Learning Stabilization Control from Observations by Learning Lyapunov-like Proxy Models

Milan Ganai, Chiaki Hirayama, Ya-Chien Chang, and Sicun Gao

***Abstract*— The deployment of Reinforcement Learning to robotics applications faces the difficulty of reward engineering. Therefore, approaches have focused on creating reward functions by Learning from Observations (LfO) which is the task of learning policies from expert trajectories that only contain state sequences. We propose new methods for LfO for the important class of continuous control problems of learning to stabilize, by introducing intermediate proxy models acting as reward functions between the expert and the agent policy based on Lyapunov stability theory. Our LfO training process consists of two steps. The first step attempts to learn a Lyapunov-like landscape proxy model from expert state sequences without access to any kinematics model, and the second step uses the learned landscape model to guide in training the learner's policy. We formulate novel learning objectives for the two steps that are important for overall training success. We evaluate our methods in real automobile robot environments and other simulated stabilization control problems in model-free settings, like Quadrotor control and maintaining upright positions of Hopper in MuJoCo. We compare with state-of-the-art approaches and show the proposed methods can learn efficiently with less expert observations.**

I. Introduction

Reward engineering remains a challenge in applying Reinforcement Learning to robotic control problems [1]. Manual design of suitable reward functions can be complicated, requiring extra sensors [2], [3] and may guide learner to acquire unintended behavior [4]. Consequently, imitation learning has become an indispensable approach that allows usage of expert demonstrations to replace reward functions, typically in the setting of learning from demonstrations (LfD) [5], [6], where the learner has access to several state-action pair sequences of desired behavior generated by the expert. However, actions may be difficult to retrieve (like with human experts) and there can be mismatch between expert and learner action spaces. Therefore, methods have been proposed for the problem of learning from observations (LfO) where access to expert behavior is limited to state sequences [7]. LfO can model many challenging forms of imitation such as learning from videos to acquire control skills. State-of-the-art methods for LfO typically adapt methods from LfD to the state observation setting. For instance, the Generative Adversarial Imitation from Observation algorithm (GAIfO) [7], [8] adapts Generative Adversarial Imitation Learning (GAIL) [9] by using Generative Adversarial Networks (GANs) [10] to learn policies that mimic expert state transition distributions. However, because of the absence of actions, models, and reward signals, LfO problems are often too challenging in the general continuous control settings [8]. Our goal is to develop efficient LfO methods for a specific but important class of continuous control problems: stabilization control, such as controlling a robot to maintain an upright standing pose, or controlling an autonomous car to track a given path. Stabilization is the basis of all advanced control problems [11]. Complex tasks can almost always be decomposed into a motion planning part and a stabilization control part, which is used for following the plans.

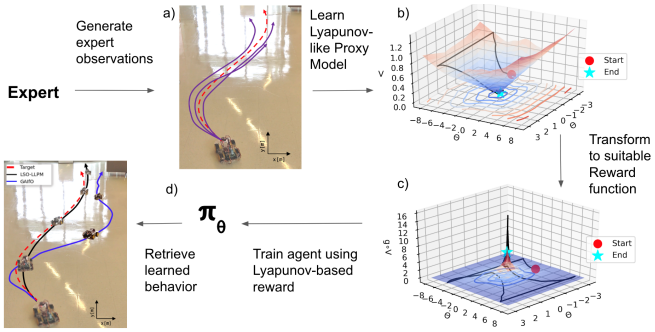

Fig. 1: (a) We collect some expert state trajectories. (b) We learn a Lyapunov-like proxy model from the state trajectories. (c) We transform the model to a reward function. (d) We subsequently train the agent to learn a policy using the reward function. In the bottom left image, we show a trajectory from proposed approach compared with that of state-of-the-art GAIfO [8].

We propose a novel two-step LfO procedure for stabilization control to achieve efficient imitation learning, supported by control-theoretic principles. We exploit the framework of Lyapunov stability theory, which provides a general structure for stabilization in nonlinear control systems [11]. The core idea of Lyapunov methods is to construct an energy-like landscape that provides sufficient conditions for control systems to converge to and stabilize at their equilibrium points. In the LfO setting, we propose methods that learn Lyapunov-like functions as an intermediate goal, which we name as *Lyapunov-like proxy models*. From observations of expert state sequences (Figure 1 (a)), we first learn a neural network model that attempts to construct an approximate Lyapunov landscape to explain convergence of expert states (Figure 1 (b)). We transform this learned landscape model (Figure 1 (c)) to guide the training of the learner policy (Figure 1 (d)). For this procedure to succeed, it is critical for the proxy model to capture the geometry of the region of attraction and

This material is based on work supported by DARPA Contract No. FA8750-18-C-0092, AFOSR YIP FA9550-19-1-0041, NSF Career CCF 2047034, NSF CCF DASS 2217723, and Amazon Research Award.

