


Controlled gliding and perching through deep-reinforcement-learning

Guido Novati,¹ L. Mahadevan,^{2,3} and Petros Koumoutsakos ^{1,*}

¹*Computational Science and Engineering Laboratory, Clausiusstrasse 33, ETH Zürich, CH-8092, Switzerland*

²*John A. Paulson School of Engineering and Applied Sciences, Harvard University,
Cambridge, Massachusetts 02138, USA*

³*Department of Organismic and Evolutionary Biology, Department of Physics, Harvard University,
Cambridge, Massachusetts 02138, USA*



(Received 15 September 2018; published 6 September 2019)

Controlled gliding is one of the most energetically efficient modes of transportation for natural and human powered fliers. Here we demonstrate that gliding and landing strategies with different optimality criteria can be identified through deep-reinforcement-learning without explicit knowledge of the underlying physics. We combine a two-dimensional model of a controlled elliptical body with deep-reinforcement-learning (D-RL) to achieve gliding with either minimum energy expenditure, or fastest time of arrival, at a predetermined location. In both cases the gliding trajectories are smooth, although energy/time optimal strategies are distinguished by small/high frequency actuations. We examine the effects of the ellipse's shape and weight on the optimal policies for controlled gliding. We find that the model-free reinforcement learning leads to more robust gliding than model-based optimal control strategies with a modest additional computational cost. We also demonstrate that the gliders with D-RL can generalize their strategies to reach the target location from previously unseen starting positions. The model-free character and robustness of D-RL suggests a promising framework for developing robotic devices capable of exploiting complex flow environments.

DOI: [10.1103/PhysRevFluids.4.093902](https://doi.org/10.1103/PhysRevFluids.4.093902)

I. INTRODUCTION

Gliding is an intrinsically efficient motion that relies on the body shape to extract momentum from the air flow, while performing minimal mechanical work to control attitude. The diversity of animal and plant species that have independently evolved the ability to glide is a testament to the efficiency and usefulness of this mode of transport. Well known examples include birds that soar with thermal winds [1], fish that employ burst and coast swimming mechanisms, and plant seeds, such as the samara [2], that spread by gliding. Furthermore, arboreal animals that live in forest canopies often employ gliding to avoid earth-bound predators, forage across long distances, chase prey, and safely recover from falls [3]. Characteristic of gliding mammals is the membrane (patagium) that develops between legs and arms. When extended, the patagium transforms the entire body into a wing, allowing the mammal to stay airborne for extended periods of time [4]. Analogous body adaptations have developed in species of lizards [5] and frogs [6].

Most surprisingly, gliding has developed in animal species characterized by blunt bodies lacking specialized lift-generating appendages. The *Chrysopelea* genus of snakes have learned to launch themselves from trees, flatten and camber their bodies to form a concave cross-section, and perform sustained aerial undulations to generate enough lift to match the gliding performance of mammalian gliders [7]. Wingless insects such as tropical arboreal ants [8] and bristletails [9] are able to glide

*petros@ethz.ch

when falling from the canopy to avoid the possibly flooded or otherwise hazardous forest understory. During descent these canopy-dwelling insects identify the target tree trunk using visual cues [10] and orient their horizontal trajectory appropriately.

Most bird species alternate active flapping with gliding, to reduce physical effort during long-range flight [11,12]. Similarly, gliding is an attractive solution to extend the range of micro air vehicles (MAVs). MAV designs often rely on arrays of rotors (i.e., quadcopters) due to their simple structure and due to the existence of simplified models that capture the main aspects of the underlying fluid dynamics. The combination of these two features allows finding precise control techniques [13,14] to perform complex flight maneuvers [15,16]. However, the main drawback of rotor-propelled MAVs is their limited flight-times, which restricts real-world applications. Several solutions for extending the range of MAVs have been proposed, including techniques involving precise perching maneuvers [17], mimicking flying animals by designing a flier [18] capable of gliding, and exploiting the upward momentum of thermal winds to soar with minimal energy expense [19,20].

Here we study the ability of falling blunt-shaped bodies, lacking any specialized feature for generating lift, to learn gliding strategies through reinforcement learning [21–23]. The goal of the RL agent is to control its descent toward a set target landing position and perching angle. The agent is modeled by a simple dynamical system describing the passive planar gravity-driven descent of a cylindrical object in a quiescent fluid. The simplicity of the system is due to a parametrized model for the fluid forces which has been developed through simulations and experimental studies [24–26]. Following the work of Ref. [27], we augment the original, passive dynamical system with active control. We identify optimal control policies through reinforcement learning, a semisupervised learning framework that has been employed successfully in a number of flow control problems [19, 28–31]. We employ recent advances in coupling RL with deep neural networks [32]. These so-called deep-reinforcement-learning algorithms have been shown in several problems to match and even surpass the performance of classical control algorithms. For example, in Ref. [33] it was shown that D-RL outperforms an interior point method for constraint optimization, using pre-computed actions from a simulation to land unmanned aerial vehicles in an experimental setting. Indeed, D-RL appears to be a very promising control strategy for perching in UAVs, a critical energy saving process [34].

The paper is organized as follows: we describe the model of an active, falling body in Sec. II and frame the problems in terms of reinforcement learning in Sec. III. In Sec. III A we present a high-level description of the RL algorithm and describe the reward shaping combining the time/energy cost with kinematic constraints as described in Sec. III B. We explore the effects of the weight and shape of the agent’s body on the optimal gliding strategies in Sec. IV. In Secs. V and VI we analyze the RL methods by comparing RL policies to the trajectories found with optimal control (OC) [27], by varying the problem formulation and by comparing RL algorithms.

II. MODEL

The motion of falling slender elliptical bodies involves rich hydrodynamics which have inspired much research [24,25,35–38]. Depending on its size, shape, density, and initial orientation, the ellipse’s descent follows motion patterns classified as steady-fall, fluttering (side-to-side oscillation of the horizontal motion), tumbling (repeated rotation around the out-of-plane direction), or transitional patterns. The aforementioned studies show that these descent patterns can be qualitatively described by a simple model consisting of ordinary differential equations (ODEs) for the ellipse’s translational and rotational degrees of freedom (see Fig. 1). This ODE-based model relies on the assumption that the descent is essentially planar (i.e., the axis of rotation of the elliptical body is orthogonal to its velocity), and it has been derived on the basis of inviscid fluid dynamics [39] with a corrective parametric model to account for viscous effects. We employ here the dimensionless form of the ODEs as originally proposed by Ref. [24] for the gravity-driven descent of a cylindrical ellipse of density ρ_s with semiaxes a and b in a quiescent fluid of density ρ_f . The nondimensionalization

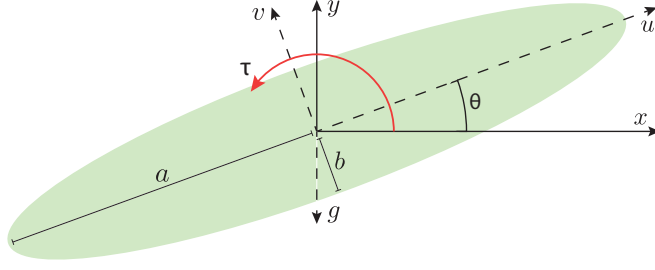


FIG. 1. Schematic of the system and degrees of freedom modeled by Eqs. (1) to (6). The position (x, y) of the center of mass is defined in a fixed frame of reference, θ is the angle between the x axis and the major axis of the ellipse, and the velocity components u and v are defined in the moving and rotating frame of reference of the ellipse.

