Aerobatic Tic-Toc Control of Planar Quadcopters via Reinforcement Learning

Zhikun Wang, Roderich Groß, Shiyu Zhao

***Abstract*—This paper studies aerobatic tic-toc control of quadcopters. Tic-toc control enables rotorcraft to fly almost in the vertical plane rather than the horizontal plane. It is one of the most challenging manoeuvrers to achieve autonomously. The problem has to our knowledge not yet been studied for quadcopters. Studying it could expand their flight envelope and improve their performance in extreme, aerobatic flight tasks. In this paper, we employ a deep deterministic gradient policy approach to train reinforcement learning (RL) controllers based on carefully designed rewards. The obtained RL controllers are shown to generate two flight modes, spin and tic-toc. We analyse the properties of these flight modes and screen out unfavourable RL controllers. The qualified RL controller is then enhanced by combining it with PID and LQR controllers which achieves better flight performance and enables the quadcopter to track a moving reference point and recover to hovering flight status. Physical simulations using Simscape are presented to verify the proposed approach.**

***Index Terms*—Variable-Pitch Propeller Quadcopter, Flight Control, Reinforcement Learning**

I. Introduction

Quadcopter unmanned aerial vehicles (UAVs) are widely used due to their simple mechanical design and control structures. Nowadays, an increasing amount of tasks pose high requirements on the manoeuvrability and anti-interference ability of quadcopter UAVs. Variable pitch propeller (VPP) quadcopters are a relatively new type of quadcopter, that can exhibit performances superior to the conventional fixed-pitch ones. Specifically, a VPP can control its pitch angle by an actuator, thereby generating forces in either positive or negative directions. As a result, a VPP quadcopter can fly upside-down, which is impossible for fixed-pitch quadcopters. Therefore, VPP quadcopters exhibit great potential in many applications that require high-performance flight.

Although VPP quadcopters have received increasing attention in recent years, studies mainly focus on fault-tolerant control [1, 2]. The great potential in manoeuvring flight has not been well explored up to now. In fact, VPP quadcopters are suitable for a variety of aerobatic flight manoeuvres. Exploring these manoeuvres could broaden the flight envelope and help address complex flight scenarios. They could be relevant in entertainment and military applications as well as in applications requiring aircraft to navigate narrow confined spaces, all of which have received increased attention in recent years.

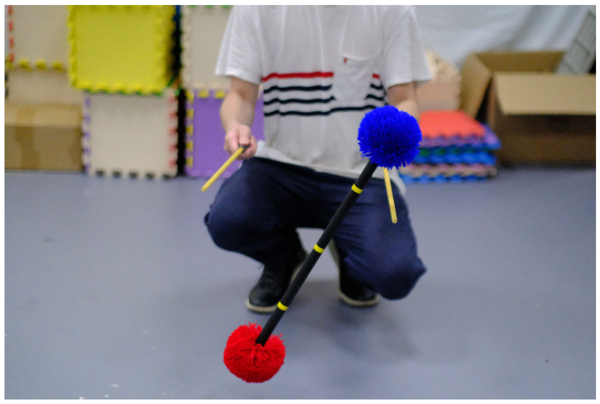


Fig. 1: A typical tic-toc (also known as ‘the pendulum’) manoeuvrer of the devil sticks.

Among aerobatic flight manoeuvres, tic-toc is one of the most challenging to be achieved autonomously. The tic-toc manoeuvre attempts to fly the UAV in a vertical plane rather than a horizontal plane. As the UAV is not able to fly steadily in a vertical plane due to the lack of vertical lift, it has to periodically swing back and forth to approximately keep a vertical flight pose. Such a periodic movement can be observed in juggling (see Fig. 1). It is a typical aerobatic manoeuvre of helicopters [3]. The work in [4, 5] realized autonomous tic-toc control of a helicopter using inverse reinforcement learning. Such a method, however, requires data of tic-toc trajectories generated by expert pilots in advance.

Up to now, an autonomous tic-toc manoeuvre of a quadcopter has not yet been reported in the literature. Moreover, how to realize it by self-learning without data generated by a skilled pilot is still an open problem. This paper studies this problem. As the dynamical system is extremely complex, we consider a simplified planar quadcopter model to simulate a VPP quadcopter [6, 7]. Even though the dynamic model is simplified to be two-dimensional, we still face many of the challenges. In particular, as the tic-toc movement is not around any equilibrium point, equilibrium-based control approaches are not applicable.

Manuscript received: September 9, 2021; Revised: December 3, 2021; Accepted: December 24, 2022.

This paper was recommended for publication by Editor Pauline Pounds upon evaluation of the Associate Editor and Reviewers’ comments.

Z. Wang and R. Groß are with the Department of Automatic Control and Systems Engineering, The University of Sheffield, United Kingdom. S. Zhao is with the School of Engineering at Westlake University and the Institute of Advanced Technology at the Westlake Institute for Advanced Study, China. Z. Wang completed part of this work when he was a visiting student at Westlake University. {zwang119, r.gross}@sheffield.ac.uk, zhaoshiyu@westlake.edu.cn

Corresponding author: Shiyu Zhao.

Digital Object Identifier (DOI): see top of this page.

