Flying Through a Narrow Gap Using End-to-end Deep Reinforcement Learning Augmented with Curriculum Learning and Sim2Real

Chenxi Xiao[1†], Peng Lu[2*†] and Qizhi He[3]

***Abstract*—Traversing through a tilted narrow gap is previously an intractable task for reinforcement learning mainly due to two challenges. First, searching feasible trajectories is not trivial because the goal behind the gap is difficult to reach. Second, the error tolerance after Sim2Real is low due to the relatively high speed in comparison to the gap's narrow dimensions. This problem is aggravated by the intractability of collecting real-world data due to the risk of collision damage. In this paper, we propose an end-to-end reinforcement learning framework that solves this task successfully by addressing both problems. To search for dynamically feasible flight trajectories, we use a curriculum learning to guide the agent towards the sparse reward behind the obstacle. To tackle the Sim2Real problem, we propose a Sim2Real framework that can transfer control commands to a real quadrotor without using real flight data. To the best of our knowledge, our paper is the first work that accomplishes successful gap traversing task purely using deep reinforcement learning.**

***Index Terms*—Quadrotor, Reinforcement Learning**

I. Introduction

A. Problem Background

Aggressive flight can enhance the maneuverability of quadrotors. For instance, in search and rescue applications, quadrotors are required to explore unstructured environments with narrow entries. The quadrotor's activity range can be enlarged if it is capable of flying through narrow gaps that are considered intractable from the aspect of a classical flight controller. Moreover, consider that quadrotors powered by batteries have a limited activity range. Avoiding obstacles by taking shortcuts through narrow gaps may reduce the power consumption by avoiding taking long detours. Accordingly, it is necessary to develop motion planners that can perform aggressive motions.

However, constrained flight motions can be difficult to perform due to the quadrotor's underactuated dynamics. One example is to fly through a tilted rectangular gap, during which the quadrotor needs to avoid collision and subject to attitude and position constraints simultaneously. However, keeping a tilted attitude may induce a large horizontal acceleration. As a result, it would generate a horizontal position shift and increase the chance of colliding with the bezel. Therefore, finding a feasible trajectory is not trivial.

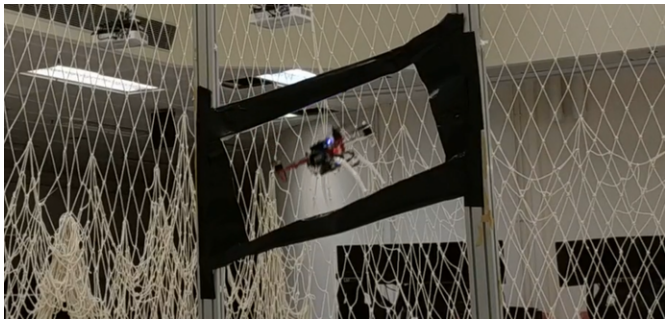


Fig. 1: The quadrotor controlled by reinforcement learning is passing through a tilted narrow gap.

Aggressive flight planning has been explored for decades [1], [2]. Conventional studies model the problem as a constrained motion planning problem that can be solved by optimizing manually defined loss functions. However, all these approaches have to simplify the problem using strong mathematical assumptions so that it can be formulated under the optimal control paradigm. During optimization, excessive prior knowledge is added (refer to Sec. II-B), such that motions that are inconsistent with the priors will be penalized during optimization. Accordingly, only solutions of a few specific patterns can be obtained, which eliminated the possibility of obtaining a solution of better patterns.

Compared to previous works, model-free reinforcement learning mainly has two advantages. Firstly, control policy can be optimized directly using unstructured environments and also under the quadrotor's strong non-linear dynamics. Convexity of loss function is still desired but is no longer a prerequisite. Secondly, the solution pattern is not biased by the aforementioned handcrafted priors. Instead, the model-free learning paradigm only relies on a reward function that indicates whether the goal has been reached. Therefore, leveraging on the reinforcement learning makes it feasible to get a solution that is better than the sub-optimal trajectory from the user-defined solution space.

B. Contributions

In this paper, we propose an end-to-end reinforcement learning solver for the quadrotor's gap traversing task. Our approach does not rely on excessive problem-orientated priors. The methodological contributions are mainly from two

* Corresponding author
† These authors contributed equally to this manuscript.
[1] Chenxi Xiao was with the Hong Kong Polytechnic University and is currently with Purdue University, xiao237@purdue.edu
[2] Peng Lu is currently with the Adaptive Robotic Controls Lab at the University of Hong Kong and was previously with the Hong Kong Polytechnic University. lupeng@hku.hk
[3] Qizhi He is with Northwestern Polytechnical University, heqizhi@mail.nwpu.edu.cn

aspects. First, due to the limited exploration ability of current reinforcement learning algorithms, searching for a feasible trajectory is not trivial. To this end, we propose to guide the exploration using curriculum learning, with which we have acquired feasible trajectories without conventional model-based motion planners. Second, transferring our learned policy to a real quadrotor is challenging due to the low error tolerance of Sim2Real as well as the intractability of collecting real trajectories. To tackle this issue, We propose a novel Sim2Real approach that enables successful Sim2Real transfer without using real flight trajectories.

II. Related Works

A. Drone control and planning by reinforcement learning

Reinforcement learning is reportedly a powerful approach for various flight control and planning tasks. Zhang et al. [3] applied Guided Policy Search (GPS) to a quadrotor collision avoidance task. The policy from GPS can outperform offline iterative LQG planner and MPC planner using an ideal quadrotor model, and a model with 5 percentage mass error. Hwangbo et al. [4] demonstrated a method to train a reinforcement learning policy that can control a real-world quadrotor from the motor thrust level. The learned policy can accomplish hovering, waypoint tracking and posture stabilization from random initial states. Molchanov et al. [5] stabilized a quadrotor using Proximal Policy Optimization (PPO) algorithm. Lambert et al. [6] stabilized a quadrotor based on a model-based reinforcement learning approach. Li et al. [7] designed a reinforcement learning policy that can track targets. Mannucci et al. [8] proposed two reinforcement learning-based algorithms to control the attitude of the aircraft.