is performed with the length scale a and the velocity scale $\sqrt{(\rho_s/\rho_f - 1)gb}$, obtained from the balance between gravity and quadratic drag [36]. The resulting set of equations depends on the dimensionless parameters $\beta = b/a$ and $\rho^* = \rho_s/\rho_f$, with $I = \beta\rho^*$ the nondimensional moment of inertia:

$$(I + \beta^2)\dot{u} = (I + 1)vw - \Gamma v - \sin\theta - Fu, \quad (1)$$

$$(I + 1)\dot{v} = -(I + \beta^2)uw + \Gamma u - \cos\theta - Fv, \quad (2)$$

$$\frac{1}{4}[I(1 + \beta^2) + \frac{1}{2}(1 - \beta^2)^2]\dot{w} = -(1 - \beta^2)uv - M + \tau, \quad (3)$$

$$\dot{x} = u \cos\theta - v \sin\theta, \quad (4)$$

$$\dot{y} = u \sin\theta + v \cos\theta, \quad (5)$$

$$\dot{\theta} = w. \quad (6)$$

Here $u(t)$ and $v(t)$ denote the projections of the velocity along the ellipse's semi-axes, $(x(t), y(t))$ is the position of the center of mass, $\theta(t)$ is the angle between the major semi-axis and the horizontal direction, and $w(t)$ is the angular velocity. Closure of the above system requires expressions for the fluid forces F_u, F_v , the torque M , and the circulation Γ . A series of studies [24–26,38], motivated by studying the motion of falling cards, have used experiments and numerical simulations to obtain a self-consistent and nondimensional parametric model for these quantities:

$$F = \frac{1}{\pi} \left[A - B \frac{u^2 - v^2}{u^2 + v^2} \right] \sqrt{u^2 + v^2}, \quad (7)$$

$$M = 0.2(\mu + \nu \|w\|)w, \quad (8)$$

$$\Gamma = \frac{2}{\pi} \left[C_R w - C_T \frac{uv}{\sqrt{u^2 + v^2}} \right]. \quad (9)$$

Here, $A = 1.4$, $B = 1$, $\mu = \nu = 0.2$, $C_T = 1.2$, and $C_R = \pi$ are nondimensional constants obtained from fitting the viscous drag and circulation to those measured from numerical simulations. Wang and coauthors show that such parametrization, while it may not be sufficient to precisely quantify the viscous fluid forces, enables the ODE model to qualitatively describe the regular and irregular motion patterns, such as fluttering and tumbling, associated with falling elliptical bodies. The values chosen for this study were obtained (by Wang and coauthors) to approximate the fluid forces at intermediate Reynolds numbers $O(10^3)$ [27], consistent with that of gliding ants [8]. The authors of Ref. [27] focus their optimal control study to $\beta = 0.1$ and $I = 20$, which represents the inertia of both the ant's body and the fluid contained inside the body-fitted ellipse. However, independently from β and I , the parametrized model for the viscous fluid forces will be consistent with Reynolds numbers $\text{Re} \approx O(10^3)$.

In active gliding, we assume that the gravity-driven trajectory can be modified by the agent modulating the dimensionless torque τ in Eq. (3), introduced by Ref. [27] as a minimal representation of the ability of gliding ants to guide their fall by rotating their hind legs [40]. Alternatively, the control torque could be achieved by the glider deforming its body to displace the center of mass or by extending its limbs to deflect the incoming flow. This leads to a natural question: How should the active torque be varied in time for the ant to achieve a particular task such as landing and perching at a particular location with a particular orientation, subject to some constraints, e.g., optimizing time, minimizing power consumption, or maximizing accuracy. This is a problem considered by Ref. [27] in an optimal control framework. Here we consider an alternative approach inspired by how organisms might learn, that of reinforcement learning [23].

III. REINFORCEMENT LEARNING FOR LANDING AND PERCHING

The tasks of landing and perching are achieved by the falling body by employing a reinforcement learning framework [21–23] to identify their control actions.

In the following we provide a brief overview of the RL framework in the context of flow control and outline the algorithms used in the present study. Reinforcement learning is a semisupervised learning framework with a broad range of applications ranging from robotics [20,41], games [32,42], and flow control [29]. In RL, the control actor (termed “agent”) interacts with its environment by sampling its states (s), performing actions (a), and receiving rewards (r). At each time step (t) the agent performs the action and the system is advanced in time for Δt , before the agent can observe its new state s_{t+1} , receive a scalar reward r_{t+1} , and choose a new action a_{t+1} . The agent infers a policy $\pi(s, a)$ through its repeated interactions with the environment so as to maximize its long term rewards. The optimal policy $\pi^*(s, a)$ is found by maximizing the expected utility:

$$J = \mathbb{E}_\pi \left[\sum_{t=0}^T \gamma^t r_{t+1} \right]. \quad (10)$$

Once the optimal policy has been inferred the agent can interact autonomously with the environment without further learning.

The RL framework solves discrete-time Markov decision processes (MDP) which are defined by the one step dynamics of the environment, described by the probability \mathcal{P} of any next possible state s' given the current state and action pair (s, a) , and by the probability distribution over reward values \mathcal{R} defined as [23]

$$\mathcal{P} = P[s_{t+1}=s' | s_t=s, a_t=a], \quad \mathcal{R} = P[r_{t+1}=r | s_t=s, a_t=a, s_{t+1}=s']. \quad (11)$$

Given the current pair (s, a) , the Markov assumption refers to the conditional independence of these probability distributions from all previous states and actions. The value function for each state $V^\pi(s)$ provides an estimate of future rewards given a certain policy π . For an MDP we define the state value function $V^\pi(s)$ as

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | s_0 = s \right], \quad (12)$$

where \mathbb{E}_π denotes expected value for the agent when it performs the policy π and γ is a discount factor ($\gamma < 1$) for future rewards. The action-value function $Q^\pi(s, a)$ satisfying the celebrated Bellman equation [43] can be written as

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | s_0 = s, a_0 = a \right]. \quad (13)$$

The state-value is the expected action-value under the policy π : $V^\pi(s) = \mathbb{E}_{a \sim \pi} [Q^\pi(s, a)]$.

The Bellman equation and value functions are also intrinsic to dynamic programming (DP) [21]. RL is inherently linked to DP and it is often referred to as approximate DP. The key difference between RL and dynamic programming is that RL does not require an accurate model of the environment and infers the optimal policy by a sampling process. Moreover, RL can be applied to non-MDPs. As such RL is in general more computationally intensive than DP and optimal control but at the same time can handle black-box problems and is robust to noisy and stochastic environments. With the advancement of computational capabilities RL is becoming a valid alternative to optimal control and other machine learning strategies [44] for fluid mechanics problems.

The functions $V^\pi(s)$, $Q^\pi(s, a)$, $\pi(s, a)$ may be approximated by tables that reflect discrete sets of states and actions. Such approaches have been used with success in fluid mechanics applications [20,28,29] by discretizing the state-action variables. However, flow environments are often characterized by nonlinear and complex dynamics which may not be amenable to binning in tabular approximations. Conversely continuous approximations, such as those employed here in, have been shown to lead to robust and efficient learning policies [31,45]. Moreover, since the dynamics of the system are described by a small number of ODEs that can be solved at each instant to determine the full state of the system, in contrast with the need to solve the full Navier-Stokes equations [31], the present control problem satisfies the Markov assumption. Therefore, we can use a feedforward network (NN) rather than recurrent neural networks (RNN) as policy approximators [46]. The use of RNN to solve partially-observable problems (non-MDPs) is explored in Sec. VI.

We follow the problem definition of a glider attempting to land at a target location as described by Ref. [27]. We consider an agent, initially located at $x_0 = y_0 = \theta_0 = 0$, that has the objective of landing at a target location $x_G = 100$, $y_G = -50$ [expressed in units of the ellipse major semi-axis (a)] with perching angle $\theta_G = \frac{\pi}{4}$. For most results we consider an elliptical glider with nondimensional moment of inertia $I \gg 1$, so that the amplitude of the fluttering motion is much smaller than the distance from the target [24]. In such cases the spread of landing locations for the uncontrolled ellipse is of order $O(1)$, much smaller than the distance $\|x_G - x_0\|$.

