.: Tail-assisted pitch control in lizards, robots and dinosaurs. Nature **481**(7380), 181–184 (2012)
31. Matiisen, T., Oliver, A., Cohen, T., Schulman, J.: Teacher-student curriculum learning. arXiv preprint arXiv:1707.00183 (2017)
32. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
33. Molchanov, A., Chen, T., Hönig, W., Preiss, J.A., Ayanian, N., Sukhatme, G.S.: Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. arXiv preprint arXiv:1903.04628 (2019)
34. Ren, H., Zhao, Y., Xiao, W., Hu, Z.: A review of uav monitoring in mining areas: Current status and future perspectives. Int. J. Coal Sci. Technol. **6**(3), 320–333 (2019)
35. Ritz, R., Hehn, M., Lupashin, S., D'Andrea, R.: Quadrocopter performance benchmarking using optimal control. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5179–5186. IEEE (2011)
36. Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive Uav control in cluttered natural environments. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1765–1772. IEEE (2013)
37. Sadeghi, F., Levine, S.: Cad2rl: Real single-image flight without a single real image. arXiv preprint arXiv:1611.04201 (2016)
38. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
39. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT press (2018)
40. Tang, S., Kumar, V.: Autonomous flying. Annu. Rev. Control Robot. Auton. Syst. **1**, 6.1–6.24 (2018)
41. Tsouros, D.C., Bibi, S., Sarigiannidis, P.G.: A review on uav-based applications for precision agriculture. Information **10**(11), 349 (2019)
42. Zhang, T., Kahn, G., Levine, S., Abbeel, P.: Learning deep control policies for autonomous aerial vehicles with Mpc-Guided policy search. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 528–535. IEEE (2016)
43. Zhao, J., Zhao, T., Xi, N., Cintrón, F.J., Mutka, M.W., Xiao, L.: Controlling aerial maneuvering of a miniature jumping robot using its tail. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3802–3807. IEEE (2013)

Halil Ibrahim Ugurlu received his B.Sc and M.Sc. degrees in Electrical and Electronics Engineering from Middle East Technical University (METU), Turkey in 2016 and 2019. He is studying as a Ph.D. researcher at Artificial Intelligence in Robotics laboratory (AiRLab) in Aarhus University, Denmark. His primary research interests lie within the areas of AI-based control and motion planning of robots.

Sinan Kalkan received his M.Sc. degree in Computer Engineering from Middle East Technical University (METU), Turkey in 2003, and his Ph.D. degree in Informatics from the University of Göttingen, Germany in 2008. After working as a postdoctoral researcher at the University of Gottingen and at METU, he became an assistant professor at METU in 2010. Since 2016, he has been working as an associate professor on problems within Computer Vision, and Developmental Robotics.

Afsar Saranli received his B.S degree in 1993 with Honors and Ph.D. degree in 2000 both from the Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara, Turkey. His M.Sc. degree is with Distinction in 1994 from the Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, London, England. During the 1995-1999 period, he was also working part-time as a science consultant to STFA Savronik Inc., a defence electronics company in Ankara, Turkey. Dr Saranl? joined IPS Automation Inc., Toronto, Canada in 2000 as a Senior Computer Scientist (later became Photon Dynamics Canada Inc.) where he worked until 2005. His focus was on signal and image processing as well as on control algorithms for vision based automated inspection systems for the automotive glass and LCD manufacturing industries. He then returned to Turkey to join the Department of Electrical and Electronics Engineering, Middle East Technical University as an Assistant Professor where he is currently a Full-Time Professor directing the Laboratory of Robotics and Autonomous Systems (RoLab). His current research interests include estimation and tracking for radar and other sensor systems, sensor based mobile robotics, dynamics and control of legged robot locomotion. He is a co-author of four US and International patents in the field of automated vision-based inspection as well as author or co-author of scientific publications in the field.

Affiliations

Halil Ibrahim Ugurlu[1] · Sinan Kalkan[2,3] · Afsar Saranli[4]

Sinan Kalkan
skalkan@metu.edu.tr

Afsar Saranli
afsars@metu.edu.tr

1 Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark

2 Department of Computer Engineering, Middle East Technical University, Ankara, Turkey

3 Visiting Researcher at the Department of Computer Science, Technology, University of Cambridge, Cambridge, UK

4 Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara, Turkey