B. Quadrotor traversing through a narrow gap

In previous studies, the gap-traversing task has been solved by Falanga et al.[1] and Loianno et al.[2] based on the conventional optimal control framework . To be specific, Falanga et al.[1] modeled the problem as an optimization problem on a pre-defined trajectory set. The trajectory is constrained to be a quadratic function that must intersect with the gap center. The vehicle's velocity and acceleration during traversing are manually defined, and the time duration of traversing is to be reduced by optimization. However, the excessive hand-crafted prior knowledge may stifle better solutions to be obtained, since there is no evidence that the involved constraints are able to generate optimal solutions. The method is also problem-orientated and not scalable to unstructured environments with additional obstacles or irregular gap dimensions. Similarly, Loianno et al.[2] also defined the problem in an optimal control paradigm with excessive priors i.e. a parabolic trajectory, constant motor thrust, zero angular velocity during traversing, and a fixed trajectory starting point.

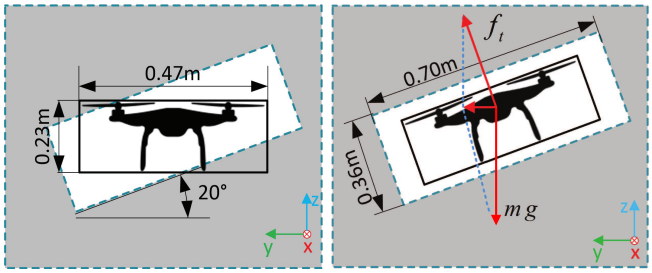
To reduce the dependencies on the aforementioned priors, literature [9] is the first known study that implemented gap traversing using reinforcement learning. A neural network is utilized to imitate trajectories from an optimal control solver. The solution was then fine-tuned by training in AirSim [10]. The final trajectory pattern is reportedly more diverse than the parabolic curve trajectories from previous studies [1], [2]. However, the initial trajectory being cloned is still obtained from the optimal control framework with excessive priors. It is known that imitation learning may still end up with local optimal solutions that are similar to demonstrations without sufficient exploration [11]. Besides, the method is still not detached from optimal control that requires excessive priors. Therefore, a pure reinforcement learning solver that can solve the problem in an end-to-end paradigm is desired. To the best of our knowledge, our work is the first instance of work that only uses a model-free reinforcement learning solver to accomplish this gap-traversing task in the real world.

III. Task and Method Overview

A. Task Statement

Our task is to plan aggressive trajectories for passing through a tilted narrow hole, as demonstrated in Fig. 1.

A direct traverse is not feasible, as shown in Fig. 2 (a). The black rectangle is the bounding box of the quadrotor. The gray background rectangle represents a wall with a tilted gap. Fig. 2 (b) shows an instance in which the geometric constraint is satisfied. But the joint force induced by motor thrusts and quadrotor's gravity will lead to additional horizontal acceleration that may lead to a collision, as shown by red arrows. In addition, the pitch angle used for dashing forward will increase the lateral area of the quadrotor's bounding box, which reduces the safe distance margin.



(a) (b)

Fig. 2: (a) A direct traverse that cannot be accomplished. (b) One possible traverse which avoids collision with the gap. However, the horizontal acceleration may lead to collision.

We demonstrate our training framework in Fig. 3. These modules (Simulation, Soft Actor-Critic, Sim2Real) are discussed in Section IV, V, VI, respectively.

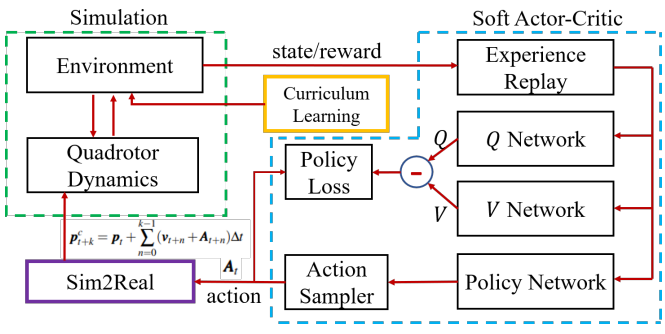


Fig. 3: An overview of our proposed training framework.

IV. Simulation Environment for Reinforcement Learning

Model-free reinforcement learning is notorious for its inefficiency in data usage. Training the policy directly in real world is impractical because a real quadrotor is too fragile to endure a large number of failure rollouts. Instead, a simulation environment is created using the dynamics described in Sec. IV-A.

A. Quadrotor Dynamics

We model the quadrotor as a rigid body with non-linear dynamics [12]. The angular acceleration is modeled as Eq. (1).

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} \tau_\phi I_{xx}^{-1} \\ \tau_\theta I_{yy}^{-1} \\ \tau_\psi I_{zz}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y \end{bmatrix} \tag{1}$$

τ_ϕ, τ_θ, τ_ψ are the roll, pitch and yaw torques, respectively. I_{xx}, I_{yy}, I_{zz} are the rotational inertia of x, y and z axis in the body frame. ω_x, ω_y, ω_z are the roll, pitch and yaw rates, respectively.