By describing the trajectory with the model outlined in Sec. II, the state of the agent is completely defined at every time step by the state vector $s := \{x, y, \theta, u, v, w\}$. With a finite time interval $\Delta t = 0.5$, the agent is able to observe its state s_t and, based on the state, samples a stochastic control policy π to select an action $a_t \sim \pi(a|s_t)$. The actuation period Δt is of the same order of magnitude as the tumbling frequency for an ellipse with $I \approx O(1)$ [25] and is analyzed in more detail in Sec. VI. We consider continuous-valued controls defined by Gaussian policies which allows for fine-grained corrections (in contrast to the usually employed discretized controls). The action determines the constant control torque $\tau_t = \tanh(a_t) \in \{-1, 1\}$ exerted by the agent between time t and $t + \Delta t$. To provide enough diversity of initial conditions to the RL method, we initialize $x(0) \sim \mathcal{U}[-5, 5]$ and $\theta(0) \sim \mathcal{U}[-\pi/2, \pi/2]$. In the case of perching and landing, each episode is terminated when at some terminal time T the agent touches the ground $y_T = -50$. Because the gravitational force acting on the glider ensures that each trajectory will last a finite number of steps we can avoid the discount factor and set $\gamma = 1$.

A. Off-policy actor-critic

We solve the RL problem with a novel off-policy actor-critic algorithm named RACER [47]. The algorithm relies on training a neural network (NN), defined by weights \mathbf{w} , to obtain a continuous approximation of the policy $\pi^{\mathbf{w}}(a|s)$, the state value $V^{\mathbf{w}}(s)$ and the action value $Q^{\mathbf{w}}(s, a)$. The network receives as input $s = \{x, y, \theta, u, v, w\} \in \mathbb{R}^6$ and produces as output the set of parameters $\{m^{\mathbf{w}}, \sigma^{\mathbf{w}}, V^{\mathbf{w}}, l^{\mathbf{w}}\} \in \mathbb{R}^4$ that are further explained below. The policy $\pi^{\mathbf{w}}(a|s)$ for each state is approximated with a Gaussian having a mean $m^{\mathbf{w}}(s)$ and a standard deviation $\sigma^{\mathbf{w}}(s)$:

$$\pi^{\mathbf{w}}(a|s) = \frac{1}{\sqrt{2\pi}\sigma^{\mathbf{w}}(s)} \exp\left[-\frac{1}{2}\left(\frac{a - m^{\mathbf{w}}(s)}{\sigma^{\mathbf{w}}(s)}\right)^2\right]. \quad (14)$$

The standard deviation is initially wide enough to adequately explore the dynamics of the system. In turn, we also suggest a continuous estimate of the state-action value function $[Q^\pi(s, a)]$. Here, rather than having a specialized network, which includes in its input the action a , we propose computing the estimate $Q^w(s, a)$ by combining the network’s state value estimate $V^w(s)$ with a quadratic term with vertex at the mean $m^w(s)$ of the policy:

$$Q^w(s, a) = V^w(s) - \frac{1}{2}l^w(s)^2[a - m^w(s)]^2 + \mathbb{E}_{a' \sim \pi} \left\{ \frac{1}{2}l^w(s)^2[a' - m^w(s)]^2 \right\}, \quad (15)$$

$$= V^w(s) - \frac{1}{2}l^w(s)^2\{[a - m^w(s)]^2 - [\sigma^w(s)]^2\}. \quad (16)$$

This definition ensures that $V^w(s) = \mathbb{E}_{a \sim \pi}[Q^w(s, a)]$. Here $l^w(s)$ is an output of the network describing the rate at which the action value decreases for actions farther away from $m^w(s)$. This parametrization relies on the assumption that for any given state $Q^w(s, a)$ is maximal at the mean of the policy.

The learning process advances by iteratively sampling the dynamical system to assemble a set of training trajectories $\mathcal{B} = \{S_1, S_2, \dots\}$. A trajectory $S = \{o_0, o_1, \dots, o_T\}$ is sequence of observations o_t . An observation is defined as the collection of all information available to the agent at time t : the state s_t , the reward r_t , the current policy $\mu_t = \{m_t, \sigma_t\}$ and the sampled action a_t . Here we made a distinction between the policy μ_t executed at time t and $\pi^w(a|s_t)$, because when the data is used for training, the weights w of the NN might change, causing the current policy for state s to change. For each new observation o_t from the environment, a number B of observations are sampled from the dataset \mathcal{B} . Finally, the network weights are updated through back-propagation of the policy (g_π) and value function gradients (g_Q).

The policy parameters $m^w(s)$ and $\sigma^w(s)$ are improved through the policy gradient estimator [48]:

$$g_\pi = \sum_{t=1}^B \frac{\pi^w(a_t|s_t)}{\mu_t(a_t|s_t)} [\hat{Q}(s_t, a_t) - V^w(s_t)] \nabla_w \log \pi^w(a_t|s_t), \quad (17)$$

where $\hat{Q}(s_t, a_t)$ is an estimator of the action value. A key insight from policy-gradient-based algorithms is that the parametrized $Q^w(s_t, a_t)$ cannot safely be used to approximate on-policy returns, due to its inaccuracy during training [46]. However, obtaining $Q^\pi(s_t, a_t)$ through Monte Carlo sampling is often computationally prohibitive. Hence, we approximate $\hat{Q}(s_t, a_t)$ with the retrace algorithm [49], which can be written recursively as

$$\hat{Q}(s_t, a_t) \approx \hat{Q}^{\text{ret}}(s_t, a_t) = r_{t+1} + \gamma V^w(s_{t+1}) + \gamma \min\{1, \rho(s_t, a_t)\} [\hat{Q}^{\text{ret}}(s_{t+1}, a_{t+1}) - Q^w(s_{t+1}, a_{t+1})]. \quad (18)$$

The importance weight $\rho(s_t, a_t) = \pi^w(a_t|s_t)/\mu_t(a_t|s_t)$ is the ratio of probabilities of sampling the action a_t from state s_t with the current policy π^w and with the old policy μ_t .

The state value $V^w(s)$ and action value coefficient $l^w(s)$ are trained with the importance-sampled gradient of the $L2$ distance from \hat{Q}^{ret} :

$$g_Q = \sum_{t=1}^B \frac{\pi^w(a_t|s_t)}{\mu_t(a_t|s_t)} [\hat{Q}^{\text{ret}}(s_t, a_t) - Q^w(s_t, a_t)] \nabla_w Q^w(s_t, a_t). \quad (19)$$

In all cases, we use NN models with three layers of 128 units each. This size was chosen to disregard any issue with the representational capacity of the network. In fact, we found that even removing one layer and halving the number of units does not prevent the RL method from achieving similar

results to those shown in Sec. IV. Further implementation details of the algorithm can be found in Ref. [47].

B. Reward formulation

We wish to identify energy-optimal and time-optimal control policies by varying the aspect and density ratios that define the system of ODEs. Optimal policies minimize the control cost $\sum_{t=1}^T c_t$ with $c_t = \Delta t$ for time-optimal policies and $c_t = \int_{t-1}^t \tau(t)^2 dt = \tau_{t-1}^2 \Delta t$ for energy-optimal policies. The actuation cost τ^2 is used as a proxy for the energy cost for consistency with Ref. [27]. Due to the quadratic drag on the angular velocity [Eq. (3)], consistent with $\text{Re} \approx O(10^3)$, the average angular velocity scales as $w^2 \approx \tau$. The input power is $w\tau$ and should scale as $\tau^{3/2}$. Therefore, the quadratic actuation cost imposes on average a stricter penalty than the input power.

In the optimal control setting, boundary conditions, such as the initial and terminal positions of the glider, and constraints, such as bounds on the input power or torque, can be included directly in the problem formulation, as described for the present system in Ref. [27]. In RL boundary conditions can only be included in the functional form of the reward. The agent is discouraged from violating optimization constraints by introducing a condition for termination of a simulation, accompanied by negative terminal rewards. For example, here we inform the agent about the landing target by composing the reward as

$$r_t = -c_t + \|x_G - x_{t-1}\| - \|x_G - x_t\|. \quad (20)$$

Note that for a trajectory monotonically approaching x_G (which was always the case in Ref. [27]) the difference between the cumulative rewards computed by RL [Eq. (10) with $\gamma = 1$] and optimal control cost function is $\sum_{t=1}^T r_t + c_t = \|x_G - x_0\| - \|x_G - x_T\|$. If the exact target location x_G is reached at the terminal state, the discrepancy between the two formulations would be a constant baseline $\|x_G - x_0\|$, which can be proved to not affect the policy [50]. Therefore, a RL agent that maximizes cumulative rewards computed with Eq. (20) also minimizes either the time or the energy cost. We remark that τ , Δt and distances are consistent with the ODE model described in Sec. II and therefore all control costs are dimensionless.