M. Ganai, C. Hirayama, Y.-C. Chang, and S. Gao are with the Department of Computer Science and Engineering, UC San Diego, USA. (Email: mganai@ucsd.edu; chirayam@eng.ucsd.edu; yac021@eng.ucsd.edu; and sicung@eng.ucsd.edu).

convergence rates (i.e. Lie derivatives of Lyapunov function) so they are consistent with expert behavior dynamics. We test our proposed procedure in real and simulated environments.

In the context of GAN-based approaches for LfO and LfD [9], [12]–[15], we can consider the Lyapunov-like proxy models as introducing a special hypothesis class for the discriminators, as opposed to using generic distributions, to achieve more robust modeling of the expert behaviors for the specific context of stabilization problems.

We describe our technical approach in Section IV. In Section V, we evaluate our approach in challenging environments for learning to stabilize from observations, including Acrobot, Quadrotor control, Automobile path-tracking, and stabilization version of MuJoCo Hopper robot. We evaluate our approach on real automobile robot environments in Section VI. Compared to state-of-the-art methods, the proposed methods can learn more efficiently from less observations of expert trajectories and produce more stable control policies.

II. Related Work

Imitation Learning. Learning from Demonstrations (LfD) problems have been the most well-studied form of imitation learning in MDPs. The learner has access to state-action trajectories of the expert without knowledge of the transition dynamics or the reward function in the MDP. Existing approaches in LfD generally fall into three categories: Behavioral Cloning [16], [17], Inverse Reinforcement Learning [18], [19], and Adversarial Imitation Learning [9], [12]–[15]. Our approaches are most related to the last category, where imitation learning is formulated in an adversarial framework of learning the policy as a generative model from the simultaneous training of a generator model and a discriminator model. The state-of-the-art algorithms are typically based on the method of Generative Adversarial Imitation Learning (GAIL) [9], which uses Generative Adversarial Networks (GANs) [10] to train a generative model that can create trajectories with a state-action occupancy measure similar to that of the expert, while the discriminator learns to provide feedback signals by differentiating the behavior distributions between the expert and the learner. Discriminator Actor-Critic (DAC) [20] is an off-policy version of GAIL that uses state-transition samples to train off-policy to achieve mode-covering in distribution matching. In general, LfD does not handle the most challenging forms of imitation, like learning from visual data of only expert state observations.

Learning from Observations (LfO). GAN-based approaches from LfD can be extended to the LfO setting since such methods can be used to only imitate state distributions without access to actions from the expert. The Generative Adversarial Imitation from Observation (GAIfO) algorithm [7], [8] uses a similar GAN framework as GAIL. Instead of training the discriminator using state-action pairs, GAIfO uses state transitions so that the imitator policy, which is the generator, produces a similar distribution of state transitions as expert. Another approach to LfO is Behavior Cloning from Observation (BCO) [21] where the agent acquires inverse dynamics experience in a self-supervised manner, which is then used to create a model to perform a task based on expert state observations. This approach does not require post-demonstration environment interactions, so it reduces the delay before the imitating agent is successful and avoids training and learning in risky environments (like autonomous vehicles). BCO nevertheless faces the issue of inaccuracies in that the learned inverse dynamics model accumulates error over time [22]–[24]. GAIfO has been shown to perform better than BCO by alleviating this compounding error issue [7], [8]. Off-policy learning methods have also been introduced to the GAN-based approaches for LfO. The works of Off-Policy Imitation Learning from Observations (OPOLO) [25] and Inverse Dynamics Disagreement Minimization (IDDM) [26] are able to accelerate the training of the learner/generator in the GAN framework by leveraging off-policy training of the inverse dynamics or action models of the environment before imitating from the expert. The works of Forward Adversarial Imitation Learning (FAIL) [27] and Imitating Latent Policies from Observation (ILPO) [28] also learn forward dynamics models and assume environment dynamics is deterministic with discrete action spaces. In contrast to this line of work, we focus on efficient learning in the setting of on-policy LfO without learning models or leveraging off-policy samples. The performance gain from off-policy methods can be used orthogonal to ours.