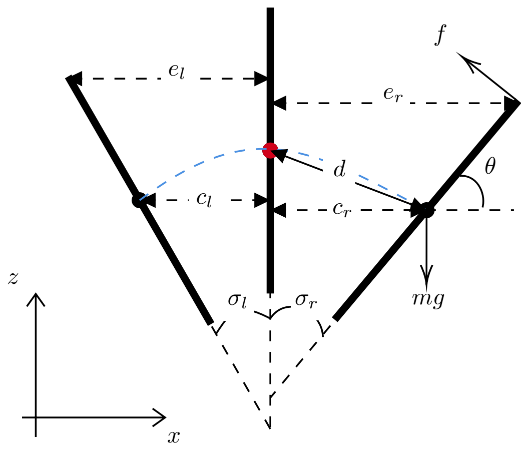


Fig. 2: Schematic diagram of the whole period of the VPP quadcopter tic-toc manoeuvrer.

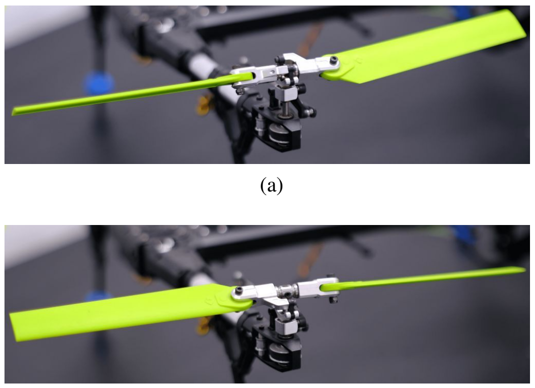


Fig. 3: A VPP actuator can use spinning speed and propeller pitch angle to change the required force and torque. A counter-clockwise rotating VPP generates a positive force with a positive propeller angle (a) and a negative force with a negative propeller angle (b).

As far as the authors are aware, the tic-toc manoeuvre has not been achieved by any conventional control approaches in the literature. In this paper, we design controllers for tic-toc aerobatic flight for planar VPP quadcopters via reinforcement learning (RL). RL is a method that enables an agent to use the reward obtained from its interactions with the environment to generate its control policy [8]. It has received significant attention in recent years due to its potential to address problems that are challenging to solve by conventional control approaches [9, 10]. Although RL has been applied to the control of multi-rotor drones, it is mainly used to achieve flight near the equilibrium point, such as throw-and-hover [11] and attitude control [12].

We use the deep deterministic gradient policy (DDPG) approach to train RL controllers based on carefully designed rewards. The obtained RL controllers are shown to generate two flight modes: spin and tic-toc. The flight performance of either mode is carefully analysed. Then, we evaluate and screen out unfavourable RL controllers by a non-dominated sorting approach [13]. Finally, we extend the remaining RL controller by introducing a compensation control, so that the tic-toc motion can follow a moving reference point, and an LQR-based recovery control, so that the quadcopter can recover from tic-toc to hovering flight. A series of studies are conducted in simulation to verify the proposed approach.

II. Problem Statement

The planar VPP quadcopter is modelled as a stick with uniform mass distribution as illustrated in Fig. 2. A force f is applied at one end of the stick. Its direction is always perpendicular to the stick. Its sign can be positive or negative (see Fig. 3). We consider only a single force as doing so is already sufficient to achieve the tic-toc manoeuvre as demonstrated by Fig. 1. Interestingly, a single force acting on the stick end is not sufficient for a quadcopter to hover. Tic-toc is one of a few flight modes that a quadcopter could use to stay in the air under these constraints.

Let d be the distance between the centre point and a reference point (i.e., the red point in Fig. 2). The control objective is to design f such that d is as small as possible. As there are no equilibrium states, it is challenging to formally define the target state. We will later quantify the objective by using rewards when designing RL algorithms.

In the following, the states of the stick and the dynamic model are presented. The position and velocity of the centre point of the stick are $[x, z]$ and $[u, w]$, respectively. The attitude of the planar quadcopter is described by θ, which is the angle between the stick and the x-axis. The spinning rate is q. Let m and l denote the mass and half length of the stick, respectively, I the moment of inertia, g the gravitational constant, and f_T and τ_T the total thrust and torque, which are given by

$$\begin{aligned} f_T &= f, \\ \tau_T &= fl. \end{aligned} \tag{1}$$

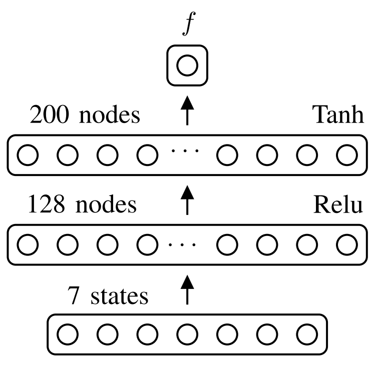
Then, the overall state vector is $[x, z, \theta, u, w, q]$ and the dynamic model is

$$\begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \dot{u} \\ \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} u \\ w \\ q \\ -f_T \sin\theta/m \\ f_T \cos\theta/m - g \\ -\tau_T/I \end{bmatrix}. \tag{2}$$

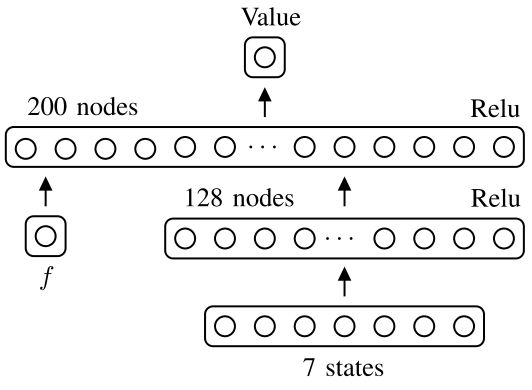
III. Reinforcement Learning Controller: Training and Analysis

A. Algorithm Structure

We apply the DDPG approach reported in [9] to train the RL controller. The approach comprises two parts, an actor Neural Network (NN) and a critic NN. The actor NN is an agent that works in the environment whereas the critic NN evaluates the performance of the agent. A deterministic policy gradient algorithm is used to update the actor NN. The critic NN is a value based deep Q-learning NN that uses state feedback and action as input while its output is a temporal-difference error used to evaluate the performance of the actor.