The episodes are terminated if the ellipse touches the ground at $y_G = -50$. To allow the agent to explore diverse perching maneuvers, such as phugoid motions, the ground is recessed between $x = 50$ and $x = 100$ and is located at $y_G = -50 - 0.4 \min(x - 50, 100 - x)$. For both time optimal and energy optimal optimizations, the desired landing position and perching angle can be favored through a termination bonus:

$$r_T = -c_T + K[e^{-(x_G - x_T)^2} + e^{-10(\theta_G - \theta_T)^2}]. \quad (21)$$

The parameter K of the terminal reward is selected such that the termination bonus is of the same order of magnitude as the cumulative control cost. Because the gravity-driven descent requires $O(100)$ RL turns and the time cost is $\Delta t = 0.5$, we use $K = K_T = 50$ when training time-optimal policies. Similarly, due to the lower numerical values of the energy-cost, we use $K = K_E = 20$ when training for energy-optimal policies. The second exponential term of Eq. (21) is added only if $95 < x_T < 105$. This avoids the possibility of locally optimal policies where the agent lands with the correct angle θ_G , but far away from the target x_G with minimal time/energy costs.

IV. RESULTS

We explore the gliding strategies of the RL agents that aim to minimize either time-to-target or energy expenditure, by varying the aspect ratio β and density ratio ρ^* of the falling ellipse. These two optimization objectives may be seen as corresponding to the biologic scenarios of foraging and escaping from predators. Figure 2 shows the two prevailing flight patterns learned by the RL agent, which we refer to as “bounding” and “tumbling” flight [27]. The name “bounding” flight is

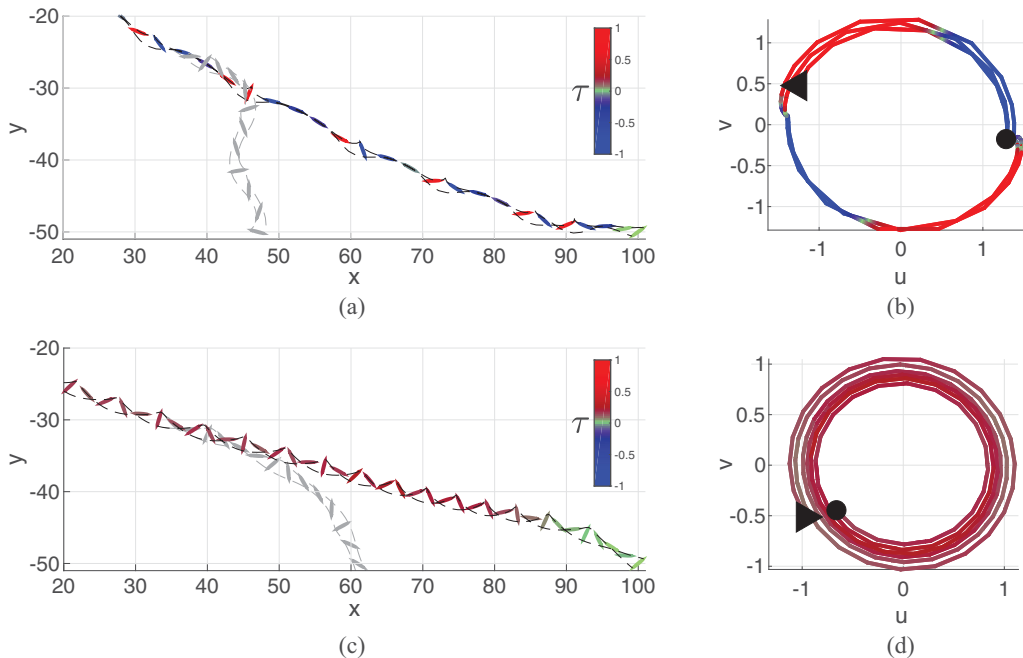


FIG. 2. Visualization of the two prevailing locomotion patterns adopted by RL agents for the active gliding model described in Sec. II. Trajectories on the x - y plane for (a) bounding ($\beta=0.1$, $\rho^*=100$) and (c) tumbling flight ($\beta=0.1$, $\rho^*=200$). The glider’s snapshots are colored to signal the value of the control torque, and the dashed black lines track the ellipse’s vertices. The grayed-out trajectories illustrate the glider’s passive descent when abruptly switching off active control. (b), (d) Corresponding trajectories on the u - v plane. For the sake of clarity, we omit the initial transient and final perching maneuverer. The trajectories are colored based on the control torque and their beginning and end are marked by a triangle and circle respectively.

due to an energy-saving flight strategy first analyzed by Refs. [51,52] with simplified models of intermittently flapping fliers.

In the present model, bounding flight is characterized by succeeding phases of gliding and tumbling. During gliding, the agent exerts negative torque to maintain a small angle of attack [represented by the blue snapshots of the glider in Fig. 2(a)], deflecting momentum from the air flow which slows down the descent. During the tumbling phase, the agent applies a rapid burst of positive torque [red snapshots of the glider in Fig. 2(a)] to generate lift and, after a rotation of 180° , recover into a gliding attitude.

The trajectory on the u - v plane [Fig. 2(b)] highlights that the sign of the control torque is correlated with whether u and v have the same sign. This behavior is consistent with the goal of maintaining upward lift. In fact, the vertical component of lift applied onto the falling ellipse is $\Gamma \dot{x}$, with $\dot{x} > 0$ because the target position is to the right of the starting position. From Eq. (9) of our ODE-based model, the lift is positive if u and v have opposite signs or if w is positive. Therefore, to create upward lift, the agent can either exert a positive τ to generate positive angular velocity, or, if u and v have opposite signs, exert a negative τ to reduce its angular velocity [Eq. (3)] and maintain the current orientation. The grayed-out trajectory shows what would happen during the gliding phase without active negative torque: the ellipse would increase its angle of attack, lose momentum and, eventually, fall vertically.

Tumbling flight, visualized in Figs. 2(c) and 2(d), is a much simpler pattern obtained by applying an almost constant torque that causes the ellipse to steadily rotate along its trajectory, thereby

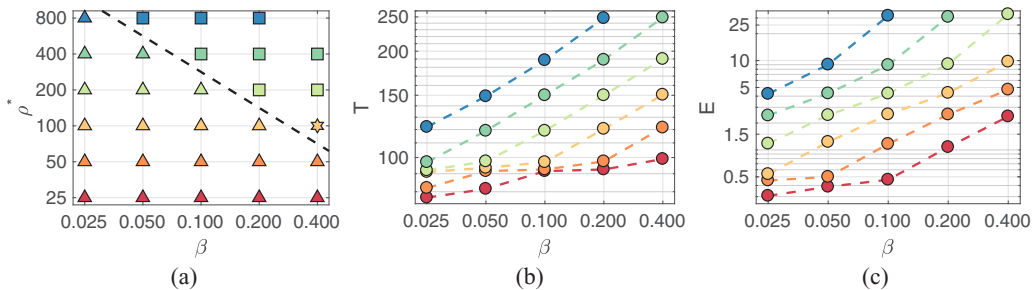


FIG. 3. Optimal solutions by sweeping the space of dimensionless parameters ρ^* and β of the ODE model outlined in Sec. II. (a) Flight pattern employed by time-optimal agents. Triangles refer to bounding flight and squares to tumbling. The policy for $\rho^*=100$ and $\beta=0.4$, marked by a star, alternated between the two patterns. The optimal (b) time-cost and (c) energy cost increase monotonically with both β and ρ^* . The symbols are colored depending on the value of ρ^* : red for $\rho^*=25$, orange $\rho^*=50$, yellow $\rho^*=100$, lime $\rho^*=200$, green $\rho^*=400$, blue $\rho^*=800$.

generating lift. The constant rotation is generally slowed down for the landing phase to descent and accurately perch at θ_G .