Similarly, we model the translational motion as Eq. (2). m is the quadrotor's mass, $\boldsymbol{a}$ is the rigid body linear acceleration, g is the gravitational constant. f_t is the total thrust. $\boldsymbol{e}_3 = [0,0,1]^T$, $\boldsymbol{R}$ is the rotational matrix from body to earth frame, $\boldsymbol{f}_d$ is the drag force induced by linear motions, which is proportional to the squared body linear velocity v_x, v_y, v_z in its x, y, z axis, respectively [13].

$$m\boldsymbol{a} = m\boldsymbol{e}_3 g + \boldsymbol{R}\boldsymbol{e}_3 f_t + \boldsymbol{f}_d \tag{2}$$

We use control distribution matrix to model the mapping relationship from motor thrusts to τ_ϕ, τ_θ, τ_ψ and f_t, as shown in Eq. (3). C_T is the thrust coefficient, and C_M is the torque coefficient. Note that the control distribution matrix corresponds to the X type quadrotor and therefore the arm length is $\frac{\sqrt{2}}{2}d$, where d is the horizontal side length of the Oriented Bounding Box (OBB).

$$\begin{bmatrix} f_t \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} C_\mathrm{T} & C_\mathrm{T} & C_\mathrm{T} & C_\mathrm{T} \\ \frac{\sqrt{2}}{2}dC_\mathrm{T} & -\frac{\sqrt{2}}{2}dC_\mathrm{T} & -\frac{\sqrt{2}}{2}dC_\mathrm{T} & \frac{\sqrt{2}}{2}dC_\mathrm{T} \\ \frac{\sqrt{2}}{2}dC_\mathrm{T} & \frac{\sqrt{2}}{2}dC_\mathrm{T} & -\frac{\sqrt{2}}{2}dC_\mathrm{T} & -\frac{\sqrt{2}}{2}dC_\mathrm{T} \\ C_\mathrm{M} & -C_\mathrm{M} & C_\mathrm{M} & -C_\mathrm{M} \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_3^2 \end{bmatrix} \tag{3}$$

B. Environmental State Variables

The state variable $\boldsymbol{s}$ used in the reinforcement learning constitutes of the following information: linear position error towards the goal state (p_x^e, p_y^e and p_z^e), linear velocities (v_x, v_y and v_z), roll and pitch angles (ϕ and θ), roll and pitch rates (ω_x and ω_y). Note that we do not implement control on the yaw channel and therefore we do not feed yaw information to the network. Each entry of the linear position error vector $\boldsymbol{p}^e$ is defined as:

$$p_i^e = \mathrm{sign}\left(p_i - p_{G_i}\right)\sqrt{|p_i - p_{G_i}|} \tag{4}$$

Subscript i corresponds to the x, y and z position channel. $\boldsymbol{p}$ is the robot position, and $\boldsymbol{p}_G$ is the position of the goal point (defined in the world frame, $\boldsymbol{p}_G$ is a fixed point located at 25 centimeters behind the gate's central point). Eq. (4) magnifies the positional error when the quadrotor is close to the gate's center, aiming to enhance the discriminability of the positional feedback in that case.

C. Reward Design

We use a simple reward function because we do not intend to restrict the solution space by excessive prior knowledge (discussed in Sec. I-A). We use a +1,000 value as the goal reward. This reward can only be acquired if the quadrotor passes through the hole without any collisions detected. The reward scaling is from parameter tuning. However, due to the difficulty of visiting the states behind the gate, this goal reward itself is too sparse to guide the training. An auxiliary penalty reward that is negative proportional to the distance $-\|\boldsymbol{p}-\boldsymbol{p}_G\|_2$ is also used. This penalty reward encourages the quadrotor to move towards the target and therefore significantly improved the training stability. Note that this auxiliary reward accumulated in the whole episode is much smaller than the goal reward because the solution should not be dominant by this auxiliary reward. Overall, the reward function $r(\boldsymbol{p})$ is given in Eq. (5)

$$r(\boldsymbol{p}) = \begin{cases} +1{,}000 & \text{(when goal is reached)} \\ -\|\boldsymbol{p}-\boldsymbol{p}_G\|_2 & \text{(otherwise)} \end{cases} \tag{5}$$

D. Simulated Gap

The environment includes a wall with a narrow gap. We terminate the simulation episode immediately when a collision between the quadrotor and the wall is detected. For this, we implemented a simple collision checker. The intersection points between the bounding box of the quadrotor and the wall are calculated in real-time. One collision is recognized if any intersection points are outside the gap's boundary. A traversing attempt is successful if no collision is detected till the quadrotor has reached the goal position$\boldsymbol{p}_G$.

V. Deep Reinforcement Learning

A. Soft Actor-Critic Algorithm

Reward sparsity is a challenge for our task since the goal reward behind the gap is difficult to reach. For this, we selected Soft Actor-Critic (SAC) algorithm [14], which has a strong ability of exploration due to the entropy term $\mathscr{H}$ (refer to Eq. (6)). Our preliminary experiments indicate that SAC converges faster than PPO [15] and DDPG [16]. Hence, SAC is chosen as the learning algorithm in this paper.

