arXiv:2203.05360v1 [cs.RO] 10 Mar 2022

Deep Residual Reinforcement Learning based Autonomous Blimp Control

Yu Tang Liu^{1,2}, Eric Price^{2,1}, Michael J. Black¹ and Aamir Ahmad^{2,1}

Abstract-Blimps are well suited to perform long-duration aerial tasks as they are energy efficient, relatively silent and safe. To address the blimp navigation and control task, in previous work we developed a hardware and software-in-theloop framework and a PID-based controller for large blimps in the presence of wind disturbance. However, blimps have a deformable structure and their dynamics are inherently non-linear and time-delayed, making PID controllers difficult to tune. Thus, often resulting in large tracking errors. Moreover, the buoyancy of a blimp is constantly changing due to variations in ambient temperature and pressure. To address these issues, in this paper we present a learning-based framework based on deep residual reinforcement learning (DRRL), for the blimp control task. Within this framework, we first employ a PID controller to provide baseline performance. Subsequently, the DRRL agent learns to modify the PID decisions by interaction with the environment. We demonstrate in simulation that DRRL agent consistently improves the PID performance. Through rigorous simulation experiments, we show that the agent is robust to changes in wind speed and buoyancy. In real-world experiments, we demonstrate that the agent, trained only in simulation, is sufficiently robust to control an actual blimp in windy conditions. We openly provide the source code of our approach at https://github.com/ robot-perception-group/AutonomousBlimpDRL.

I. INTRODUCTION

Autonomous unmanned aerial vehicles (UAVs) are becoming increasingly popular for various tasks, such as search and rescue, pavload (medicine, food) delivery in difficult-tonot able to accomplish long-term missions due to their short battery life. The situation is opposite for the fixed-wings, which requires moving constantly to stay airborne. Therefore, for tasks involving long flight times, carrying more payload and hovering over a small region, the use of autonomous blimps is an attractive solution. However, autonomous blimp control remains a challenging problem, which we address in this paper using a learning based approach.

Classic blimp controller design usually relies on PID controllers [2], [3] and nonlinear control [4], [5]. PID struggles with plant nonlinearity, and nonlinear control methods require a dynamic model of the system which is often difficult to acquire (i.e. friction, wind, aerodynamic effect, etc.). Deep reinforcement learning (DRL), on the other hand, is a new



Fig. 1: Our autonomous blimp during a flight. Unlike common designs, our blimp has thrust vectoring which increases its agility. Inset: its gazebo model.

control framework that has achieved success in a variety of applications that present similar challenges [6], [7].

For blimp control, the knowledge of some physical parameters, such as friction, aerodynamic effect, etc., are not negligible but remain difficult to estimate. A model-free DRL approach is particularly useful in such a case as it allows an agent to learn a control policy without any pre-specified physics and without the need to estimate those parameters explicitly. However, training such a model-free DRL agent requires significant amount of data and computational resources. The trained agent could also often learn unexpected and unsafe maneuvers as it only exploits the reward function. For example, reach areas, aerial cinematography and wildlife monitoring an autonomous blimp agent can learn to fly backwards and [1]. Current solutions rely on quadcopters and fixed-wings. still receive a high reward, while in reality such behavior Although quadcopters can hover in a fixed position, they are is undesired as it can damage the hardware. Our insight to address these two issues is to leverage a classical model-free approach, e.g., PID, to constraint the policy search space. We do this by developing a novel framework based on deep residual RL (DRRL) [8] that combines the advantages of both classical control and reinforcement learning. The use of the classical method in this framework not only provides stability in training but also implicitly outlines a safe behavior for the agent, constraining the policy search space and avoiding undesired behaviors.

> The training process can also be unstable due to the partial observable nature of the environment, e.g. wind and buoyancy, and this effect is exacerbated by the time-delayed blimp dynamics. To address this issue, we integrate an LSTM (long short term memory) [9] layer in our policy model to reduce the effect of partial observability.

> Nevertheless, our DRRL framework still requires substantial training experience to derive a working RL policy. We address this problem by training the agent in a software-in-the-loop (SITL) simulation setup [10] and parallelizing it to accelerate

¹Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany. ²Institute for Flight Mechanics and Controls, The Faculty of Aerospace Engineering and Geodesy, University of Stuttgart, 70569 Stuttgart, Germany. (yutang.liu, eric.price, black)@tuebingen.mpg.de, aamir.ahmad@ifr.unistuttgart.de

this process.

necessary to i) address the issue of sim-to-real gap, and ii) maintain smoothness in the actuator commands. Thus, in simulation, we apply domain randomization during training to improve the robustness of the agent. To protect the actuator and reduce the effects of chattering, we only include the increment of the actuator command instead of the actuator command action space of the agent.

In summary, the novel contribution of this paper is a modelfree DRRL-based approach for autonomously controlling a large blimp in forward velocity, yaw and altitude, simultaneously, in outdoor moderate wind conditions. Through rigorous simulations we show that our method outperforms state-of-theart approaches based on a PID and is robust to different flight contexts, e.g., changes in wind speed and buoyancy. Finally, through real-world experiments, we demonstrate that using our approach we obtain a robust control policy that seamlessly generalizes to the real blimp.