Lyapunov-based Methods in Reinforcement Learning and Imitation Learning. Lyapunov-based approaches have been recently introduced in model-free learning tasks [29]–[36]. [29], [30] solves constrained MDPs with Lyapunov methods to constrain a policy search space during each training iteration. They formulate a constraint value function as a Lyapunov function and update the policy with Lyapunov constraints. The work of [31] constructs candidate Lyapunov functions from value functions in an actor-critic framework, using Lyapunov decreasing condition as critic value to enhance stability properties of neural control policies. The work of [33] performs self-learning of almost-Lyapunov functions, used as a critic function to accelerate policy training.

A Lyapunov-based approach for LfD problems by [34] relies on Lyapunov theory and local quadratic constraints to establish safety and stability guarantees for deep neural network control systems. The methods assume that the environment is a linear dynamical system and do not consider more complex environments. Other Lyapunov-based methods in [36]–[38] learn globally asymptotic system dynamics (transition model) and then plan for trajectories toward the goal state by leveraging the prediction models, with error compounding issues similar to BCO [21]–[24].

III. Preliminaries

Markov Decision Processes and Learning from Observations. We consider imitation learning in Markov Decision Processes (MDPs) [39], [40]. MDPs are defined as $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the action space. $P(s_{t+1}|s_t, a_t)$ is the transition probability of reaching state s_{t+1} after action a_t is taken at state s_t, where t denotes a time-step but does not directly affect

the transition probability. In the standard RL setting, the agent receives $r(s, a, s')$ where $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and γ is the discount factor. In the imitation learning setting, the reward function is assumed unknown, and the goal is to train the agent to perform the task given expert observation trajectories, so we can write the MDP for imitation problems as $\langle \mathcal{S}, \mathcal{A}, P, C \rangle$ where C is the cost function that measures the deviation. In the LfO setting, there is no action information in the expert trajectories, so the cost function for the learner is defined as $C : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ which only assigns cost after comparing states between the learner and the expert. The LfO agent attempts to learn a policy $\pi_\phi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ such that sampling from policy π_ϕ produces a distribution of state-action-cost tuples $\{(s_i, a_i, c_i)\}$ under environment dynamics. The goal of LfO is to minimize cumulative of the cost function C along trajectories generated by the agent's policy π_ϕ.

Stability and Lyapunov Methods. In stabilization problems, the agents control dynamical systems:

$$\dot{x}(t) = f(x(t), u(t)), u(t) = h(x(t)), x(0) = x_0, \quad (1)$$

where $x(t)$ takes values in an n-dimensional state space $X \subseteq \mathbb{R}^n$, $f : X \to \mathbb{R}^n$ is a Lipschitz-continuous vector field, $h : X \to \mathbb{R}^m$ is a control function. Each $x(t) \in X$ is called a state vector and $u(t) \in \mathbb{R}^m$ is a control vector. The notion of stability is then formally defined as:

Definition 1 (Lyapunov and Asymptotic Stability)**.** A system of Eq. 1 is *Lyapunov stable* at the origin $x = 0$, if for all $\epsilon > 0$, there exists $\delta = \delta(\epsilon) > 0$ such that for all $\|x(0)\| < \delta$, then $\|x(t)\| < \epsilon$ for all $t \geq 0$. The system is *locally asymptotically stable* at the origin if it is Lyapunov stable and there exists $\delta > 0$ such that if $\|x(0)\| < \delta$, then $\lim_{t\to\infty} x(t) = 0$. $\|\cdot\|$ is typically the Euclidean norm.

Definition 2 (Lie Derivatives)**.** Consider the system in Eq. 1 and let $V : X \to \mathbb{R}$ be a continuously differentiable function. The *Lie derivative* of V over f is defined as

$$L_f V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \dot{x}_i(t). \quad (2)$$

The Lie derivative measures the change of V over time along the direction of the system dynamics.

Definition 3 (Lyapunov Conditions for Asymptotic Stability)**.** Consider a controlled system Eq. 1 with an equilibrium at the origin, i.e. $\exists u \in \mathbb{R}^m$ so $f(0, u) = 0$. Suppose there is a continuously differentiable function $V : X \to \mathbb{R}$ satisfying $V(0) = 0$; $\forall x \in X \setminus \{0\}$, $V(x) > 0$; and $L_f V(x) < 0$. Then V is a *Lyapunov function*. The system f is asymptotically stable at the origin if Lyapunov function V can be found.