In Fig. 3 we report the effect of the ellipse’s shape and weight on the optimal strategies. The system of ODEs described in Sec. II is characterized by nondimensional parameters β and $I = \rho^*\beta$. Here we independently vary the density ratio ρ^* and the aspect ratio β in the range $[25, 800] \times [0.025, 0.4]$. For each set of dimensionless parameters we train a RL agent to find both the energy-optimal and time-optimal policies. The flight strategies employed by the RL agents can be clearly defined as either bounding flight or tumbling only in the time-optimal setting, while energy-optimal strategies tend to employ elements of both flight patterns. In Fig. 3(a), time-optimal policies that employ bounding flight are marked by a triangle, while those that use tumbling flight are marked by a square. We find that lighter and elongated bodies employ bounding flight while heavy and thick bodies employ tumbling flight. Only one policy, obtained for $\rho^* = 100$, $\beta = 0.4$ alternated between the two patterns and is marked by a star. These results indicate that a simple linear relation $\rho^*\beta = I \approx 30$ [outlined by a black dashed line in Fig. 3(a)] approximately describes the boundary between regions of the phase-space where one flight pattern is preferred over the other. In Figs. 3(b) and 3(c) we report the optimal time costs and optimal energy costs for all the combinations of nondimensional parameters. Fig. 3(c) shows that, the control torque magnitude (which we bound to $[-1, 1]$, Sec. III) required to reach x_T increases with I . In fact, we were unable to find time-optimal strategies for $I > 160$ and energy-optimal strategies for $I > 80$. While the choice of objective function should not affect the set of reachable landing locations, the energy cost discourages strong actuation strategies.

Once the RL training terminates, the agent obtains a set of opaque rules, parametrized by a neural network, to select actions. These rules are approximately optimal only for the states encountered during training, but can also be applied to new conditions. In fact, we find that the policies obtained through RL are remarkably robust. In Fig. 4(a) we apply the time-optimal policy for $\rho^* = 200$ and $\beta = 0.1$ to a new set of initial conditions along the x coordinate. Despite the agent never having encountered these position during training, it can always manage to reach the perching target. Similarly, in Fig. 4(b) we test the robustness with respect to changes to the parameters of the ODE model $\Psi = \{A, B, \mu, \nu, C_T, C_R\}$. At the beginning of a trajectory, we vary each parameter according to $\hat{\Psi}_i = \Psi_i \cdot \xi$ where ξ is sampled from a log-normal distribution with mean 1 and standard deviation σ_ξ . The color contour of Fig. 4(b) represent the envelopes of 10^4 trajectories for $\sigma_\xi = 0.1$ (blue), 0.2 (green), and 0.4 (orange). Surprisingly, even when the parameters are substantially different from those of the original model, the RL agent always finds its bearing and manages to land in the neighborhood of the target position. These results suggest that the policies

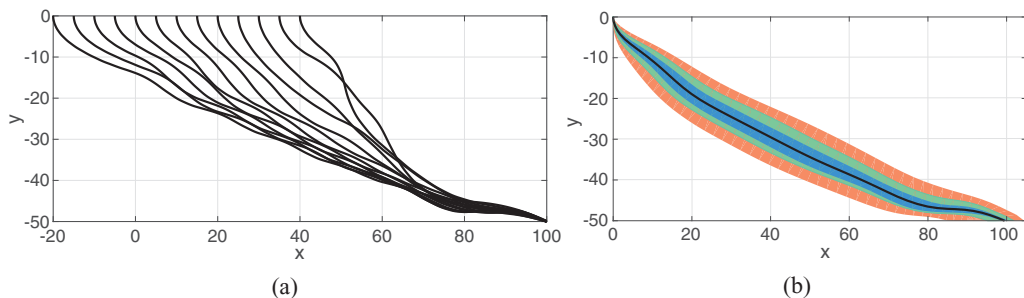


FIG. 4. Robustness of the trained RL agent. The agent is able to land in the neighborhood of the target x_G despite (a) starting its trajectory from initial conditions not seen during training or (b) applying proportional noise to the parameters of the model outlined in Sec. II. The color contours of panel (b) represent the envelopes of 10^4 trajectories for different values of standard deviation of the proportional log-normal noise. The blue contour corresponds to $\sigma_\xi = 0.1$, green to 0.2, orange to 0.4. These results are obtained with the time-optimal policy for $\beta=0.1$ and $\rho^*=200$.

learned through RL for the ODE system may be employed to land the glider at the target position within more accurate simulations of the surrounding flow or in experimental setups.

V. COMPARISON WITH OPTIMAL CONTROL

Having obtained approximately optimal policies with RL, we now compare them with the trajectories derived from optimal control (OC) by Ref. [27] for $\rho^* = 200$ and $\beta = 0.1$. In Fig. 5, we show the energy optimal trajectories, and in Fig. 6 we show the time optimal trajectories. In both cases, we find that the RL agent slightly surpasses the performance of the OC solution: the final energy-cost is approximately 2% lower for RL and the time-cost is 4% than that of OC. While in principle OC should find locally optimal trajectories, OC solvers (in this case GPOPS, see Ref. [27]) convert the problem into a set of finite-dimensional subproblems by discretizing the time. Therefore the (locally) optimal trajectory is found only up to a finite precision, in some cases allowing RL, which employs a different time-discretization, to achieve better performance. The finding that deep RL may surpass the performance of OC is consistent with the results obtained by [33] in the context of landing policies for unmanned aerial vehicles.

The RL and OC solutions qualitatively find the same control strategy. The energy-optimal trajectories consist in finding a constant minimal torque that generates enough lift to reach x_G by steady tumbling flight. The time-optimal controller follows a “bang-bang” pattern that alternately

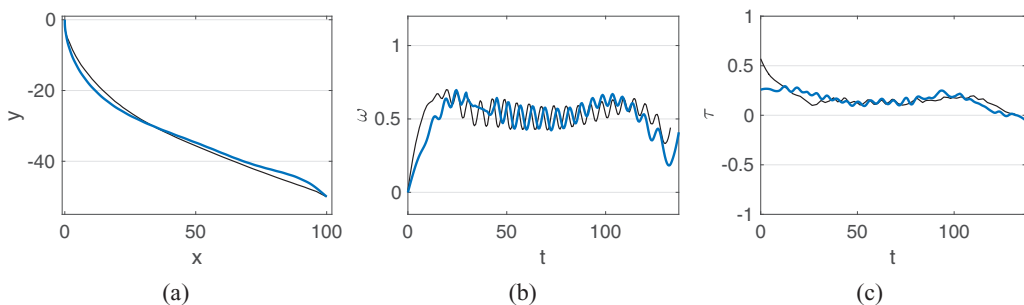


FIG. 5. Energy-optimal (a) x - y trajectory, (b) angular velocity, and (c) control torque of the present gliding model obtained by reinforcement learning (blue lines) and optimal control by Ref. [27] (black lines) for $\beta = 0.1$ and $\rho^* = 200$.