(a) Actor NN structure



(b) Critic NN structure

Fig. 4: The neural networks of the DDPG training algorithm.

The NN structures are shown in Fig. 4. Subsequent layers are fully connected. The output of the actor NN is the force f that acts at the end of the planar quadcopter. There are 7 state feedbacks which are obtained by the agent from its interaction with the environment: $\left[\sin(\theta), \cos(\theta), \dot{\theta}, x, z, \dot{x}, \dot{z}\right]$. The state feedback quantities are also used to design rewards. The whole NN is built in the Matlab Deep Reinforcement Learning toolbox environment.

B. Training Process

As there are no target equilibrium states, we must design a representative reward to reflect our control objective. The rewards are explained in the following. Positive and negative rewards, respectively, are used to encourage and penalize certain behaviour.

1) To reward the centre point of the quadcopter for approaching the target location, we design the following distance deviation penalty function:

$$r(d) = -0.1d^2 - 100d_{\text{far}},$$

where

$$d_{\text{far}} = \begin{cases} 0, & \text{if } d < 4; \\ 1, & \text{if } d \geq 4. \end{cases}$$

TABLE I: Planar quadcopter parameters

Parameter	Symbol	Value	Unit
Original mass	m_o	1	kg
Mass	m_n	0.665	kg
Gravitational acceleration	g	10	m/s^2
Original moment of inertia	I_o	0.083	kg$\cdot$ m^2
Moment of inertia	I_n	0.007	kg$\cdot$ m^2
Original length	l_o	0.5	m
Length	l_n	0.175	m
Original force limit	$\max f_{\text{o}}$	200	N
Force limit	$\max f_{\text{n}}$	25	N
Force changing rate limit	$\max \dot{f}_{\text{n}}$	1000	N/s

Here, d is a non-negative distance value and d_{far} gives a strong penalty when the centre point gets too far away from the reference point. When $d_{\text{far}} = 1$, the episode is stopped.

2) To reward the quadcopter for assuming a vertical attitude (i.e. θ close to $-\pi/2$), we use reward function

$$r(\theta) = -0.01(\theta + \pi/2)^2,$$

where $\theta \in [-\pi, \pi)$.

3) To minimize the required force magnitude, a force penalty is designed as

$$r(f) = -0.01f^2.$$

4) To encourage the quadcopter to remain in the air for a long time, we increase the reward with the flying time by

$$r(t) = 0.1t,$$

where t is the time elapsed since the start of the episode.

In total, the reward function is

$$R = r(d) + r(\theta) + r(f) + r(t). \tag{3}$$

For each episode, the quadcopter starts from the initial hovering state, which is $x_0 = 0, z_0 = 0, \theta_0 = 0$. The target reference position is randomly generated, $x_t, z_t \in [-1, 1]$. The latter helps strengthening the RL controller's generalization ability and stability [14]. The parameters of the quadcopter used for training are provided with subscript o in Table I. Within each episode, the quadcopter tries to reach, and remain close to the target position. If the quadcopter flies too far away (more than 4 m), the episode will be marked as a failure and stop. We train the DDPG agent for 30000 episodes, with each episode lasting at most 10 s with 0.02 s sampling time. All episodes whose returns are greater than -50 are saved for further analysis.

C. Analysis of Results

The trained RL controllers exhibit two flight modes. The first is a spin mode, where the planar quadcopter spins around a fixed position, therefore θ varies from 0 to 360 degrees (see Fig. 5). The second is a tic-toc mode, where the planar quadcopter swings around a point back and forth, therefore θ is constrained in a bounded interval (see Fig. 6).

In either of the flight modes, the entire flight can be split into two phases. The first is a settling phase, in which the

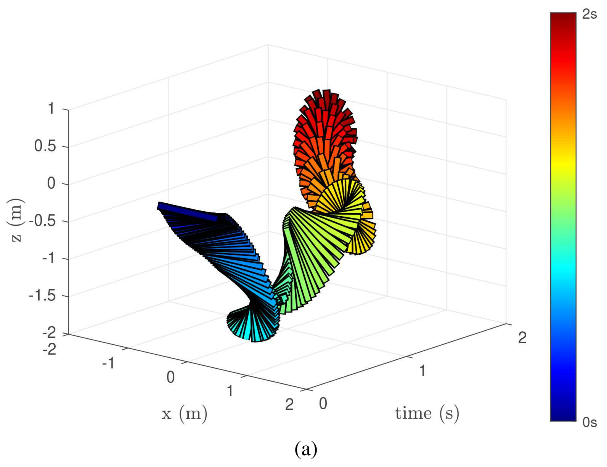
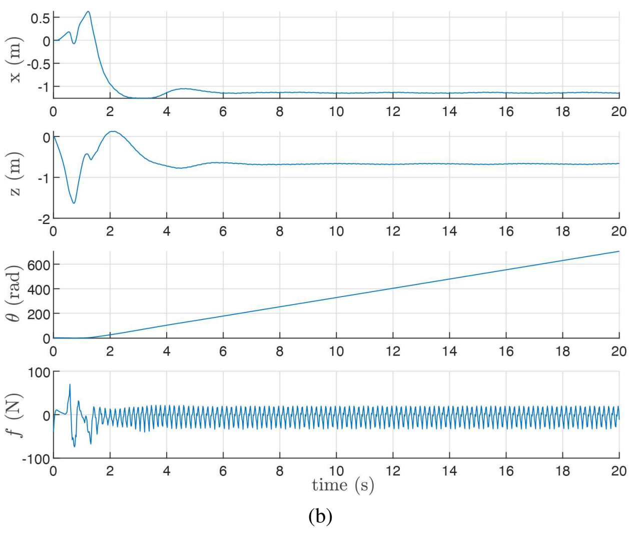