II. RELATED WORK

Control methods for blimps and airships, which have similar control schemes, have been well studied [11]. Classic approaches usually rely on PID controllers [2], [3], [12], [13]. While being simple and robust, they often suffer from plant nonlinearity. To overcome this, advanced approaches have been developed using nonlinear control theory, such as inverse optimal tracking control [14], dynamic inversion control [4], backstepping control [15], robust control [16], and model predictive control [17]. However, optimal control usually requires an accurate dynamic model which can be difficult to acquire, while robust control handles parameter uncertainty by trading-off the performance. Another key drawback is the lack of any real-world experiments and validation in most of these works. The buoyancy of a real blimp can change significantly due to the fluctuations in temperature over short time periods. The weight distribution could also vary and thus reduce the altitude control performance. Unfortunately, these effects have not been addressed in any of the prior works so far.

On the other hand, recently there has been a surge of interest in applying RL to robotics [18]. The earliest works include Gaussian processes (GPs) for system identification of a blimp [19] and its combination with value iteration and Q-learning approaches [20], [21] for blimp altitude control only. Despite sample efficiency, GPs are hard to scale up with problem dimensions and demand higher computational resources. As a result, they are able to achieve success only on low dimensional tasks, such as 1-D altitude control, whereas in our approach we show that the agent can feasibly and successfully learn a 3-dimensional task (forward velocity, yaw and altitude control). DRL, on the other hand, leverages deep neural networks (NNs) for policy approximation. Thus, its policy class can be used for higher dimensional tasks. For example, authors in [22] train two DQN agents for rudder and elevator control of a blimp, respectively, and demonstrate better performance than a PID controller in simulation. The

main challenge with DRL, however, is the lack of sample Furthermore, to deploy the agent on the real blimp, it is efficiency. In order to scale up the DRL formulation with the problem dimension, a highly increased amount of environment interactions is needed by the agent. Other challenges include, but are not limited to, adapting a trained policy to real-world scenarios [23] and action smoothness [24]. Furthermore, issues such as partial observability, disturbances and noise could also lead to unexpected behaviors. As described in the introduction, itself (e.g., rotor acceleration instead of rotor speed) in the in our approach we address all these issues through our novel DRRL-based framework, training parallelization, domain randomization and action space design.

III. METHODOLOGY

In this section, we first describe the blimp and the MDP problem formulation. Then we introduce the main goal of the work, the *blimp control task* (Sec.III-C). The objective in the *blimp control task* is to navigate the blimp to any given waypoint within the space $L^3 m^3$, where L is the dimension of the bounding box. This is followed by our novel DRRL-based framework that describes our approach for the *blimp control* task. Subsequently, we describe yaw control task (Sec.III-E), where the aim is to control the blimp to a desired yaw angle with the tail rotor. The aim of this simplified task is to perform ablation study.

A. Preliminaries

We first briefly describe our blimp (complete details are in our previous work [10]), which has 8 actuators. The two main motors (thrusters), $m_{1,2}$, are attached to a servo, n_0 , which allows thrust vectoring. At the tail of the blimp, four fins, $f_{0:3}$, two positioned vertically and two horizontally, control yaw and pitch angle, respectively. There is a tail motor, m_0 , attached to the lower vertical fin, generating horizontal thrust allowing further yaw controllability. Therefore, the state vector of the actuators can be denoted as

$$s_t^{act} = (m_{(0:2)}, n_0, f_{(0:3)})_t \in \mathbb{R}^8,$$
(1)

B. Markov Decision Process

We consider the RL problem as an infinite horizon discrete time Markov Decision Process, M, defined by a tuple (S, A, P, R, γ) [25]. At any time step $t \in \mathbb{R}^+$ and state $s_t \in \mathbb{R}^S$, an agent draws an action a_t from a continuous action space $a \in \mathbb{R}^A$ given the policy distribution $a_t \sim$ $\pi_{\theta}(\cdot|s_t)$ parameterized by θ . The environment then samples the next state from an unknown transition distribution, i.e. $s_{t+1} \sim P(\cdot|s_t, a_t)$. A reward is received based on some reward function $r_t = R(s_t, a_t)$. Given the discount factor $\gamma \in [0,1)$, the goal of the agent is to obtain the optimal policy parameter θ that maximizes the expected value of the cumulative discounted reward (2),

$$\pi^* = \operatorname*{argmax}_{\pi_{\theta}} \mathbb{E}\left[\sum_{t}^{\infty} \gamma^t r_t | a_t \sim \pi(\cdot | s_t), s_{t+1} \sim P(\cdot | s_t, a_t)\right]$$
(2)

C. Blimp Control Task

We formulate the problem as a path following task as seen in previous works [22], [26], [27]. In this setting, an imaginary path reference is generated based on waypoints for the controller to follow. Casting the path following task as a DRL problem, in this section we derive the observation space and action space representation.

Since the blimp does not have a lateral movement control, we only consider longitudinal, altitude, and velocity control. This allows us to easily decompose the problem into planar, altitude, and velocity control. The objective of the planar control is to control the blimp to arrive at any waypoint in the xy-plane, the altitude control is to reach the desired z, and the velocity control is to track the desired velocity.



Fig. 2: The origin indicate the position of GPS sensor in NED coordinate.