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t) + \alpha\mathscr{H}\left(\pi(\cdot|\mathbf{s}_t)\right)\right] \tag{6}$$

Where r is the step reward, $\mathbf{s_t}$, $\mathbf{a_t}$ are the state and action in the time step t. α is a weight parameter that determines the importance of the entropy term (α is subsumed into the reward r through scaling reward r by α^{-1} [14]). The optimal policy π^*_{Maxent} is given by (7):

$$\pi^*_{\mathrm{MaxEnt}}(\mathbf{a}_t|\mathbf{s}_t) = \exp\left(\frac{1}{\alpha}\left(Q^*_{\mathrm{soft}}(\mathbf{s}_t,\mathbf{a}_t) - V^*_{\mathrm{soft}}(\mathbf{s}_t)\right)\right) \tag{7}$$

The soft Q function Q^*_{soft}, soft V function V^*_{soft} are given by the soft learning framework, which are defined as Eq. (8) and Eq. (9). γ is the reward discount factor.

$$Q^*_{\text{soft}}(\mathbf{s}_t,\mathbf{a}_t) = r_t + \mathbb{E}_{(s_{t+1},\ldots)\sim\rho_\pi}\left[\sum_{l=1}^{\infty}\gamma^l\left(r_{t+l}+\alpha\mathscr{H}\left(\pi^*_{\text{MaxEnt}}\left(\cdot|\mathbf{s}_{t+l}\right)\right)\right)\right] \tag{8}$$

$$V^*_{\text{soft}}(\mathbf{s}_t) = \alpha\log\int_A \exp\left(\frac{1}{\alpha}Q^*_{\text{soft}}\left(\mathbf{s}_t,\mathbf{a}'\right)\right)d\mathbf{a}' \tag{9}$$

We approximate the policy with a neural network π_ϕ. This policy network has 2 linear hidden layers with 256 neural units in each layer. ReLU activation function is used in all hidden layers. We use reparameterization trick [17] to sample actions i.e. $\mathbf{a}_t = f_\phi(\varepsilon_t;\mathbf{s}_t)$, where ε_t is a noise signal sampled from a Gaussian distribution defined by the network output. We limit the action magnitude of each channel to (-1, 1) by a Tanh function. The overall network structure is given in Fig. 4.

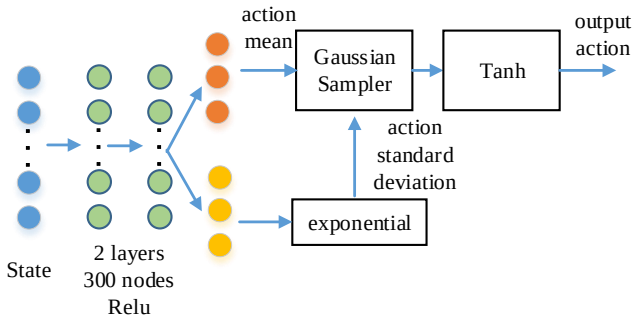


Fig. 4: Architecture of the policy network that predicts the distribution of actions conditioned on the input state. Reparameterization trick is used for sampling actions.

Q^*_{soft} function and V^*_{soft} are also approximated with neural networks Q_θ and V_ψ. Both of the two networks contain 3 hidden layers with 300 neural units in each layer. To prevent the overestimation of Q value, we follow the double-Q learning [18] [19] to approximate the Q^*_{soft} with the minimum output of two parallel Q networks.

We trained all these networks with Adam optimizer at a learning rate of 5×10^{-4} and a batch size of 1024. We identify that using a smaller learning rate (less than 1×10^{-4}) may lead to collapsed solution trajectories since it cannot follow the update speed of curriculum learning (refer to Sec. V-B) while using a large learning rate (larger than 2×10^{-3}) may reduce training stability. The reward discount factor γ is 0.99. We initialize the weights of the output layer in Q_θ and V_ψ as uniform values in $(-3\times10^{-3}, 3\times10^{-3})$, because we want to initialize the estimation of Q^*_{soft} and V^*_{soft} as roughly zero compared to the relatively large episodic reward. We believe this can alleviate the bias in selecting initial actions and may accelerate the training.

B. Curriculum Learning

We incorporate our proposed curriculum learning framework to address the reward sparsity issue. Curriculum learning [20] is a training technique that divides the training process into a sequence of subtasks with increased difficulty levels. which is known to be able to improve the convergence by letting the agent learn on a simplified problem at the beginning stage [21].

We design a curriculum with two training phases. In phase 1, the gap's dimensions gradually reduce from 1.5m × 1m to 1m × 0.5m. This phase lasts for 100,000 episodes. We control the gap's dimension by increasing the difficulty factor f_1 with the episode e_1, as described in Eq. (10). w and h are the width and height of the gap.

$$\begin{aligned} f_1 &= \min\left(0.5\sqrt{\tfrac{e_1}{10{,}000}}, 1.0\right) \\ w &= 1.5 - 0.5\cdot f_1 \\ h &= 1.0 - 0.5\cdot f_1 \end{aligned} \tag{10}$$

In phase 2, we adjust the difficulty factor f_2 according to Eq. (11). Phase 2 is used to refine the policy under the most difficult configuration. The phase 2 lasts for 500,000 steps in total. which shrinks the gap dimension from 1.0 × 0.5 to 0.6 × 0.3.

$$\begin{aligned} f_2 &= \min\left(0.5\sqrt{\tfrac{e_2}{150{,}000}}, 1.0\right) \\ w &= 1.0 - 0.4\cdot f_2 \\ h &= 0.5 - 0.2\cdot f_2 \end{aligned} \tag{11}$$

The best policy is chosen as the one with the maximized score $s = f_2 r^*$, where r^* is the exponential moving average of the episode reward r at episode e_2 ($r^*_{e_2+1} = 0.95r^*_{e_2} + 0.05r_{e_2}, r^*_0 = r_0$).

The curriculum learning changes the environmental configuration as the training proceeds. This means that the experience stored in the replay buffer may be obsolete. Therefore, we limit the size of our replay buffer to 100,000 and discard old data when the replay buffer is full. Empirically, the reward curve is stable when the replay buffer size varies from 10,000 to 500,000.