Fig. 5: Results of a quadcopter holding its position by using the spin mode neural network. (a) shows the quadcopter's motion during the first 2 seconds, whereas (b) shows the state evolution during the full 20 seconds duration. The colour bar represents time. It takes the quadcopter about 2 seconds to reach the manoeuvre phase.

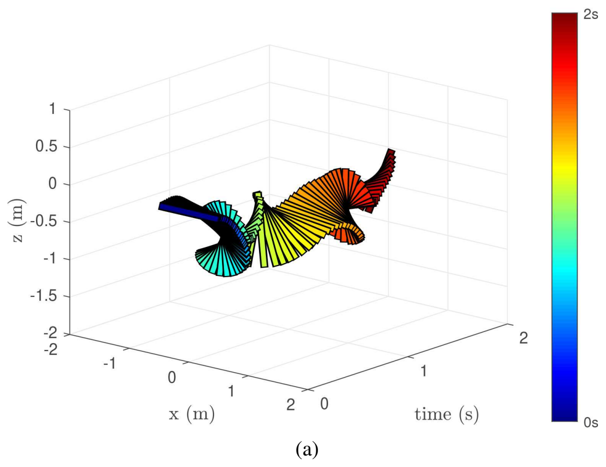
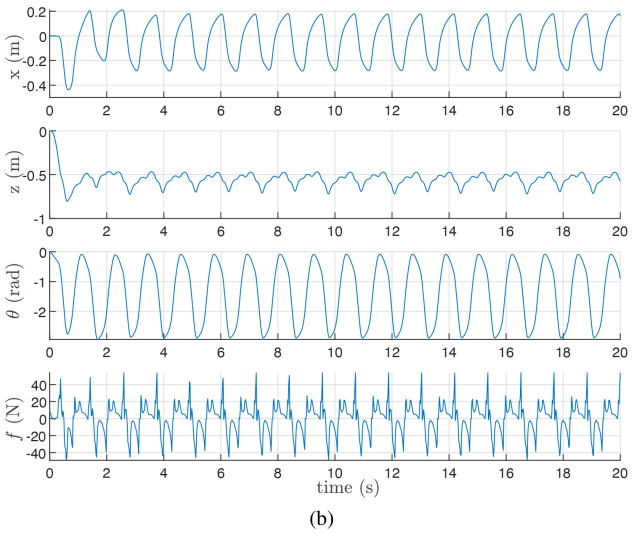



Fig. 6: Results of a quadcopter holding its position by using the tic-toc mode neural network. (a) shows the quadcopter's motion during the first 2 seconds, whereas (b) shows the state evolution during the full 20 seconds duration. The colour bar represents time. The quadcopter succeeds in switching to the tic-toc manoeuvre with very little initialization time.

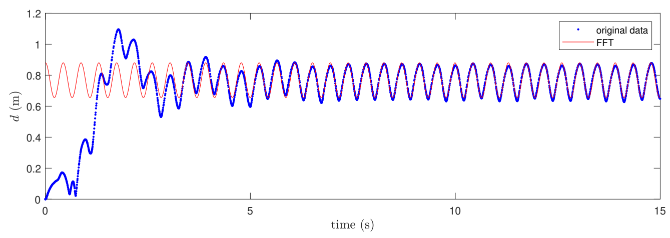


Fig. 7: Settling time quantification through comparison of the original distance curve and the Fourier fitting curve. To avoid interference via the initialization phase, we exclude the first 5 seconds when generating the Fourier fitting curve. The fitting distance function shown in the figure is $d_f(t) = 0.76 + 0.11\cos(14.46t) - 0.001\sin(14.46t)$, and the settling time is around 2.8 s.

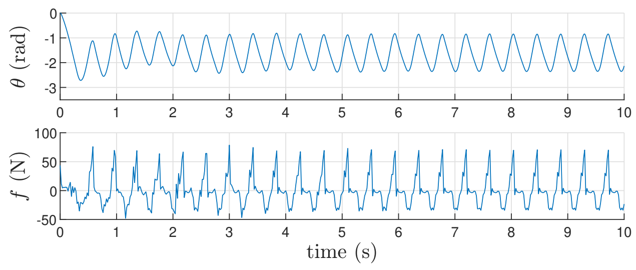
planar quadcopter starts from a horizontal attitude and then gradually converges to a steady periodic motion. During this phase, there will be large position and attitude deviations. The second is a steady periodic phase, in which the states vary in a steady periodic manner.