Given the blimp position at $\overline{O} = (0, 0, 0)$ and velocity $v_o \in \mathbb{R}$, a target waypoint at $\overline{g} = (l_r, \psi_r, z_r)$ in body frame cylindrical coordinates with desired velocity $v_g \in \mathbb{R}$ (Fig.~2), the control objective of the planar control is the minimization of the relevant distance and yaw angle, i.e. $\min_{a \in A}(|l_r|, |\psi_r|)$. The objective of the altitude and velocity control is to minimize the relevant altitude and the relevant velocity, respectively, i.e., $\min_{a \in A}(|z_r|), \min_{a \in A}(|v_r|)$, where $v_r = v_g - v_o$.

We denote the velocity vector of the blimp as $\overline{V} = (u, v, w) \in \mathbb{R}^3$, and attitude (roll, pitch, yaw) as $\overline{\Phi} = (\phi, \theta, \psi) \in \mathbb{R}^3$. Assuming near zero lateral movement in the blimp (i.e. $v, \phi \simeq 0$), the velocity and pitch angle can be encoded by velocity magnitude $(v_o = ||(u, v, w)||_2)$ and the altitude velocity $(w = v_o \sin \theta)$, alone. Therefore, the base state vector is

$$s_t^{blimp} = (l_r, \psi_r, z_r, v_r, v_o, w)_t.$$
 (3)

We augmented the base state vector with additional components, based on the insights as explained below. It was observed that the training progress becomes more stable if yaw velocity, ω_{ψ} augmented to the base state vector. The airspeed sensor readings, $v_{air} \in \mathbb{R}$, were augmented to enhance robustness against the wind. To prevent overshoot when reaching a waypoint in the planar control task, we augmented ψ'_r , the relative yaw angle of the blimp with respect to the subsequent waypoint. Consequently, the extended state representation is

$$s_t^{blimp'} = (s^{blimp}, \omega_{\psi}, v_{air}, \psi'_r)_t.$$
(4)

D. Novel DRRL-based framework

Our DRRL framework consists of two controllers – a stability provider and a performance optimizer, respectively.



Fig. 3: DRRL control diagram. Note that input and output of the policy network and PID controllers are scaled and clipped to the range (-1, 1)

The classical approach offers stability guarantees and basic tracking performance which is the role usually played by a PID controller or a robust controller to enlarge region of attraction. Performance optimizer within this framework is a DRL agent that can learn to adjust the control decisions of the stability provider in order to maximize its own reward function. The control command from these two controllers are then mixed by a mixer $f^{mix}(\cdot)$, which we will described later. The overall structure is displayed in Fig. 3. In this paper, we choose PID controller as our stability provider for its simplicity and robustness. It integrates well with the DRL agent as it is also a model-free method. No dynamic model is required with this combination. Though its performance degrades quickly outside the tuned speed range, the system nevertheless remains stable and can still bring the blimp closer to the waypoint.

1) PID Controller: The PID command, $a_t^{pid} \in \mathbb{R}^4$ is determined as follows (5),

where $f^{pid}(x) = k_p x + k_i \int x + k_d \dot{x}$. Since it is difficult to design a PID-based servo control, we leave this completely for the DRL agent to control.

2) Observation and Action Space for the DRL agent: The full actuator state, s_t^{act} , is described in (1). Since we forbid differential thrust, symmetric actuators are always in the same state. Thus, we feedback only one of them (i.e. $m_1 = m_2$, $f_0 = f_1$, $f_2 = f_3$). The tail motor is controlled and observed together with bottom fin (i.e. $m_0 = f_2$). The reduced state of actuators is therefore defined as $s_t^{act'} = (m_1, n_0, f_{(0,2)})_t \in \mathbb{R}^4$. The full state s_t for the DRL formulation, as used in (2), is now obtained below as the concatenation of $s_t^{blimp'}$, $s_t^{act'}$, and a_t^{pid}

$$s_t = (s_t^{blimp'}, s_t^{act'}, a_t^{pid})$$
(6)

Note that all states are scaled to the range [-1, 1] and zeroinitialized. The RL command, $a_t^{RL} \in \mathbb{R}^4$, is chosen based on the state vectors. Then the joint action command, $a_t \in \mathbb{R}^4$, is simply the mixture of RL and PID actions.

$$a_t^{RL} \sim \pi(\cdot|s_t)$$

$$a_t = f^{mix}(a_t^{pid}, a_t^{RL})$$
(7)

hybrid mixer (8-10). Absolute mixer offers RL agent more authority and is expected to have the highest performance after convergence at the cost of performance drop during exploration. This property is reversed for the relative mixer. Since the absolute mixer is too aggressive and requires rigorous yaw command to control the tail motor, m_0 , together with tuning of the beta parameter, whereas the relative mixer is too conservative to change the system's inherent stability properties, we introduce a hybrid mix as an intermediate solution.