VI. Sim2Real Transfer

Discrepancies between the simulation and real quadrotors are non-negligible. Therefore, it is difficult to transfer the policy trained in simulation directly to real quadrotors. To solve this problem, a wide variety of Sim2Real approaches have been proposed [22], [23], [24], [25]. Nevertheless, most of these approaches need to utilize real-world data either in fine-tuning stage or in training stage. However, acquiring real-world data is challenging in our case (discussed in Sec. VII). To solve this problem, we developed a control framework that can enhance generalization without utilizing real-world data.

A. Simulation to Real Transfer Framework

An overview of our framework is shown in Fig. 5. The proposed framework is incorporated both in training and testing. Here we define the linear and angular acceleration command as $\boldsymbol{A}_t$. $\boldsymbol{A}_t$ is then converted into an incremental positional displacement starting from the current position $\boldsymbol{p}_t$.

Let $\boldsymbol{p}_t$, $\boldsymbol{v}_t$ denote the position and velocity of the quadrotor at time step t, respectively. We propose to design the position command as follows:

$$\boldsymbol{p}^c_{t+1} = \boldsymbol{p}_t + \boldsymbol{v}_t\Delta t + \frac{1}{2}\boldsymbol{A}_t\Delta t^2 \tag{12}$$

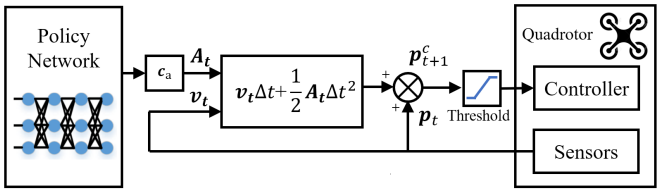


Fig. 5: Our proposed Sim2Real transfer framework. Position command $\boldsymbol{p}^c_{t+1}$ at time step $t+1$ is calculated using the acceleration command $\boldsymbol{A}_t$ and positional and velocity feedback $\boldsymbol{p}_t$, $\boldsymbol{v}_t$ at time step t

where $\boldsymbol{p}^c_{t+1}$ denotes the position command for the next time step $(t+1)$, Δt denotes the time interval between the two time steps.

$\boldsymbol{p}^c_{t+1}$ will be sent to the position controller for execution. The velocity $\boldsymbol{v}_t$ and position $\boldsymbol{p}_t$ are measured by sensors in real time. In our implementation, the policy network's output $\mathbf{a}$ is limited to $(-1,1)$ by a $tanh$ function. To convert this output value back to $\boldsymbol{A}_t$, a scaling parameter $\boldsymbol{c}_a$ is used following the output of the policy network. For $\boldsymbol{c}_a$, we set 40 rad/s^2 for angular channels and 12 m/s^2 for the altitude channel.

Our approach theoretically can work with continuous output, model-free reinforcement learning algorithms other than SAC, since it doesn't require any modification of the existing reinforcement learning architectures.

B. Randomization

Randomization is an effective way to enhance the success rate of Sim2Real transfer [25], [26]. In our training, we use two types of randomization: (1) Observation noise that represents the uncertainty of sensors. (2) Dynamics randomization that represents the model inaccuracy.

Noise (1) is modeled as additive noise sampled from Gaussian distributions $\mathcal{N}(\mu,\sigma^2)$. The mean value μ of noise is zero. The standard deviation σ is given in Table. I. The initial state of the quadrotor is randomized by generating from zero-mean Gaussian distributions with standard deviations given in Table. II, which enables to plan trajectories starting from a wide region rather than only from the origin.

TABLE I: Environmental randomization

	position p_x, p_y, p_z	angle ϕ, θ, ψ	linear velocity v_x, v_y, v_z	angular velocity ω_x, ω_y, ω_z
σ	0.002 m	0.01 rad	0.05 m/s	0.05 rad/s

TABLE II: Initialization randomization

	Initial linear velocity v_{x0}, v_{y0}	Initial angular velocity ω_{x0}, ω_{y0}, ω_{z0}	Initial position p_{x0}, p_{y0}, p_{z0}
σ	$0.01m/s$	$0.01rad/s$	$p_{x0}, p_{y0}: 0.5m$ $p_{z0}: 0.2m$

The dynamics randomization aims at pushing the learning algorithm to generalize on a wide range of quadrotor parameters. For this, we leverage additive zero mean Gaussian distributions, with standard derivation σ given in Table. III.

C. Traversing through gaps with various dimensions

To demonstrate the feasibility of our approach, we firstly evaluate the traversing success rate of our policy with various gap dimensions. The dimension of our quadrotor is 0.47m $\times$ 0.47m $\times$ 0.23m. The dynamics parameters of the quadrotor are $m=1.2\ kg$, total thrust $f_t=19.6\ N$, rotational inertia $I_{xx}=I_{yy}=0.007\ kg\cdot m^2$, $I_{zz}=0.014kg\cdot m^2$, thrust coefficient $C_T=6\times10^{-6}\ N/(rad/s)^2$ and torque coefficient $C_M=8\times10^{-8}\ N\cdot m/(rad/s)^2$, which is consistent with our real quadrotor. Both the training and testing stages are conducted in the simulation we built, which runs on a laptop with intel i7-8750H CPU and Nvidia GTX 1060 GPU. The tilted angle is fixed to 20 degrees in both training and testing as an example. We evaluate our approach on a wide variety of gap dimensions, with 1,000 episodes evaluated per experiment. The success rate is shown in Table IV.