In order to quantify the settling time, we use trigonometric functions to fit the steady periodic curve. As illustrated in Fig. 7, the distance d between the quadcopter centre point and the reference point is shown by the blue curve. We can fit the blue curve in the steady periodic phase by

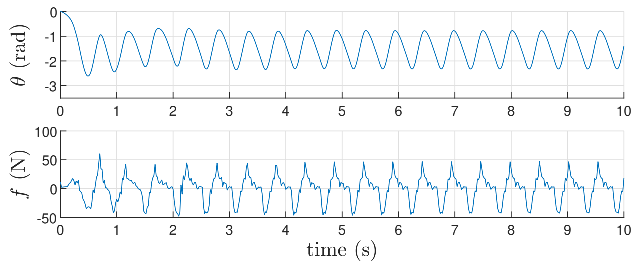
$$d_f(t) = A_0 + A_1\cos(\omega t) + A_2\sin(\omega t).$$

Once $|d(t) - d_f(t)|$ remains below a threshold (for example, 0.1) the settling phase ends and the corresponding time is the settling time. We have checked all the trained controllers and noticed that it takes a longer settling time for the spin mode (around 6 s) than for the tic-toc mode (around 3 s).

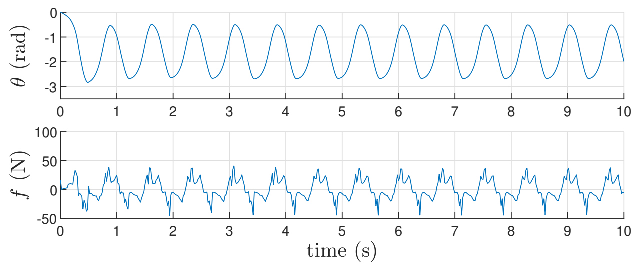
By comparing the performance of the spin and tic-toc modes as shown in Figs. 5 and 6, the tic-toc mode does not exhibit significant fluctuations during the settling phase and reaches the steady periodic phase faster. In terms of space occupation, the spin mode takes more space than the tic-toc mode (observed from the XZ plane). In terms of required force, the spin mode requires high force (70 N maximum) in the settling phase and then much less force (25 N maximum)



(a)



(b)



(c)

Fig. 8: Comparison of different NNs that have similar reward value where (a) shows a high-speed tic-toc mode NN performance, (b) shows a medium-speed tic-toc mode NN performance and (c) shows a low-speed tic-toc mode NN performance. It is difficult to distinguish these performances by only considering the reward function.

TABLE II: Comparison among three tic-toc manoeuvre NNs with different speeds.

	High-speed	Medium-speed	Low-speed
Swing Angle (rad)	1.12	1.38	2.26
Time Period (s)	0.45	0.56	0.94
Maximum Force (N)(+)	73.68	50.34	32.65
Maximum Force (N)(-)	-46.43	-55.28	-48.34

in the steady periodic phase. As a comparison, the required force of the tic-toc mode does not vary significantly during different phases and the maximum value is 50 N.

While we have incorporated an angle penalty reward function $r(\theta)$, why could the controller still learn the spin mode? This is because the total reward function considers other properties as well and we stored all the RL NNs whose overall rewards were greater than the set threshold.

IV. Reinforcement Learning Controller: Further Evaluation and Screening

In the last section, we showed that the tic-toc flight mode has less settling time and lower space occupation than the spin mode. It should be noted that different training episodes can lead to different controllers, which may all achieve the tic-toc mode but have very different performances. The inherent diversity of solutions (and performances) is fundamentally due to the total reward function being composed of a mixture of metrics, and the trained controllers may place different emphasis on different metrics. In this section, we evaluate different tic-toc controllers and show how to screen out solutions according to additional metrics.

Figure 8 shows three examples to demonstrate the performance of different controllers. The examples could be classified to be high-speed, medium-speed, low-speed controllers based on the periodic time of their steady phases. The specific values of the maximum swing angles, periodic time, and maximum forces of these examples are given in Table II, where the parameters of the dynamical system used for training is given in Table I. As can be seen, the smaller the swing angle and the shorter the time period, the larger the force that is required.

In the rest of the section, we introduce three metrics to evaluate different RL controllers and propose a method to screen out unfavourable ones.

A. Evaluation Metrics

To give an overall evaluation of the performance of the NN, we use the following three metrics:

1) The first metric is the mean distance between the centre and reference points. It is denoted as d_{mean}.
2) The second metric is the maximum space occupation s_{max}, which is defined as

$$s_{\text{max}} = \max\left(|e_r|, |e_l|\right), \tag{4}$$

where e_r and e_l denote respectively the rightmost and leftmost distance of the stick's top end from the vertical plane (see Fig. 2).
3) The third metric, f_{max}, is the maximum magnitude of the force during the entire control process including both settling and steady periodic phases. This metric would be relevant for practical realizations of the RL controller.

It must be noted that the three metrics could not be designed as rewards during the training process. That is because they are defined for the entire control process and can not be used for timely feedback to evaluate the performance of the training NN controller.

It is favourable if d_{mean}, s_{max}, and f_{max} are small; the smaller the better.

B. Network Screening

This subsection addresses how to screen a large number of RL controllers based on the aforementioned three metrics.