$$f_{abs}^{mix}(x,y) = (1-\beta)x + \beta y \tag{8}$$

$$f_{rel}^{mix}(x,y) = x(1+\beta y) \tag{9}$$

$$f_{hyb}^{mix}(x,y) = (1-\alpha)f_{abs}^{mix}(x,y) + \alpha f_{rel}^{mix}(x,y), \quad (10)$$

where $(\alpha, \beta) \in [0.0, 1.0]$. To reduce the effect of chattering in the actuator state, we avoid mapping joint command, a_t , to the actuator state directly. Instead, it is first mapped to the increment of actuator state $\delta s_t^{act'} \in \mathbb{R}^4$ by element-wise multiplication with a constant vector, $c \in \mathbb{R}^4$, and then update the actuator state. The process is described below in (11). This way, we prohibit sudden significant changes in actuator states. Since our electronic speed control filters small changes, the damage from chattering effect is almost diminished. The disadvantage of this approach is that the control agility is reduced due to an additional pole introduced at the origin.

$$\delta s_t^{act'} = c \odot a_t$$

$$s_{t+1}^{act'} \leftarrow s_t^{act'} + \delta s_t^{act'}$$
(11)

We summerized the overall control architecture in Fig. 3.

3) Reward Function: The navigation requires moving the robot in space by specifying a target position or following a sequence of waypoints. The reward function is defined by (12)

$$r_t = w_0 r_t^{success} + w_1 r_t^{track} + w_2 r_t^{act} + w_3 r_t^{bonus}, \quad (12)$$

where $w_{0:3} \in \mathbb{R}$. The agent receives a success reward, $r_t^{success}$, if the task is completed, i.e., a waypoint is successfully reached within a certain threshold ϵ . Tracking reward, r_t^{track} , indicates the tracking performance as defined in (13). Action reward, r_t^{act} , is defined to regularize actuator commands. Bonus reward, r_t^{bonus} , specifies additional desired control property of preventing overshoot.

$$r_t^{success} = \begin{cases} 1 & \text{if } d(s_{blimp}, s_{target}) \le \epsilon \\ 0 & \text{otherwise} \end{cases}$$

$$r_t^{track} = -i_0 |z_r| - i_1 |l_r| - i_2 |\psi_r| - i_3 |v_r|, \qquad (13)$$

$$r_t^{act} = -j_0 |m_0| - j_1 |m_1| - j_1 |m_2|, \qquad r_t^{bonus} = -k_0 |\psi_r'| / (1 + l_r),$$

where $d(s_{blimp}, s_{target})$ measures euclidean distance between the blimp and the target waypoint position's 2D projections on the ground plane. Parameters, $w_{0:3}, i_{0:3}, k_0 \in \mathbb{R}$, are derived following questions through our experiments. Does LSTM via manual tuning and $j_{0:1} \in \mathbb{R}$ approximate the energy consumption of the rotors. The bonus reward is designed to reduce overshoot by reducing the relative yaw angle to the DRRL agent improve the PID controller? How does the

We introduce 3 types of mixer: absolute, relative, and the next waypoint when the blimp approaches current target waypoint.

E. Yaw Control Task

Here, the agent observes yaw-related states and outputs a a PID controller. The objective is to minimize the relative yaw angle, i.e. $\min_{a \in A}(|\psi_r|)$. Concretely, the observation space is defined as $s_t^{yaw} = (\psi_r, \omega_{\psi}, a_{\psi}^{pid})_t$ and the action space $a_t^{yaw} = f^{mix}(\pi(\cdot|s_t^{yaw}), a_{\psi,t}^{pid})$ where $s_t^{yaw} \in \mathbb{R}^3$ and $a_{t}^{yaw} \in \mathbb{R}$. The actuator state update is described as in (11) with $c \in \mathbb{R}$. Different from (12), the reward function in this task, only combines success reward and tracking reward, or $r_t^{yaw} = w_0 r_t^{success} + w_1 r_t^{track}$. The task is considered as success if the agent's relative yaw angle is kept small for a certain period of time, as described in (14).

$$r_t^{success} = \begin{cases} 1 & \text{if } T(|\psi_r| \le \epsilon) \ge T_{success} \\ 0 & \text{otherwise} \end{cases}$$
(14)
$$r_t^{track} = -|\psi_r|,$$

where $T(\cdot)$ is a time counting function and $T_{success} \in \mathbb{R}^+$.

F. Training Setup

In this section, we describe the important factors attributed to the robustness of the trained policy. At the beginning of each training episode, several waypoints are sampled in the space) based on the position of previous waypoint. When the blimp reaches the current waypoint, it receives a success reward and activates the next waypoint. During training, observations and actions are injected with noise. Lastly, we apply domain randomization and sample new environment variable for each episode according to Table.1.

variable	range
wind in xy direction (m/s)	[-1.5, 1.5]
wind in z direction (m/s)	[-0.15, 0.15]
buoyancy (100%)	[0.9, 1.1]
freeflop angle (\cdot)	[0.0, 1.5]
collapse (\cdot)	[0.0, 0.02]
deflation rate (\cdot)	[0.0, 1.5]

TABLE 1: Freeflop angle, collapse, and deflation rate determine maximum fin decline angle, the stiffness of the cable that holds the fin, and the rigidity of the blimp hull respectively, which jointly affect fin decline angle and reduce the lift and drag force [10].

We train the policy network with PPO, which has achieved strong benchmark performance and training stability. To accelerate training, we parallelize the simulation and raise the speed up to 14 folds of the real-time. The architecture of our policy network (Fig. 4) includes an LSTM layer to reduce the impact of partial observability (e.g. wind, bouyancy, etc.).