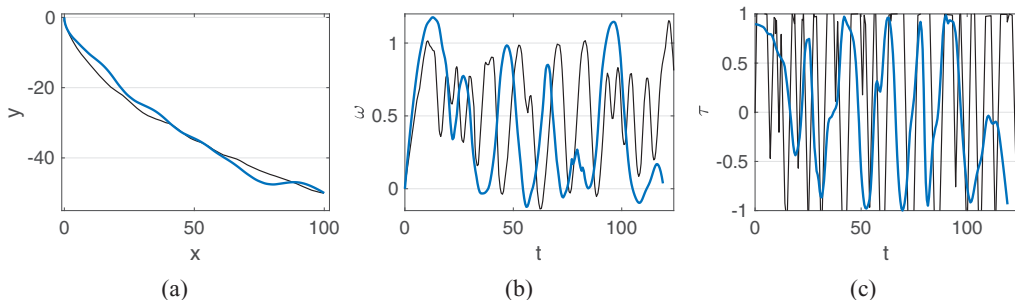


FIG. 6. Time-optimal (a) x - y trajectory, (b) angular velocity, and (c) control torque of the present gliding model obtained by reinforcement learning (blue lines) and optimal control by Ref. [27] (black lines) for $\beta = 0.1$ and $\rho^* = 200$.

reaches the two bounds of the action space as the glider switches between gliding and tumbling flight. However, the main drawback of RL is having only the reward signal to nudge the system towards satisfying the constraints. We can impose arbitrary initial conditions and bounds to the action space (Sec. III), but we cannot directly control the terminal state of the glider. Only through expert shaping of the reward function, as outlined in Sec. III B, can we train policies that reliably land at x_G with perching angle θ_G . The precision of the learned control policies may be evaluated by observing the distribution of outcomes after changing the initial conditions. In Fig. 7 we perturb the initial angle θ_0 . We find that the deviation from x_G and θ_G are both of the order $O(10^{-2})$ and that the improvement of RL over OC is statistically consistent for all initial conditions.

One of the advantages of RL relative to optimal control, beside not requiring a precise model of the environment, is that RL learns closed-loop control strategies. While OC has to compute *de novo* an open-loop policy after any perturbation that drives the system away from the planned path, the RL agent selects action contextually and robustly based on the current state. This suggests that RL policies from simplified models can be transferred to related more accurate simulations [45] or robotic experiments (for example, see Ref. [53]).

VI. ANALYSIS OF THE LEARNING METHODS

The RL agent starts the training by performing actions haphazardly, due to the control policy which is initialized with random small weights being weakly affected by the state in which the agent finds itself. Since the desired landing location is encoded in the reward, the agent’s trajectories gradually shift towards landing closer to x_G . To have a fair comparison with the trajectories obtained through optimal control, the RL agents should be able to precisely and reliably land at the prescribed

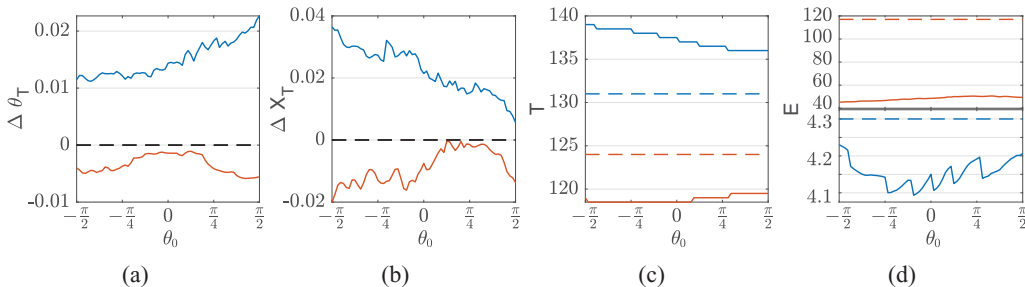


FIG. 7. (a) Landing angle, (b) position, (c) time cost, and (d) energy cost of the energy-optimal (blue lines, corresponding to Fig. 5) and time-optimal (orange lines, corresponding to Fig. 6) policies after varying the initial angle θ_0 . The dashed lines correspond to the trajectory found by optimal control for $\theta_0 = 0$ by Ref. [27].

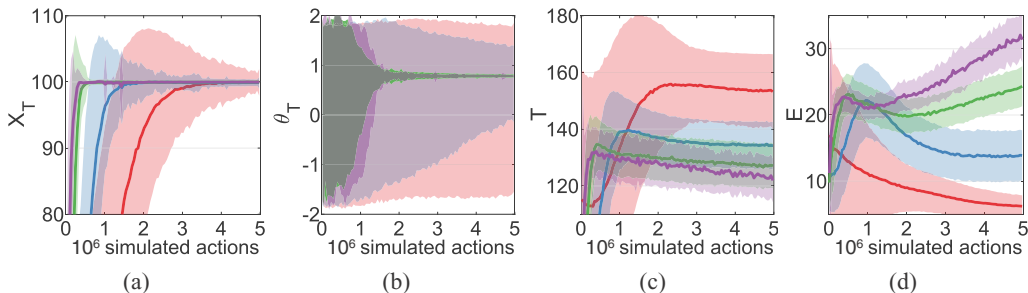


FIG. 8. Distribution (mean and contours of one standard deviation) of (a) landing position, (b) angle, (c) time-cost, and (d) energy-cost during training of the time-optimal policy with varying the actuation frequency. Time between actions $\Delta t = 0.5$ (purple), $\Delta t = 2$ (green), $\Delta t = 16$ (blue), $\Delta t = 64$ (red).

target position. In general, the behaviors learned through RL are qualitatively convincing, however, depending on the problem, it can be hard to obtain quantitatively precise control policies.

In this section we explore how the definition of the RL problem, as well as the choice of learning algorithm, affects the precision of the resulting policy. In Fig. 8 we show the effect of the time-discretization Δt of the RL decision process. For example, for $\Delta t = 64$ the agent performs only two or three actions per simulation. In this case, the actions need to be chosen precisely because the agent has few opportunities to correct its descent. For this reason, it is more challenging for the RL algorithm to find these optimal actions, which can be observed by the increased training time required for the agent to reliably land around x_G . Greater actuation frequency allows higher time-resolution of the optimal policy and therefore, in this case, lower optimal time-cost. We note that we perform one training step (gradient update) per agent’s action. Therefore, training with $\Delta t = 0.5$ is faster than training with $\Delta t = 64$ because simulating each action requires 128 times fewer simulated time steps.

In Sec. III A we defined an agent being able to observe all the state variables of the ODE ($s := \{x, y, \theta, u, v, w\}$), satisfying the assumption of Markovian dynamics required by RL. This assumption is often voided in practical application of RL where only partial information (e.g., sensor measurements) are available to the agent (examples with fluid dynamics include [19,29,31,45]). Moreover, a smaller state representation has the potential benefit of the policy/value NN having fewer parameters which can be trained more effectively by avoiding the curse of dimensionality. We explore this idea with Fig. 9, where we measure the training outcomes when hiding the velocities $\{u, v, w\}$ from the agent. Hiding these quantities may be biologically plausible because velocities are more difficult to accurately measure from visual information. Figure 9 indicates that despite the

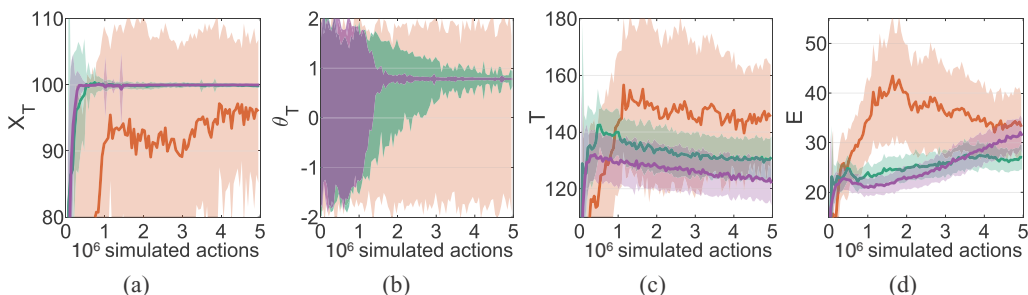


FIG. 9. Distribution (mean and contours of one standard deviation) of (a) landing position, (b) angle, (c) time-cost, and (d) energy-cost during training of the time-optimal policy for the fully observable problem defined in Sec. III (purple), by an agent that observes its state as $\{x, y, \theta\}$ (orange), and by an agent which observes $\{x, y, \theta\}$ and models its policy and value function with a RNN (green).