One approach is to assign different weights to the three metrics according to ones' own preferences and then use a weighted summation of the three metrics as a single metric. Since many networks perform well on one metric but worse on another, one overall metric may not be sufficient to choose a suitable controller. Therefore, we use a screening algorithm

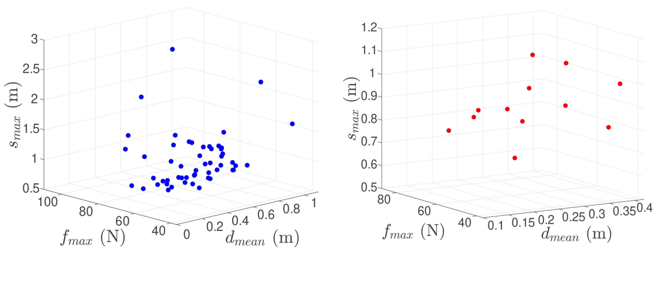


Fig. 9: Screening of trained NN controllers. Each axis represents a different metric; (a) shows the point cloud of all trained NNs and (b) shows the Pareto-efficient frontier point cloud of all trained NNs.

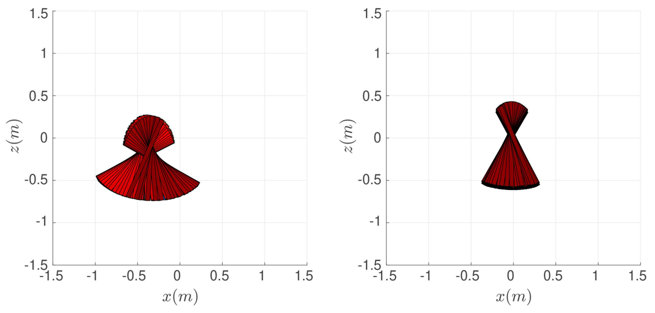


Fig. 10: Simulation results (steady periodic phase) of the planar quadcopter with or without trajectory compensation. The trained NN strikes a good balance among all three objective functions (d_{mean}, s_{max} and f_{max}). (a) shows the simulation results of the NN controller without trajectory compensation and (b) shows the simulation results of the NN controller with trajectory compensation. The overall maximum space occupation is decreased by 53%.

to identify the set of best NNs from all the trained NNs. Here "best" refers to NNs that are not dominated by other NNs.

Figure 9(a) shows all trained NNs as a point cloud in the metric space, where the axes correspond to the three metrics d_{mean}, s_{max} and f_{max}, respectively. Figure 9(b) shows the corresponding Pareto-efficient frontier.

While this method screens out a large portion of unfavourable NNs, we can further reduce the selection based on our preferences. For example, if we want the space occupation to be as small as possible, we could choose the bottom-left red dot in Fig. 9(b). In addition, we can choose NNs from anywhere on the Pareto optimal surface to meet multiple metric requirements.

V. Reinforcement Learning Controller: An Extension

This section extends the RL controllers to further improve their performance. In addition, a recovery controller is designed to restore hovering flight and a NN migration method is provided to realize the control of different dynamic models by the target NN.

A. Trajectory Compensation

A problem of the RL controllers is that there may exist steady-state errors between the average position of the centre point and the reference point. We now seek to suppress this steady error. Moreover, we seek to further decrease the maximum space occupation s_{max}.

We design a trajectory compensation method that uses a PID controller to compensate the steady-state error and reduce s_{max}. The idea of this method is to design a swing trajectory to offset unnecessary displacement, thereby reducing the maximum manoeuvrer distance deviation. In our scenario, the error between actual and desired trajectories is given to the PID controller as the input where the output value is the trajectory that needs to be compensated.

Figure 10 verifies the effectiveness of the proposed controller. As can be seen, the compensation method can reduce the average steady-state error from $[-0.54, -0.22]$ to $[-0.05, -0.12]$. Moreover, s_{max} decreased from 0.98 m to 0.51 m. However, this comes at the cost of the maximum force increasing from 78 N to 165 N.

The advantage of this compensation controller is that we can flexibly enhance the flight performance of an existing trained RL controller according to our needs, instead of training new RL controllers.

B. Recovery Controller System

Our trained RL controller can only achieve tic-toc flight. If the planar quadcopter was to restore hovering flight, a new controller would need to be designed and integrated.

We introduce an LQR controller for hovering flight control based on a modified version of (1) and (2). The modification is to change the single force as in (1) to two forces applied on the two ends of the stick. As a result, (1) becomes

$$\begin{aligned} f_T &= f_r + f_l, \\ \tau_T &= (f_r - f_l)l, \end{aligned}$$

where f_r and f_l are the forces acting on right and left ends of the VPP quadcopter, respectively. Then, we linearise the system based on the dynamic model presented in (2). A standard LQR controller is designed based on the linearised model. The design of the LQR controller is omitted here.

By combining such a controller with the RL controller, a quadcopter could start from a hovering position, switch to the tic-toc flight, and finally switch back to the hovering mode. It should be noted that two forces are required for hovering and only one force is needed for tic-toc. The overall control structure is illustrated in Fig. 11.