IV. EXPERIMENT DESIGN AND SETUP

With the real world scenario in mind, we address the architecture reduce the impact of partial observability? What are the properties of the mixers in the DRRL framework? Can agent perform under the presence of disturbance, noise, and parameter uncertainty?

A. Experimental Setup and compared methods

We integrate our DRRL training environment in the ROS/Gazebo SITL simulation following the OpenAI-Gym framework. The PPO implementation is based on RLlib. The agent is trained on a single computer (AMD Ryzen Threadripper 3960X, 24x 3.8GHz, NVIDIA GeForce RTX 2080 Ti, 11GB). Our simulated blimp model is designed based on our real robotic blimp (see Fig. 1). The following methods are evaluated and compared to each other.

- DRRL agent: our proposed approach.
- PID: the PID described in Sec. III-D.1
- **Baseline**: the baseline is a cascade PID controller, well-tuned to the simulation environment. Our previous work [10] has shown that we could deploy it to the real world without tuning, which implies a reliable quality of the simulation and robustness of such approach. This controller directly controls the actuator to follow the velocity reference from a path planner instead of the waypoints (as used by the above 2 methods) and relies on an extended Kalman filter for state estimation and noise filtering.

B. Task Suite

In this section, we describe the design of the two control tasks that were introduced previously. The *Yaw control task* (III-E) is a simplified task to evaluate different design options. The goal is to acquire the best possible configuration to then train a near-optimal policy for *blimp control task* (III-C) within limited amount of time. To ensure reproducibility, the training experiments are conducted with 3 different seeds. Table 2 displays the parameters for both tasks.

1) Yaw Control Task: We carry out an empirical ablation study on training stability of DRRL agent with different PID controller, different policy, and mixer combination. We first use a PD and then a P-control, which correspond to a good and a poor PID controller, respectively, for this task. PD control has stability guarantee while P control is only marginally stable.



Fig. 4: Following [28], we initialize the output layer with small weights (e.g. $1e^{-12}$), apply *tanh* activation function, and integrate normalization layers to stabilize training. h_{t-1} : hidden state. r_{t-1} : previous reward. a_{t-1}^{RL} :previous action. v_t : estimated value from the critic.

2) Blimp Control Task: We design the DRRL agent for this task based on the conclusion from the ablation study of the *yaw control task*. Despite the difference in task complexity, the PPO agent hyperperameter remains identical. We first examine the training progress of the DRRL agent and compare to the PID controller. The training is performed 3 times with different seeds to ensure the reproducibility.

Then, we investigate its robustness and characteristics in different wind context w.r.t the PID and the Baseline methods. This comparison is performed on results averaged over 7 runs, each lasting for 30 minutes and for 2 desired trajectories (coil and square). Furthermore, these are subject to random uniformly sampled wind direction. The square trajectory consists of 4 waypoints and has 80 meters between each waypoint. The coil trajectory has 30 meter radius covered by 15 waypoints in total. Consecutive waypoints on it are separated by 45 degrees and 42.4m in their projection on the X-Y plane and by 2m in the Z direction. As the square trajectory has longer edges, it is easier to track it as compared to the coil. The coil trajectory is more challenging due to the shorter inter-waypoint distance. In this case, the blimp has to constantly slow down to control the yaw angle which can cause altitude loss.

We test the trained agent on a real blimp with 40 meters square trajectory. The real blimp has several different properties compared to the simulation, e.g. trim weight difference, buoyancy, maximum thrust etc. Many of these effects are not domain randomized during training. That is to say, it is a new flight context that the DRRL agent has not encountered before and thus it pose a great challenge for generalization.

V. EXPERIMENTAL EVALUATION

A. Yaw Control Task

Fig. 5a shows that the final performance of DRRL with LSTM architecture increases the performance of good/poor PID control by 47%/13%. Without LSTM, the improvement is only 33%/13%. The maximum performance drop during exploration is 7%/25% with LSTM and 31%/47% without LSTM. This result suggests that the LSTM is an important building block for our DRRL framework. It can effectively stabilize and accelerate the training progress and reduce the performance drop during training.

The properties of the mixer type can be observed through the training progress in Fig. 5b. Unsurprisingly, the absolute mixer has the largest performance growth and drop during training and achieves the highest final performance for both good and poor PIDs, since it grants the DRRL agent more control authority. On the contrary, the relative mixer neither improves nor degrades the PID performance. Equation (9) suggests that, as the agent control is dependent on the PID, the DRRL agent has no control authority if PID control is small. That is, when the yaw error is close to 0, P control and DRRL agent offer no control to reduce the angular velocity and overshoot the target angle. Consequently, the marginally stable system property remains unchanged. Lastly, the hybrid mix retains the properties from both mixers and achieves the intermediate performance as expected. Although absolute