We train neural networks to learn approximate Lyapunov landscapes based on the expert state observations by enforcing Lyapunov conditions to be satisfied along expert trajectories. Because the procedure is learning-based and cannot guarantee the Lyapunov conditions throughout the entire state space, we use the name *Lyapunov-like proxy model* for our LfO setting.

GAN-based Approaches in Imitation Learning. Adversarial Imitation Learning approaches [9], [12]–[15] rely largely on the usage of Generative Adversarial Networks (GANs) [10]. The GAN architecture pits two neural networks against each other in order to make one neural network produce data distributions similar to that of the training data. The two neural networks are called the Generator (G) and Discriminator (D) respectively. G attempts to fool D by making its output distribution p_g similar to training data distribution p_{data} given data from a prior on input noise variables p_z. D is trained to maximize the probability of assigning the correct label to both training data examples and samples from G, while G is concurrently trained to minimize $\log(1 - D(G(z)))$. The minimax two-player game can thus be formulated with value function $W(D, G)$ [10]:

$$\min_G \max_D W(D, G) = \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))] \quad (3)$$

In GAIL [9], p_{data} is the expert trajectory of state-action pairs, and G is the imitation policy π_ϕ that needs to be trained and from which state-action pairs are sampled. The goal is to make the learner's state-action distribution be close to that of the expert. For LfO problems, GAIfO [8] uses similar paradigm, replacing state-action pairs with state transitions.

IV. Learning Stabilization Control

At a high-level, our algorithm comprises two steps. We first train a neural Lyapunov-like proxy model using the expert trajectories. This step of training uses violation of the Lyapunov conditions as the loss function, adapting the conditions for asymptotic stability from Definition 3 to the LfO setting. The learned Lyapunov function subsequently provides the reward signal for training the learner's agent. For this second step of training to succeed, we need to define a transformation of the Lyapunov-like proxy model from the first step using convex functions. The overall algorithm called LSO-LLPM, short for Learning Stabilization Control from Observations through Learning Lyapunov-like proxy Models, is shown in Algorithm 1. Line $1 - 3$ is the first step of training the Lyapunov-like proxy model (Section IV-A), and Line $4 - 7$ is the second step of using Proximal Policy Optimization (PPO) [41] to optimize the learner's policy given a reward function derived from the Lyapunov-like proxy model (Section IV-B).

A. Learning Neural Lyapunov-like proxy Models

In the first step, we learn Lyapunov-like proxy models to capture the stabilization behavior of the expert. The shape of the region of attraction [11] and the convergence rate of this approximate Lyapunov landscape are important because we need to avoid misleading the learner in the second training step to pursue behaviors that cannot be achieved because of constraints in the underlying dynamics. For instance, when controlling a vehicle to track a given path, if we define an aggressive Lyapunov landscape to aim for a fast convergence rate, then the vehicle may be forced to attain a high speed

Algorithm 1 LSO-LLPM

Require: Expert state-only trajectories: $\tau_E = \{(s, s')\}$, randomly initialized policy network π_ϕ, randomly initialized Lyapunov-like proxy model V_θ, and hyperparameters $c, \beta_1, \beta_2 \in \mathbb{R}^+$.

1: **for each** $(s, s') \in \tau_E$ **do**

2: Update V_θ by taking gradient descent steps on the following loss function:

$$V_\theta^2(0) + \max(0, -V_\theta(s)) + \beta_1(c + (V_\theta(s') - V_\theta(s))/\Delta t)^2$$

3: **end for**

4: **for** $i = 0, 1, 2, ...$ **do**

5: Sample trajectories $\tau_i \sim \pi_\phi$

6: Update π_ϕ by performing PPO update steps with the following reward function:

$$g(V_\theta(s)) + \beta_2 \min(0, (V_\theta(s) - V_\theta(s'))/\Delta t)$$

7: **end for**

and fail to stabilize. On the other hand, if the Lyapunov landscape is too smooth, then the learner may attempt to take conservative actions that are not sufficient to drive the system to the target equilibrium.