C. Network Migration

When we train an RL controller, we need to specify a set of parameters of the planar quadcopter such as its mass and length. However, once we apply the trained controller in practice, the parameters may vary across different platforms. To solve this problem, we could adjust the force and torque generated by the trained RL controller according to the

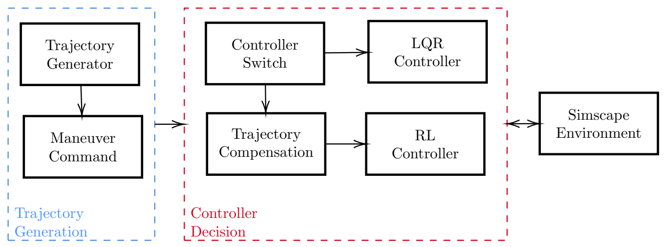


Fig. 11: Overall control system structure used for the narrow space passing simulation. The structure is divided into three parts, trajectory generation (shown in blue), controller decision (shown in red) and simulation environment.

specific parameters of the target platform to

$$\begin{aligned} f_n &= \frac{m_n}{m_o} f_o, \\ \tau_n &= \frac{l_o I_n}{I_o l_n} \tau_o, \end{aligned} \tag{5}$$

where parameters and variables with subscript o and n correspond to the original and new parameters, respectively.

Substituting the migration equations (5) to (2) gives

$$\begin{aligned} f_o \sin\theta / m_o &= f_n \sin\theta / m_n, \\ f_o \cos\theta / m_o &= f_n \cos\theta / m_n, \\ \tau_o l_o / I_o &= \tau_n l_n / I_n. \end{aligned}$$

Therefore, once the parameters of the standard training model and the target model are known, we can migrate the RL controller to the target model to make their performance consistent.

D. Simulation Validation

We study three simulation scenarios to examine the performance of the integrated system, comprising the RL controller, compensation controller, network migration subsystem and hovering recovery controller.

The simulation is conducted in Simscape, a physical simulation environment in Matlab. We discard aerodynamic forces caused by the obstacles. Table I lists the physical parameters of the VPP quadcopter model as used in training (with subscript o) and simulation (with subscript n).

In the first scenario, as shown in Fig. 12, from $t = 0$ to $t = 5$, the quadcopter switches from a hovering position to a tic-toc flight mode. Starting from $t = 5$, it tracks a moving reference point upward thereby passing a narrow passage. At $t = 12$, it switches successfully back to hovering.

In the second scenario, noises and wind disturbance are considered during validation. Noise with a signal-to-noise ratio of 15 dB is added to all the feedback states. The wind disturbance is 2 N along x-axis and 2 N along z-axis from 3 s to 6 s. Moreover, actuator constraints are added to the simulation where the maximum thrust of the actuator is limited to 25 N and the maximum thrust changing rate is limited to 1000 N/s [15]. The result presented in Fig. 13 indicates that our designed control system has good robustness (green lines represent the duration of the wind disturbance).

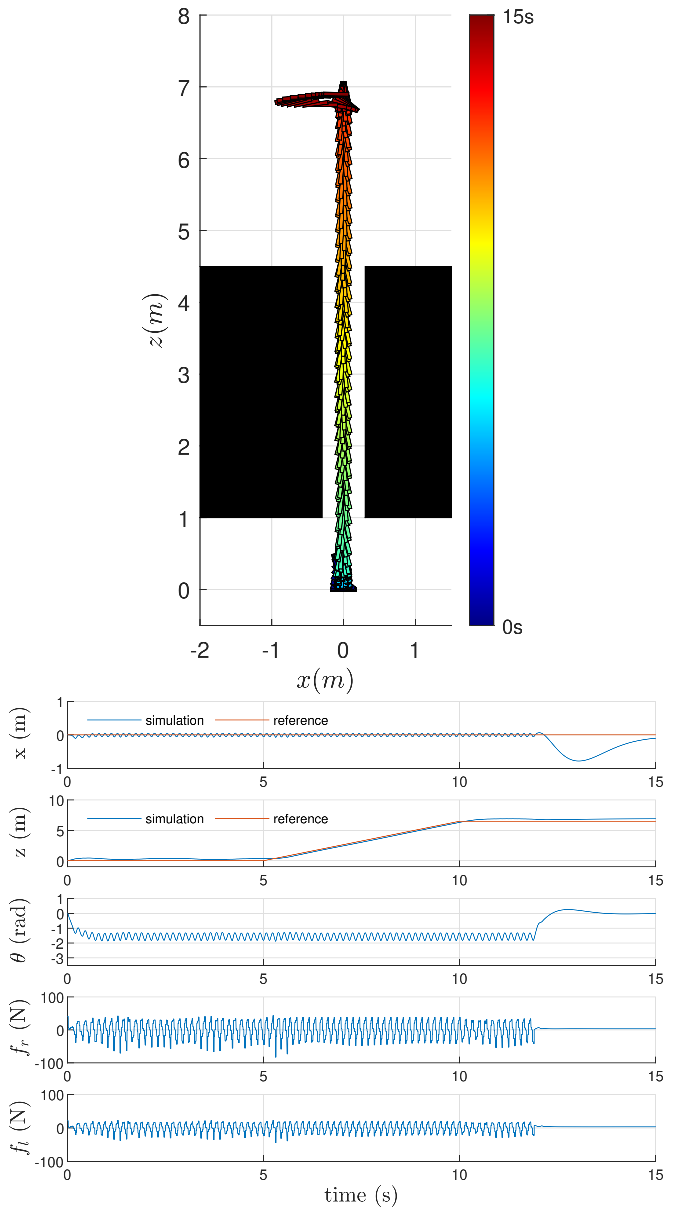


Fig. 12: Simulation results of passing through vertical obstacles with the NN migration, where the overall maximum force was 84 N and maximum space occupation was 0.4 m.