Setup details				
Group	Name	Value		
PPO	learning rate schedule	$[1e^{-4}, 5e^{-6}]$		
	γ	.999		
	λ	.9		
	horizon	800		
	batch size	22,400		
	mini-batch size	2,048		
	sgd iterations	32		
	gradient clip	1		
Environment	simulation frequency [hz]	30		
	policy frequency [hz]	10		
	observation noise [%]	2		
	action noise [%]	5		
	L [m]	200		
Yaw Control	training time [day]	1		
	est. wall clock time [hr]	1.7		
	good PID (k_p, k_i, k_d)	(1, 0, .5)		
	poor PID (k_p, k_i, k_d)	(1, 0, 0)		
	(α, β)	(.5, .5)		
	reward scale	.1		
	T _{success} [s]	5		
	ε[%]	10		
	$w_{0:1}$	(1, 1)		
DI: C I	<i>c</i>	.1		
Blimp Control	training time [day]	28		
	est. wall clock time [hr]	48		
	k_{thrust}^{plu}	(.7, .01, .05)		
	k_{ψ}^{pia}	(.3, .003, .0075)		
	k_z^{pid}	(2, .02, 1.0)		
	(α, β)	(.5, .5)		
	reward scale	.05		
	ε [m]	5		
	$w_{0:3}$	(100, .9, .1, .1)		
	$i_{0:3}$	(.6, .2, .1, .1)		
	<i>j</i> 0:1	(.5, 1)		
	k_0			
	C	$(4 \ 4 \ 1 \ 05)$		

TABLE 2: horizon and γ are relatively large due to the long response time of the blimp. In the blimp control task, altitude tracking reward weight, i_0 , has greater value than other planar terms since altitude loss is not considered as part of the success reward. reward scale is tuned to keep value loss within the range [-1, 1] to reduce the side-effect from the gradient clip and stabilize training.

mixer appears to be the best design, it is important to note that it can reduce the training stability. This effect is amplified in higher dimensional space.

We draw the following conclusions from the yaw control experiment: 1) The LSTM plays an important role to stabilize and accelerate training. 2) The choice of the mixer has significant impact on the training stability and final performance. 3) In addition to the importance of LSTM and mixer, the design choice of the PID controller is also important. By including a derivative term boosts nearly 50% of the initial performance.

B. Blimp Control Task

Following the conclusion from the ablation study (V-A), we design the DRRL with LSTM policy and absolute mixer as well as with the hybrid mixer. Even though the absolute mixer appears to be the best configuration in simpler task, the agent with this mixer consistently failed to obtain any functional policy and got stuck in a local optima where the policy always commands maximum tail rotor and results in repeated rotation movement. The hybrid mixer, on the other hand, successfully



(a) LSTM policy stabilizes and accelerates the training progress and reduces the performance drop.



(b) Absolute mixer is the most aggressive as it provides the most performance growth as well as performance drop during exploration, and vice versa.

Fig. 5: Ablation study on the effect of LSTM and mixer type. Red dotted curve represent the average PID control performance. Note that it is not a coincidence that the initial performance of the agent is similar to PID since the agent output layer weights are initialized close to zero.

to grow steadily.

As demonstrated in Fig. 7, the agent successfully tracks both the square and coil trajectory, which implies that it does not overfit to any specific tracks and can generalize to any desired trajectory in the 3D space. The baseline trajectory is more consistent compare to the others. This is because EKF provides smoother state estimation while both the DRRL agent and PID control receive only noisy raw observations. On the other hand, although the trajectories of the DRRL agent and the PID controller look fairly similar, the DRRL agent has much less overshoots compared to both the other methods in the coil trajectory (Fig. 7b) as it can observe the subsequent waypoint. In the coil trajectory, we observe that the baseline struggles following denser waypoints. To prevent altitude loss, the baseline applies a constraint on the maximum yaw angular rate, which limits the maximum turn radius and reduces its agility.

In Tab. 3 we summarize the robustness tests and comparison of methods over different trajectory types, wind speeds and buoyancy. The DRRL agent receives highest amount of 'total reward' in 4 out of 6 experiment combinations. Higher 'success reward' implies that the agent can track more waypoints within the total time span. During experiments, although the desired velocity is 3m/s, the baseline seems to achieve only 2m/s. As a consequences, it traverses less total distance and receives less amount of success reward. In terms of tracking reward, the baseline significantly outperform others. Since the tracking reward is dominated by the altitude loss, this suggests that stabilizes the training progress (Fig. 6). It reaches 60% of the the baseline can keep track of the altitude better than PID final performance within the first 2000 episodes and continues and DRRL agent. In the coil trajectory, although the PID and



Fig. 6: The blimp control task training progress with our DRRL integration. The red dotted line indicates the PID performance. The hybrid mixer provides better training stability compare to absolute mixer.

the DRRL agent can follow the trajectory well, we observe significant loss in altitude. PID control does not have sufficient speed to maintain the altitude and continues to sink, while the DRRL agent relies on the thrust vectoring to loiter at the desired altitude. Similarly, reducing the buoyancy can impair the altitude control of the RL agent and PID control. The baseline, while being worst at overall waypoint tracking, tracks the altitude well. It achieves this via thrust vectoring and because, by design, the baseline's primary task includes maintaining the altitude.



(a) The square trajectory has long edge and sharp corner. It is served to test tracking performance and overshoot reduction.



(b) The coil trajectory has shorter distance between the waypoints. It is served for agility and altitude control test.

Fig. 7: Behavior comparison of different controllers in no wind condition.