TABLE III: Dynamics randomization

	rotational inertia $I=[I_{xx},I_{yy},I_{zz}]^T$	motor's max thrust $T_{max}=[T_{max1},\ldots,T_{max4}]^T$
σ	$0.15I$	$0.05T_{max}$

TABLE IV: Evaluation of the policy in simulation. We demonstrate the success rate (in %) for various gap dimensions (width & height, in meters)

width \ height	0.38	0.36	0.34	0.32	0.30
1.0	95.1%	93.0%	86.4%	70.5%	49.2%
0.9	90.0%	88.5%	83.5%	70.8%	46.6%
0.8	78.4%	75.8%	72.0%	58.6%	40.9%
0.7	45.6%	44.6%	42.8%	36.3%	24.0%
0.6	14.7%	12.6%	13.8%	11.6%	7.9%

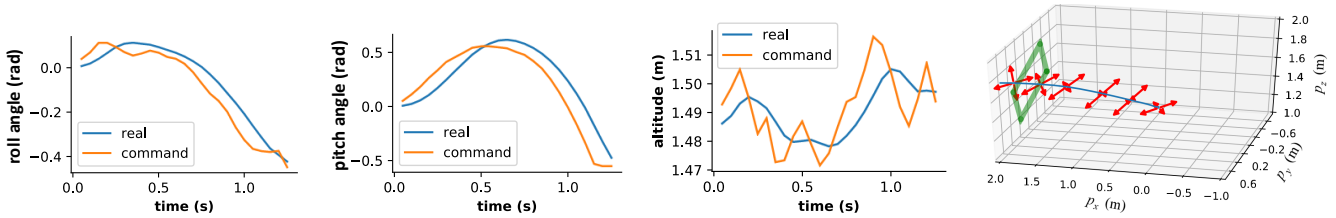
We demonstrate the learned policy by showing plots of the altitude and attitude data (Fig. 6). The pitch angle gradually increases to obtain a fast dashing speed. Then it gradually decreases because a large pitch angle may increase the chance of collision. The quadrotor finally takes advantage of the inertial velocity for the hole-traversing.

D. Real World Experimental Configuration

To show the feasibility of our proposed Sim2Real method, we then test our approach on a real F330 quadrotor. The parameters from model identification are the same as the counterparts in Sec. VI-C. The width of the gap is 0.7m and the height is 0.36m. The tilt angle is 20 degrees. We set the quadrotor's absolute maximum roll/pitch angle as 0.55 rad (about 31.5 degrees) to prevent losing altitude due to limited motor thrust.

We utilize Vicon mocap system to provide the position and velocity feedback. The whole reinforcement learning framework was running on an onboard Upboard computer with ROS. The system structure is shown in Fig. 7. The positional channels (outer loops) are controlled at 50 Hz while the attitude is controlled by the onboard Pixhawk controller at 250 Hz rate. Our code is released at: https://github.com/arclab-hku/reinforcement_learning.

We demonstrate the results of real world experiment. We conducted 37 trials of experiments. 15 of them are successful, which takes up about 40.54%. This success rate is close to 44.6% we achieved in the simulation. The traversing snapshots



(a) Roll command and response. (b) Pitch command and response (c) Altitude command and response (d) Trajectory of traversing

Fig. 6: Experimental data for traversing through a 20 degree tilted gap. Quadrotor attitude and altitude data are shown in (a), (b), and (c). (d) shows a trajectory that passes through the narrow gap successfully in our simulation.

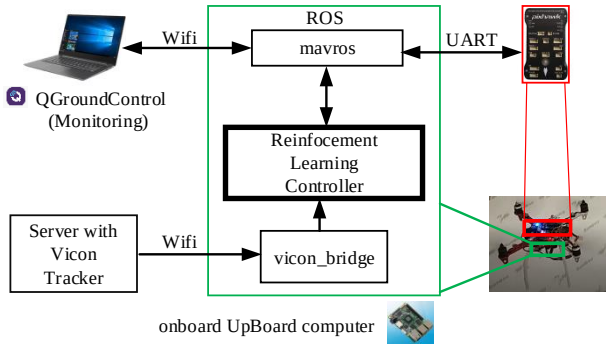


Fig. 7: The experimental configuration of our real-world experiment. The reinforcement learning controller is on an onboard Upboard computer. A Pixhawk module is used for flight control. Vicon tracker is used for position feedback.

are shown in Fig. 8. The video can be found at https://youtu.be/gfAfFnjN18A.

The key state variables in actual flights are demonstrated in Fig. 9. It can be observed that the action pattern closely matches the simulated counterparts. This demonstrates that our Sim2Real framework can effectively transfer the policy from simulation to a real quadrotor.

E. Performance without curriculum learning

We demonstrate the smoothed episodic reward r' (smoothed by $r'_{t+1} = 0.995r'_t + 0.005r_{t+1}$) in Fig. 10, with 95% confidence intervals. The cyan curve corresponds to the results with curriculum learning enabled, while the pink curve corresponds to the result with curriculum learning removed. Benefits from the curriculum learning, the cyan curve can maintain a high reward level during the whole training process. In comparison, the pink curve shows that the agent is not able to find the goal reward when the curriculum learning is removed, proving that curriculum learning can both improve the learning speed and stability.

F. Performance without Sim2Real transfer framework

We find it intractable to transfer a policy that directly exerts control on the attitude and altitude channels without using our proposed Sim2Real framework. For safety considerations, we only tested this transfer in simulation: we trained the policy using the simulated dynamics model and then transferred it to a quadrotor model controlled by PX4 firmware in Gazebo. No successful trajectory is achieved with a total number of 30 rollouts while at the same scenario we can achieve a success rate of 44.6% in the simulation using the proposed framework.

A planning result in Gazebo is shown in Fig. 11. It is seen that the attitude and altitude response is oscillatory, making it difficult to track the commands.

VII. Discussions

A. Other Sim2Real Approaches

Other recent proposed approaches mainly include: (1) learn a model of inverse dynamics that can predict required actions directly in the target domain [22]. (2) learn an adaptive policy that can be fine-tuned by real-world data [24] [27]. Unfortunately, none of these approaches is effective in our system.