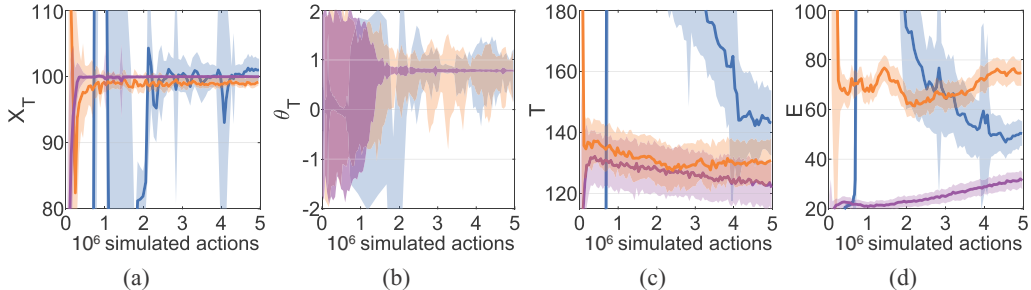


FIG. 10. Distribution (mean and contours of one standard deviation) of (a) landing position, (b) angle, (c) time-cost and (d) energy-cost during training of the time-optimal policy with three state-of-the-art RL algorithms: RACER (purple, Sec. III A), normalized advantage functions [56] (blue), and proximal policy optimization [57] (orange). RACER reliably learns to land at the target position by carefully managing the pace of the policy update.

smaller parametrization of the NN, it is more difficult for an agent to find accurate landing policies from partial information. A common work-around employed in the machine-learning community is to swap the conventional feedforward NN with recurrent NN (RNN), specifically here we use the LSTM model [54]. RNNs learn to extrapolate hidden state variables from the time-history of states. Here it is straightforward to see how velocities may be approximated from a time-history of positions. However, RNNs are notoriously harder to train [55]. The RNN agent does learn to accurately land at x_T , but it requires more iterations to find the correct perching angle θ_T or reach the time-cost of an agent with full observability.

Most RL algorithms are designed to tackle video-games or simulated robotics benchmarks. These problems challenge RL methods with sparse rewards and hard exploration. Conversely, RACER was designed for problems modeled by PDEs or ODEs with the objective of obtaining precise optimal control policies with the potential downside of reduced exploration. In fact, RACER relies on the agent having an informative reward function (as opposed to a sparse signal). In Fig. 10 we show the learning progress for three state-of-the-art RL algorithms. RACER manages to reliably land in the proximity of x_G , after the first 1000 observed trajectories. The precision of the distribution of landing locations, obtained here by sampling the stochastic policy during training, can be increased when evaluating a trained policy by choosing deterministically at every turn the action corresponding to its mean $m(s)$ (as in Fig. 7). Normalized advantage function (NAF) [56] is an off-policy value-iteration algorithm which learns a quadratic parametrization of the action value $Q^\theta(s, a)$, similar to the one defined in Eq. (16). One of the main differences with respect to RACER is that the mean $m^\theta(s)$ of the policy is not trained with the policy gradient [Eq. (17)] but with the critic gradient [Eq. (19)]. While the accuracy of the parametrized $Q^\theta(s, a)$ might increase during training, $m^\theta(s)$ does not necessarily correspond to better action, leading to the erratic distribution of landing positions in Fig. 10. Proximal policy optimization (PPO) [57] is an on-policy actor-critic algorithm. This algorithm's main difference with respect to RACER is that only the most recent (on-policy) trajectories are used to update the policy. This allows estimating $Q^\pi(s, a)$ directly from on-policy rewards [58] rather than with an off-policy estimator (here we used retrace 18), and it bypasses the need for learning a parametric $Q^\theta(s, a)$. While PPO has led to many state-of-the-art results in benchmark test cases, here it does not succeed to center the distribution of landing positions around x_G . This could be attributed to the high variance of the on-policy estimator for $Q^\pi(s, a)$.

VII. CONCLUSIONS

We have demonstrated that reinforcement learning can be used to develop gliding agents that execute complex and precise control patterns using a simple model of the controlled gravity-driven descent of an elliptical object. We show that RL agents learn a variety of optimal flight patterns and

perching maneuvers that minimize either time-to-target or energy cost. The RL agents were able to match and even surpass the performance of trajectories found through Optimal Control. We also show that the RL agents can generalize their behavior, allowing them to select adequate actions even after perturbing the system. Finally, we examined the effects of the ellipse's density and aspect ratio to find that the optimal policies lead to either bounding flight or tumbling flight. Bounding flight is characterized as alternating phases of gliding with a small angle of attack and rapid rotation to generate lift. Tumbling flight is characterized by continual rotation, propelled by a minimal almost constant torque. Ongoing work aims to extend the present algorithms to three dimensional Direct Numerical Simulations of gliders and using lessons learned from these simulations for the perching and gliding of UAVs.

ACKNOWLEDGMENTS

We thank Siddhartha Verma for helpful discussions and feedback on this manuscript. This work was supported by European Research Council Advanced Investigator Award No. 341117. Computational resources were provided by Swiss National Supercomputing Centre (CSCS) Project No. s658.

-
- [1] C. D. Cone, Thermal soaring of birds, *Am. Sci.* **50**, 180 (1962).
 - [2] A. Azuma and Y. Okuno, Flight of a samara, *Alsomitra macrocarpa*, *J. Theor. Biol.* **129**, 263 (1987).
 - [3] R. Dudley, G. Byrnes, S. Yanoviak, B. Borrell, R. Brown, and J. A. McGuire, Gliding and the Functional Origins of Flight: Biomechanical Novelty or Necessity? *Annu. Rev. Ecol. Evol. Syst.* **38**, 179 (2007).
 - [4] S. M. Jackson, Glide angle in the genus *petaurus* and a review of gliding in mammals, *Mammal Rev.* **30**, 9 (2000).
 - [5] A. Mori, and T. Hikida, Field observations on the social behavior of the flying lizard, *Draco volans sumatranus*, in Borneo, in *Copeia* (American Society of Ichthyologists and Herpetologists, Lawrence, KS, 1994), pp. 124–130.
 - [6] M. G. McCay, Aerodynamic stability and maneuverability of the gliding frog *Polypedates dennysi*, *J. Expt. Biol.* **204**, 2817 (2001).
 - [7] J. J. Socha, Kinematics: Gliding flight in the paradise tree snake, *Nature* **418**, 603 (2002).
 - [8] S. P. Yanoviak, R. Dudley, and M. Kaspari, Directed aerial descent in canopy ants, *Nature* **433**, 624 (2005).
 - [9] S. P. Yanoviak, M. Kaspari, and R. Dudley, Gliding hexapods and the origins of insect aerial behaviour, *Biol. Lett.* **5**, 510 (2009).
 - [10] S. P. Yanoviak, and R. Dudley, The role of visual cues in directed aerial descent of *Cephalotes atratus* workers (hymenoptera: Formicidae), *J. Exp. Biol.* **209**, 1777 (2006).
 - [11] J. M. V. Rayner, Bounding and undulating flight in birds, *J. Theor. Biol.* **117**, 47 (1985).
 - [12] P. Paoletti and L. Mahadevan, Intermittent locomotion as an optimal control strategy, *Proc. Roy. Soc. A: Math. Phys. Engg. Sci.* **470**, 20130535 (2014).
 - [13] D. Gurdan, J. Stumpf, M. Achtelik, K. M. Doth, G. Hirzinger, and D. Rus, Energy-efficient autonomous four-rotor flying robot controlled at 1 khz, in *Proceedings of the IEEE International Conference on Robotics and Automation* (IEEE, Roma, Italy, 2007), pp. 361–366.
 - [14] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, A simple learning strategy for high-speed quadcopter multi-flips, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'10)* (IEEE, Anchorage, Alaska, 2010), pp. 1642–1648.
 - [15] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, Cooperative grasping and transport using multiple quadrotors, in *Distributed Autonomous Robotic Systems* (Springer, Berlin, 2013), pp. 545–558.
 - [16] M. Müller, S. Lupashin, and R. D'Andrea, Quadcopter ball juggling, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)* (IEEE, San Francisco, California, 2011), pp. 5113–5120.