C. Real World Test

The result of the real test flight is displayed in Fig.8 and Table4. We reduce the square size to 40 meters as oppose to 80 meter in simulation due to the limitation of the test field. The wind speed was measured in average 6m/s which was 4 times more than the DRRL agent had experienced in the simulation. Nevertheless, the DRRL agent could still hold its own position under the gusts and successfully reached several waypoints. Note that the row of trajectory snapshots in Fig. 8 show only a part of the complete trajectory. We also provide

Robustness Evaluation							
Trajectory	wind $[m/s]$	controller	r	r ^{success}	r ^{track}	r ^{act}	r ^{bonus}
square	0	DRRL	0.0017	0.0027	-0.2143	-0.4308	-0.0107
-		PID	-0.0055	0.0030	-0.4073	-0.4177	-0.0112
		Baseline	0.0025	0.0020	-0.1050	-0.5896	-0.0136
	0.5	DRRL	0.0058	0.0036	-0.2128	-0.5106	-0.0112
		PID	-0.0050	0.0029	-0.3901	-0.4186	-0.0111
		Baseline	0.0020	0.0019	-0.1101	-0.5879	-0.0128
	1	DRRL	0.0022	0.0026	-0.1927	-0.4304	-0.0111
		PID	-0.0077	0.0022	-0.3601	-0.4437	-0.0133
		Baseline	-0.0002	0.0016	-0.1123	-0.5849	-0.0141
coil	0	DRRL	0.0189	0.0082	-0.4253	-0.6330	-0.0100
		PID	0.0282	0.0106	-0.4890	-0.5758	-0.0118
		Baseline	-0.0032	0.0013	-0.1501	-0.5795	-0.0207
	0.5	DRRL	0.0451	0.0118	-0.2448	-0.6172	-0.0118
		PID	0.0363	0.0118	-0.4435	-0.5733	-0.0124
		Baseline	0.0017	0.0021	-0.1256	-0.5786	-0.0218
	1	DRRL	0.0397	0.0108	-0.2504	-0.6211	-0.0124
		PID	0.0245	0.0089	-0.3737	-0.6074	-0.0122
		Baseline	-0.0014	0.0016	-0.1373	-0.5780	-0.0192
	buoyancy[%]						
square	0.93	DRRL	-0.0001	0.0026	-0.2466	-0.4369	-0.0117
1		PID	-0.0009	0.0025	-0.4232	-0.4231	-0.0113
		Baseline	0.0026	0.0019	-0.1049	-0.5899	-0.0137
	1.07	DRRL	0.0043	0.0030	-0.1861	-0.4262	-0.0114
		PID	-0.0037	0.0030	-0.3707	-0.4226	-0.0113
		Baseline	0.0022	0.0018	-0.1044	-0.5816	-0.0135

TABLE 3: The wind has the same speed for all runs in one trial but the directions are uniformly sampled. Rewards presented are averaged over all timesteps and trials. Higher reward is preferred. The overall performance is indicated by the reward column, r.







(b) Real world DRRL agent's flight.

Fig. 8: Real World Experiments: Although the flight context is the same for both real-world flights, the wind gusts arrive at different times. Therefore, the baseline is only provided as a reference instead of a comparison. The top rows in each of the above figures show a part of the trajectory taken by the blimp to reach subsequent waypoints.

the baseline as a reference. But they are not comparable since the gusts were strong and arrived irregularly, and hence, the method that received more gusts would obtain less reward.

Real Flight Evaluation						
Trajectory	controller	r	r ^{success}	r^{track}	r^{act}	r^{bonus}
square	DRRL	0.0027	0.0022	-0.0991	-0.3969	-0.0155
	Baseline	0.0006	0.0013	-0.0961	-0.3445	-0.0109

TABLE 4: Averaged rewards by DRRL agent and baseline approaches during real-world flights.

VI. CONCLUSIONS

In this work, we presented a novel framework based on DRRL for the blimp control task. It leverages an RL agent to improve the basic PID control performance through interaction with the environment. We presented and evaluated several techniques to stabilize the training progress and enhance the robustness of the trained RL agent, e.g., domain randomization, LSTM layer, and a hybrid mixer in the DRRL framework. Extensive robustness tests were conducted that demonstrated the DRRL agent's capability to improve the PID performance and outperform it as well as another baseline approach. Through real blimp flights in outdoor environment and windy [18] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in conditions, we demonstrated that the trained policy could even generalize to a real scenario without any modification.