(1) We have tried an inverse dynamic model as an attempt of the Sim2Real transfer (refer to [22]). However, it is intractable to fit an accurate global model or local models around aggressive trajectories, because a real quadrotor is fragile and therefore intensive data sampling around aggressive trajectories is not feasible. We have tried to use Ornstein-Uhlenbeck noise for model identification, but the noise magnitude should also be limited due to safety considerations. Hence, it is hard to bridge the data distribution gap between the identification phase and the validation phase.

(2) We seek an antidote in fast adaptive meta-learning by applying the Reptile algorithm [27]. By generating 1,000 quadrotors with dynamics randomization in our simulation, we intended to find a well-initialized model, and then fine-tune the model by the data acquired from the target domain. We use a Gazebo environment for the experiment. Using 5 shots of training, we achieve at most 3 successful rollouts out of 30 rollouts in total, which is a mundane performance compared to 10 successful rollouts achieved by our Sim2Real transfer framework.

(3) Other approaches that require real world data for domain transfer such as [28] are also intractable to be applied due to the difficulty of sampling a large number of aggressive trajectories from the real-world. This is because almost any failure trials would damage the quadrotor e.g. break propellers.

B. Failure pattern analysis

We aim to get the best performance on a real-world quadrotor rather than on the simulated counterparts. We can achieve more than 90% success rate in our simulation if we decrease the noise injected for Sim2Real, but it will degenerate the performance on a real quadrotor.

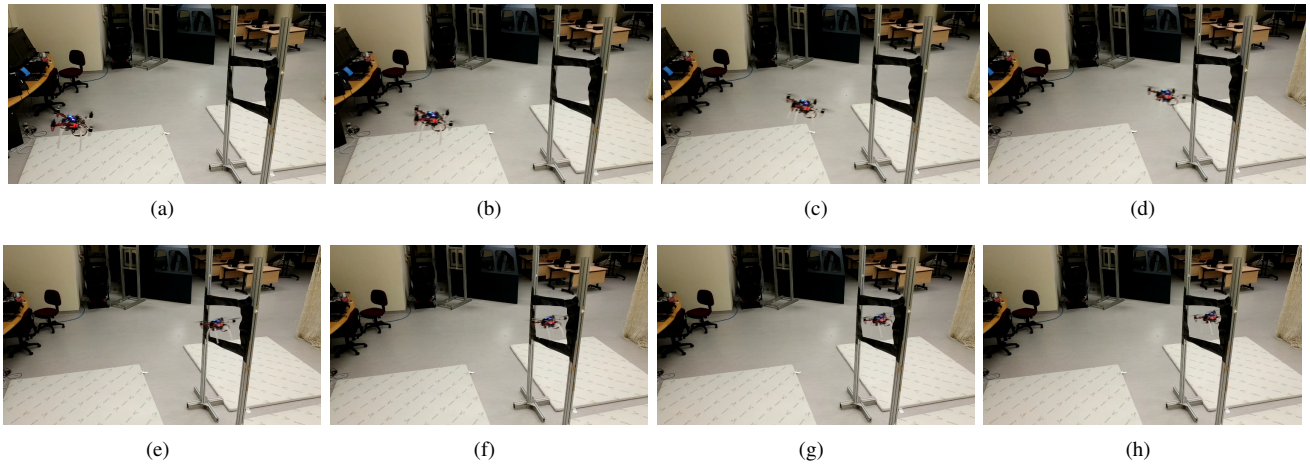
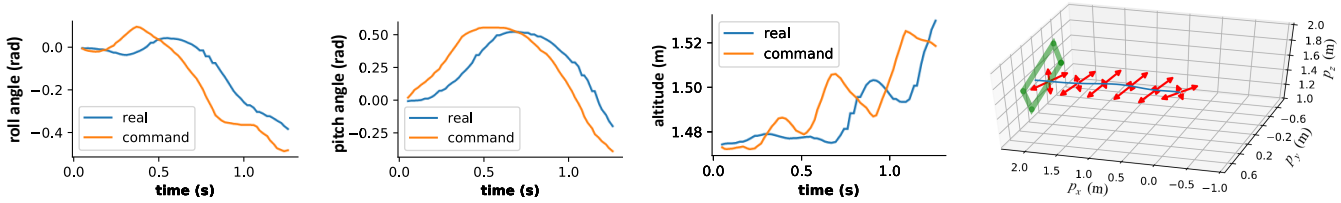


Fig. 8: Snapshots of the motions performed during the gap-traversing task.



(a) Roll command and response. (b) Pitch command and response. (c) Altitude command and response. (d) Trajectory of traversing

Fig. 9: Quadrotor states in a real-world experiment. (a), (b) and (c) show the command and response. (d) is the recorded trajectory that passed the narrow gap successfully.

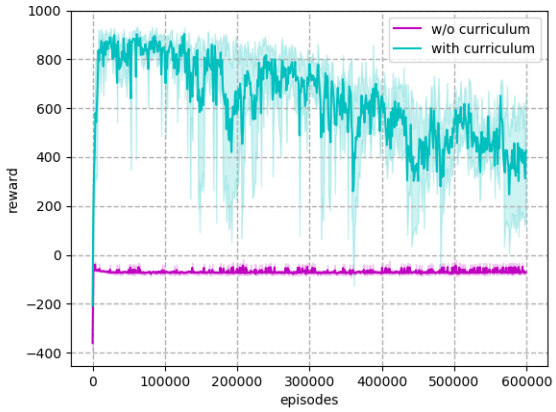


Fig. 10: Comparison of smoothed episodic reward of two training configurations: 1) With curriculum learning 2) Without curriculum learning. Both configurations have the same gap dimension (0.6x0.3) at the 600,000th episode. But only the former case can find the solution trajectory reliably.