In the third scenario, we assume that the parameters of the real system can not be measured accurately. The values of the parameters used in (5) are mismatched, with different degrees of uncertainty. In particular, these parameters are sampled from uniform distributions within a mismatch percentage as follows $a_n = a_r(1 + k_a), k_a \in [-b, b]$, where a_r is the real value of the parameter, a_n is the inaccurate value of the parameter used in Eq. (5), k_a is a random variable drawn uniformly from $[-b, b]$, and b is the model parameters mismatch percentage. We test the controller with different model parameters mismatch percentage $b \in \{0.1, 0.2, \ldots, 0.8\}$. For each tested mismatch percentage, we repeated the simulation for 100 episodes and calculate the average success rate. The episode is marked as success when the agent can perform

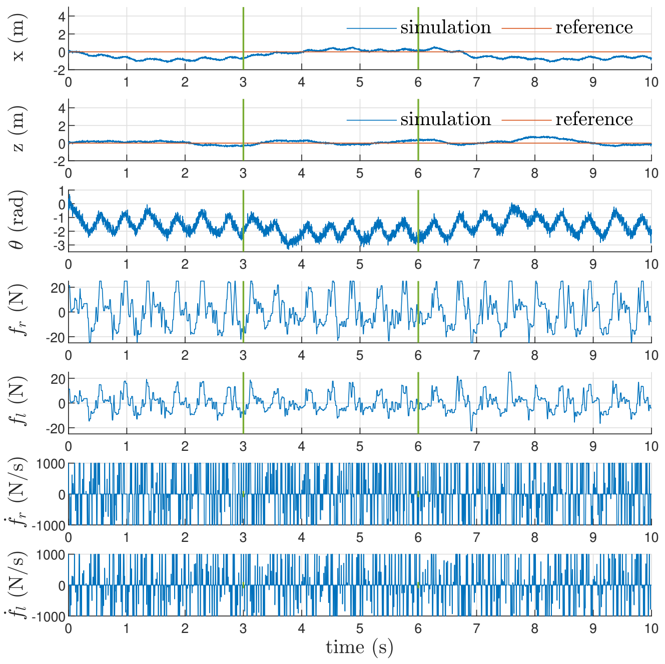


Fig. 13: Results of a VPP quadcopter in the tic-toc mode facing external disturbances and noises.

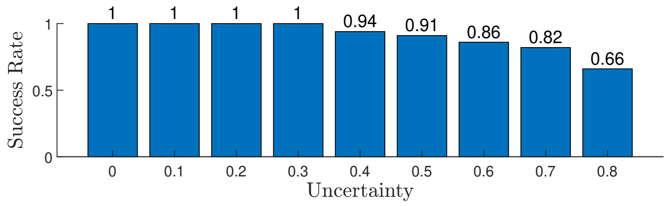


Fig. 14: Results of a VPP quadcopter holding its position with tic-toc mode under different parameter mismatches.

the tic-toc manoeuvrer for 5 seconds. The result is shown in Fig. 14. It indicates that our trained controller reliably performs the tic-toc manoeuvre if the mismatch is not large. This suggests that the reinforcement learning based control system has certain generalization ability to handle model mismatches.

The findings demonstrate the potential application of the proposed control approach. To the best of our knowledge, no other methods has previously achieved autonomous tic-toc manoeuvrer with a quadcopter model.

VI. Conclusion

This paper presented for the first time an RL NN controller which was trained on a planar quadcopter model to successfully perform the tic-toc manoeuvre. We extended the controller and demonstrated its ability to perform position tracking and narrow vertical tunnel passage in a simulation environment. The supplementary video contains these and other demonstrations. In this paper, flying through a narrow gap is an example to show the potential applications of our proposed control system. Our study aims to explore the limit of the manoeuvrability of VPP quadcopters. It could deepen our understanding of the dynamical features of VPP quadcopters and lead to more interesting and practical control strategies. Future work will consider controllers acting in more realistic scenarios and validating them on physical quadcopters.

VII. Acknowledgement

The authors would like to thank Kai Eivind Wu for his helpful suggestions on the evaluation and screening part of our controller.

References

[1] Z. Wang, R. Groß, and S. Zhao, "Controllability analysis and controller design for variable-pitch propeller quadcopters with one propeller failure," *Advanced Control for Applications: Engineering and Industrial Systems*, p. e29, 2020.
[2] A. Baldini, R. Felicetti, A. Freddi, S. Longhi, and A. Monteriù, "Actuator fault tolerant control of variable pitch quadrotor vehicles," in *Proceedings of 21st IFAC World Congress*, vol. 53, no. 2. IFAC, 2020, pp. 4095–4102.
[3] F3CN, "Helicopter Manoeuvre Descriptions," 2020. [Online]. Available: https://www.f3cn.org/index.php/system/files/archive/Annex%205F.1%20F3N%20Manoeuvre%20Descriptions.pdf
[4] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems*, 2007, pp. 1–8.
[5] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.
[6] T. Tomić, M. Maier, and S. Haddadin, "Learning quadrotor maneuvers from optimal control and generalizing in real-time," in *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1747–1754.
[7] G. Wu and K. Sreenath, "Safety-critical control of a planar quadrotor," in *Proceedings of 2016 American Control Conference (ACC)*. IEEE, 2016, pp. 2252–2258.
[8] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.
[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[10] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.
[11] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.
[12] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.
[13] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 97–112, 2016.
[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.
[15] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3, pp. 1213–1247, 2020.