REFERENCES

- [1] L. F. Gonzalez, G. A. Montes, E. Puig, S. Johnson, K. Mengersen, and K. J. Gaston, "Unmanned aerial vehicles (uavs) and artificial intelligence revolutionizing wildlife monitoring and conservation," Sensors (Basel, Switzerland), vol. 16, no. 1, p. 97, Jan 2016, 26784196[pmid]. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/26784196
- [2] J. Azinheira, P. Rives, J. Carvalho, G. Silveira, E. de Paiva, and S. Bueno, "Visual servo control for the hovering of all outdoor robotic airship," in Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), vol. 3, May 2002, pp. 2787-2792 vol.3.
- [3] T. Takaya, H. Kawamura, Y. Minagawa, M. Yamamoto, and A. Ohuchi, "Pid landing orbit motion controller for an indoor blimp robot," Artificial Life and Robotics, vol. 10, no. 2, pp. 177-184, Nov 2006. [Online]. Available: https://doi.org/10.1007/s10015-006-0385-9
- [4] A. Moutinho and J. Azinheira, "Stability and robustness analysis of the aurora airship control system using dynamic inversion," in Proceedings of the 2005 IEEE International Conference on Robotics and Automation, April 2005, pp. 2265-2270.
- [5] S. Q. Liu, Y. J. Sang, and J. F. Whidborne, "Adaptive sliding-modebackstepping trajectory tracking control of underactuated airships, Aerospace Science and Technology, vol. 97, p. 105610, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/ S1270963819310181
- [6] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry, "Autonomous helicopter flight via reinforcement learning," in Proceedings of the 16th International Conference on Neural Information Processing Systems, ser. NIPS'03. Cambridge, MA, USA: MIT Press, 2003, p. 799-806.
- [7] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning,' Nature, vol. 588, no. 7836, pp. 77-82, Dec 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2939-8
- [8] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual Policy Learning," arXiv e-prints, p. arXiv:1812.06298, Dec. 2018.
- [9] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735
- [10] E. Price, Y. T. Liu, M. J. Black, and A. Ahmad, "Simulation and control of deformable autonomous airships in turbulent wind," in 16th International Conference on Intelligent Autonomous System (IAS), Jun. 2021.

- [11] Y. Liu, Z. Pan, D. Stirling, and F. Naghdy, "Control of autonomous airship," in 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dec 2009, pp. 2457-2462.
- [12] H. Zhang and J. Ostrowski, "Visual servoing with dynamics: control of an unmanned blimp," in Proceedings 1999 IEEE International Conference on Robotics and Automation, vol. 1, May 1999, pp. 618-623 vol.1.
- 13] S. van der Zwaan, A. Bernardino, and J. Santos-Victor, "Vision based station keeping and docking for an aerial blimp," in Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), vol. 1, Oct 2000, pp. 614-619 vol.1.
- [14] T. Fukao, T. Kanzawa, and K. Osuka, "Inverse optimal tracking control of an aerial blimp robot," in Proceedings of the Fifth International Workshop on Robot Motion and Control, 2005. RoMoCo '05., June 2005, pp. 193-198.
- [15] J. R. Azinheira and A. Moutinho, "Hover control of an uav with backstepping design including input saturations," IEEE Transactions on Control Systems Technology, vol. 16, no. 3, pp. 517-526, May 2008.
- [16] L. Cheng, Z. Zuo, J. Song, and X. Liang, "Robust three-dimensional pathfollowing control for an under-actuated stratospheric airship," Advances in Space Research, vol. 63, no. 1, pp. 526-538, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S027311771830704X
- [17] H. Fukushima, R. Saito, F. Matsuno, Y. Hada, K. Kawabata, and H. Asama, "Model predictive control of an autonomous blimp with input and output constraints," in 2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, Oct 2006, pp. 2184-2189.
- robotic applications: a comprehensive survey," Artificial Intelligence Review, vol. 55, no. 2, pp. 945-990, Feb 2022. [Online]. Available: https://doi.org/10.1007/s10462-021-09997-9
- [19] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in Proceedings 2007 IEEE International Conference on Robotics and Automation, April 2007, pp. 742-747.
- [20] A. Rottmann and W. Burgard, "Adaptive autonomous control using online value iteration with gaussian processes," in 2009 IEEE International Conference on Robotics and Automation, May 2009, pp. 2106-2111.
- [21] A. Rottmann, C. Plagemann, P. Hilgers, and W. Burgard, "Autonomous blimp control using model-free reinforcement learning in a continuous state and action space," in 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2007, pp. 1895-1900.
- C. Nie, Z. Zheng, and M. Zhu, "Three-dimensional path-following [22] control of a robotic airship with reinforcement learning," International Journal of Aerospace Engineering, vol. 2019, p. 7854173, Mar 2019. [Online]. Available: https://doi.org/10.1155/2019/7854173
- [23] Y. Zhang, T. Liu, M. Long, and M. Jordan, "Bridging theory and algorithm for domain adaptation," in Proceedings of the 36th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09-15 Jun 2019, pp. 7404-7413. [Online]. Available: https://proceedings.mlr.press/v97/zhang19i.html
- S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko, "Regularizing action policies for smooth control with reinforcement learning," in 2021 IEEE International Conference on Robotics and Automation (ICRA), May 2021, pp. 1810-1816.
- [25] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. Bradford Books, February 1998.
- [26] J. Rao, Z. Gong, J. Luo, and S. Xie, "A flight control and navigation system of a small size unmanned airship," in IEEE International Conference Mechatronics and Automation, 2005, vol. 3, July 2005, pp. 1491-1496 Vol. 3.
- [27] H. Saiki, T. Fukao, T. Urakubo, and T. Kohno, "Hovering control of outdoor blimp robots based on path following," in 2010 IEEE International Conference on Control Applications, Sep. 2010, pp. 2124-2129.
- [28] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, "What matters for on-policy deep actor-critic methods? a large-scale study," in International Conference on Learning Representations, 2021. [Online]. Available: https:// //openreview.net/forum?id=nIAxjsniDzg