Failures are caused by 1) inappropriate timing to start tilting, which implies that inaccurate decisions can still be made by the reinforcement learning agent. 2) inaccurate tracking of the altitude. The error in the altitude channel cannot be reduced swiftly once emerges, because the time constant in the altitude control channel is larger than the counterparts in attitude channels. Note that the controller only has fractions of a second for stabilization because the peak dashing speed of our quadrotor can be more than $3m/s$. A better altitude control algorithm that has a faster control response (such as the incremental nonlinear dynamic inversion in [13]) may contribute to a higher success rate.

C. Generalizability

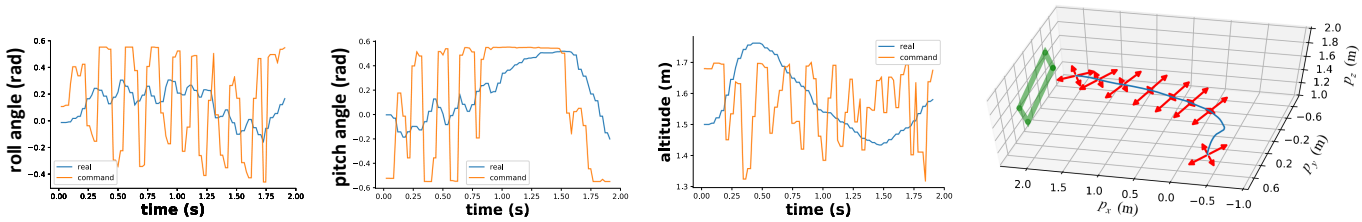
The proposed Sim2Real transfer framework, which does not need accurate parameters of the real quadrotor, makes the proposed approach less dependent on the model of the quadrotor and easy to be generalized. Because of this, the trained network in simulation can be successfully applied to the real quadrotor without training on the real data and achieves a similar success rate as in simulation. This demonstrates the generalizability of the proposed approach. The proposed Sim2Real transfer framework can be generalized to systems with similar dynamics as the quadrotor.

D. Limitation

For performing aggressive flights using reinforcement learning approaches, angle and rate limits can be violated. One approach to attenuate this issue is to design reward functions which penalize the actions that violate the rate limit. This approach can attenuate the issue but cannot eradicate it. Our proposed Sim2Real framework takes a further step by always keeping the rate limit within its maximum range. However, the maximum rate limit is a function of quadrotor state. Simply using a constant rate limit value would be harmful when generalizing to larger tilt angles.

VIII. Conclusion

We proposed a novel deep learning framework which enables the quadrotor to pass through narrow gaps without



(a) Roll command and response. (b) Pitch command and response. (c) Altitude command and response. (d) Trajectory of traversing

Fig. 11: Quadrotor response from a failure trajectory without using our proposed sim2real transfer framework in Gazebo environment. (a), (b) and (c) show the command and response. The commands are oscillatory, which leads to task failure. (d) is the corresponding recorded trajectory. The quadrotor collided with the wall.

training using real-world data. Two key challenges were addressed: 1) the sparse reward issue was solved by designing a curriculum learning framework, and 2) the Sim2Real transfer issue was addressed by proposing a novel framework which does not depend on the model parameters. Experimental results showed that the trained policy can achieve a similar success rate when applied to the real quadrotor without additional training. Future work would be to extend our work to scenarios with larger tilted angles using a more dexterous quadrotor, and to feed the gap's tilt angle to the network input, which can facilitate our proposed method to address varying tilting angles without the need to re-train the model.

REFERENCES

[1] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 5774–5781.

[2] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2016.

[3] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 528–535.

[4] 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.

[5] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," *arXiv preprint arXiv:1903.04628*, 2019.

[6] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[7] S. Li, T. Liu, C. Zhang, D.-Y. Yeung, and S. Shen, "Learning unmanned aerial vehicle control for autonomous target following," *arXiv preprint arXiv:1709.08233*, 2017.

[8] T. Mannucci, E.-J. van Kampen, C. de Visser, and Q. Chu, "Safe exploration algorithms for reinforcement learning controllers," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 4, pp. 1069–1081, 2017.

[9] J. Lin, L. Wang, F. Gao, S. Shen, and F. Zhang, "Flying through a narrow gap using neural network: an end-to-end planning and control approach," *arXiv preprint arXiv:1903.09088*, 2019.

[10] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.

[11] Y. Guo, J. Choi, M. Moczulski, S. Bengio, M. Norouzi, and H. Lee, "Self-imitation learning via trajectory-conditioned policy for hard-exploration tasks," *arXiv*, pp. arXiv–1907, 2019.

[12] D. Shi, X. Dai, X. Zhang, and Q. Quan, "A practical performance evaluation method for electric multicopters," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1337–1348, 2017.

[13] P. Lu and E.-J. van Kampen, "Active fault-tolerant control for quadrotors subjected to a complete rotor failure," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4698–4703.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[16] 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.

[17] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in neural information processing systems*, 2015, pp. 2575–2583.

[18] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, pp. 2613–2621, 2010.

[19] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 2094–2100.

[20] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[21] J. Sharma, P.-A. Andersen, O.-C. Granmo, and M. Goodwin, "Deep q-learning with q-matrix transfer learning for novel fire evacuation environment," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[22] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint arXiv:1610.03518*, 2016.

[23] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[24] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[25] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[26] 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, 2019.

[27] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[28] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Conference on Robot Learning*. PMLR, 2020, pp. 1162–1176.