- [17] J. Thomas, M. Pope, G. Loianno, E. W. Hawkes, M. A. Estrada, H. Jiang, M. R. Cutkosky, and V. Kumar, Aggressive flight with quadrotors for perching on inclined surfaces, *J. Mech. Robot.* **8**, 051007 (2016).
- [18] M. F. Bin Abas, A. S. B. M. Rafie, H. B. Yusoff, and K. A. B. Ahmad, Flapping wing micro-aerial-vehicle: Kinematics, membranes, and flapping mechanisms of ornithopter and insect flight, *Chin. J. Aeronaut.* **29**, 1159 (2016).
- [19] G. Reddy, A. Celani, T. J. Sejnowski, and M. Vergassola, Learning to soar in turbulent environments, *Proc. Natl. Acad. Sci. USA* **113**, E4877 (2016).
- [20] G. Reddy, J. Wong-Ng, A. Celani, T. J. Sejnowski, and M. Vergassola, Glider soaring via reinforcement learning in the field, *Nature* **562**, 236 (2018).
- [21] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic Programming and Optimal Control* (Athena Scientific, Belmont, MA, 1995), Vol. 1.
- [22] L. P. Kaelbling, M. L. Littman, and A. W. Moore, Reinforcement learning: A survey, *J. Artif. Intell. Res* **4**, 237 (1996).
- [23] R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An introduction* (MIT Press, Cambridge, MA, 1998), Vol. 1.
- [24] A. Andersen, U. Pesavento, and Z. J. Wang, Analysis of transitions between fluttering, tumbling and steady descent of falling cards, *J. Fluid Mech.* **541**, 91 (2005).
- [25] A. Andersen, U. Pesavento, and Z. J. Wang, Unsteady aerodynamics of fluttering and tumbling plates, *J. Fluid Mech.* **541**, 65 (2005).
- [26] Z. J. Wang, J. M. Birch, and M. H. Dickinson, Unsteady forces and flows in low Reynolds number hovering flight: Two-dimensional computations vs. robotic wing experiments, *J. Exp. Biol.* **207**, 449 (2004).
- [27] P. Paoletti, and L. Mahadevan, Planar controlled gliding, tumbling, and descent, *J. Fluid Mech.* **689**, 489 (2011).
- [28] S. Colabrese, K. Gustavsson, A. Celani, and L. Biferale, Flow Navigation by Smart Microswimmers Via Reinforcement Learning, *Phys. Rev. Lett.* **118**, 158004 (2017).
- [29] M. Gazzola, B. Hejazialhosseini, and P. Koumoutsakos, Reinforcement learning and wavelet adapted vortex methods for simulations of self-propelled swimmers, *SIAM J. Sci. Comput.* **36**, B622 (2014).
- [30] M. Gazzola, A. A. Tchieu, D. Alexeev, A. de Brauer, and P. Koumoutsakos, Learning to school in the presence of hydrodynamic interactions, *J. Fluid Mech.* **789**, 726 (2016).
- [31] G. Novati, S. Verma, D. Alexeev, D. Rossinelli, W. M. van Rees, and P. Koumoutsakos, Synchronisation through learning for two self-propelled swimmers, *Bioinspir. Biomimet.* **12**, 036001 (2017).
- [32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fiedelnd, G. Ostrovski *et al.*, Human-level control through deep-reinforcement-learning, *Nature* **518**, 529 (2015).
- [33] A. Waldock, C. Greatwood, F. Salama, and T. Richardson, Learning to perform a perched landing on the ground using deep reinforcement learning, *J. Intell. Robot. Syst.* **92**, 685 (2018).
- [34] K. Hang, X. Lyu, H. Song, J. A. Stork, A. M. Dollar, D. Kragic, and F. Zhang, Perching and resting—A paradigm for UAV maneuvering with modularized landing gears, *Sci. Robot.* **4**, eaau6637 (2019).
- [35] A. Belmonte, H. Eisenberg, and E. Moses, From Flutter to Tumble: Inertial Drag and Froude Similarity in Falling Paper, *Phys. Rev. Lett.* **81**, 345 (1998).
- [36] L. Mahadevan, W. S. Ryu, and A. D. T. Samuel, Tumbling cards, *Phys. Fluids* **11**, 1 (1999).
- [37] R. Mittal, V. Seshadri, and H. S. Udaykumar, Flutter, tumble and vortex induced autorotation, *Theor. Comput. Fluid Dyn.* **17**, 165 (2004).
- [38] U. Pesavento, and Z. J. Wang, Falling Paper: Navier-Stokes Solutions, Model of Fluid Forces, and Center of Mass Elevation, *Phys. Rev. Lett.* **93**, 144501 (2004).
- [39] H. Lamb, *Hydrodynamics* (Cambridge University Press, Cambridge, 1932).
- [40] S. P. Yanoviak, Y. Munk, M. Kaspari, and R. Dudley, Aerial manoeuvrability in wingless gliding ants (*Cephalotes atratus*), *Proc. Roy. Soc. London B: Biol. Sci.* **277**, 2199 (2010).
- [41] S. Levine, C. Finn, T. Darrell, and P. Abbeel, End-to-end training of deep visuomotor policies, *J. Mach. Learn. Res.* **17**, 1334 (2016).

- [42] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, Mastering the game of go with deep neural networks and tree search, *Nature* **529**, 484 (2016).
- [43] R. Bellman, On the theory of dynamic programming, *Proc. Natl. Acad. Sci. USA* **38**, 716 (1952).
- [44] T. Duriez, S. L. Brunton, and B. R. Noack, *Machine Learning Control-Taming Nonlinear Dynamics and Turbulence* (Springer, Berlin, 2017).
- [45] S. Verma, G. Novati, and P. Koumoutsakos, Efficient collective swimming by harnessing vortices through deep reinforcement learning, *Proc. Natl. Acad. Sci. USA* **115**, 5849 (2018).
- [46] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2000), pp. 1057–1063.
- [47] G. Novati, and P. Koumoutsakos, Remember and forget for experience replay, in *Proceedings of the 36th International Conference on Machine Learning* Vol. 97 (PMLR, Long Beach, California, 2019), pp. 4851–4860.
- [48] T. Degris, M. White, and R. S. Sutton, Off-policy actor-critic, in *Proceedings of the 29th International Conference on Machine Learning* (Omnipress, Edinburgh, Scotland, 2012), pp. 179–186.
- [49] R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare, Safe and efficient off-policy reinforcement learning, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2016), pp. 1054–1062.
- [50] A. Y. Ng, D. Harada, and S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in *Proceedings of the 16th International Conference on Machine Learning (ICML'99)* (Morgan Kaufmann Publishers Inc., San Francisco, California, 1999), Vol. 99, pp. 278–287.
- [51] M. J. Lighthill, Introduction to the scaling of aerial locomotion, *Scale effects in animal locomotion* (Cambridge University Press, Cambridge, UK, 1977), pp. 365–404.
- [52] J. M. V. Rayner, The intermittent flight of birds, *Scale effects in animal locomotion* (Academic Press, New York, 1977), pp. 437–443.
- [53] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, S. V. Sun, P. Abbeel, and S. Levine, Deep reinforcement learning for tensegrity robot locomotion, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, Singapore, 2017), pp. 634–641.
- [54] F. A. Gers, J. Schmidhuber, and F. Cummins, Learning to forget: Continual prediction with LSTM, *Neural Comput.* **12**, 2451 (2000).
- [55] I. Sutskever, *Training Recurrent Neural Networks* (University of Toronto, Toronto, Ontario, Canada, 2013).
- [56] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, Continuous deep q-learning with model-based acceleration, in *Proceedings of the 33rd International Conference on Machine Learning*, Vol. 48 (PMLR, New York, 2016), Vol. 48, pp. 2829–2838.
- [57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, [arXiv:1707.06347](https://arxiv.org/abs/1707.06347).
- [58] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, High-dimensional continuous control using generalized advantage estimation, *4th International Conference on Learning Representations, (ICLR) (2016)*, [arXiv:1506.02438](https://arxiv.org/abs/1506.02438).