We represent the Lyapunov-like proxy model over state space, $V_\theta : S \to \mathbb{R}$, as a neural network (typically two-layer fully connected networks are sufficient in our examples), which is randomly initialized. We sample observations from the expert trajectories and perform stochastic gradient descent on the parameters to minimize the violations of Lyapunov conditions using the following loss function:

$$V_\theta(0)^2 + \max(0, -V_\theta(s)) + \beta_1(c + L_f V_\theta(s))^2 \quad (4)$$

Recall that $L_f V_\theta$ is the Lie derivative of V_θ along the system dynamics f that we do not have knowledge of. Instead, we approximate the Lie derivative by the finite difference of the Lyapunov function over each discrete time step, $L_f V_\theta = (V_\theta(s') - V_\theta(s))/\Delta t$, where s and s' are consecutive state observations in the trajectory. We know that this is a good approximation when Δt is small.

The first term $V_\theta(0)^2$ ensures that the equilibrium point corresponds to a Lyapunov value of zero, which is the lowest across the state space because the second term $\max(0, -V_\theta(s))$ requires that the Lyapunov function value be non-negative at all sampled points, which ideally generalizes to other regions in the state space. The third term $\beta_1(c + L_f V_\theta(s))^2$ is a critical design. It controls the convergence rate as measured by the Lie derivative $L_f V_\theta$. Here we deviate from the standard Lyapunov conditions of only requiring the Lie derivative to be negative by forcing it to take the value of some positive constant rate c. With this requirement, the overall landscape V_θ will be shaped through learning so each step taken by the expert will be considered as a unit step toward stabilization. In this way, we capture the convergence rate by the Lyapunov-like proxy model which can then properly guide the learner in the second training step. We stop gradients for $V_\theta(s')$ in calculating $L_f V_\theta$.

B. Policy Learning from the Lyapunov-like proxy Model

After learning the Lyapunov-like proxy model, we transform it into a reward function that the learner can maximize using standard policy optimization procedures such as PPO [41]. The reward is defined as:

$$g(V_\theta(s)) + \beta_2 \min(0, (V_\theta(s) - V_\theta(s'))/\Delta t) \quad (5)$$

where $g(x)$ is a convex function (fixed through all environments) for scaling the values of the Lyapunov-like proxy models so that the learner can receive sufficiently strong reward feedback in each step. We also want to maintain stability when the agent is already close to the target equilibrium point. Suitable choices of $g(x)$ include $-\log(x)$ and $-\log(1 - e^{-kx^2})$ for some $k > 0$. The second term of Eq. 5 reduces the reward when the Lie derivative becomes positive, which prevents the learner from taking large steps near the equilibrium state. In this manner, we observe fast converge of the agent to the equilibrium state. Overall, the Lyapunov candidate function acts as a proxy for the reward function so that PPO will take steps to increase the reward and attempt to reproduce stabilization control policies that converge at a similar rate as the trajectories from the expert.

V. Experiments in Simulated Environments

We evaluate our algorithm LSO-LLPM for nonlinear control problems and compare with the state-of-the-art algorithm GAIfO. For fair comparison, we implement GAIfO using PPO without entropy loss. Evaluation environments include Automobile path-tracking control [42] and Acrobot [43], as well as high dimensional tasks like Quadrotor [44], and Hopper Standing, and Walker Standing. Acrobot is a classical control task simulated within OpenAI Gym [45], Automobile path-tracking control and Quadrotor were simulated using PythonRobotics [46], and Hopper and Walker Standing environments were simulated with MuJoCo [47]. We used NNs with $2 - 3$ hidden layers with $64 - 2048$ neurons.

Baselines and Evaluation Metrics. We focus on comparing with the state-of-the-art on-policy LfO approach GAIfO [8] because of the same assumptions of having access to observations only and on-policy training of the learner without modeling the environment transitions. In particular, GAIfO has been shown to perform consistently better than BCO [21]. There are a range of other imitation learning baselines with different additional assumptions, like LfD methods with access to actions (GAIL [9], DAC [20]), and off-policy LfO methods (OPOLO [25]) that assume ability to pre-train inverse transition models to accelerate learning progress. This performance gain is orthogonal to on-policy learning objective in our setting. Consequently, we focus on comparing with GAIfO in the evaluation and analysis.

We evaluate the performance along the following metrics: First, we measure the overall performance of the learner with respect to a varying number of sampled expert trajectories that are provided to both algorithms. Second, we analyze the learning efficiency by the learning curves over time.

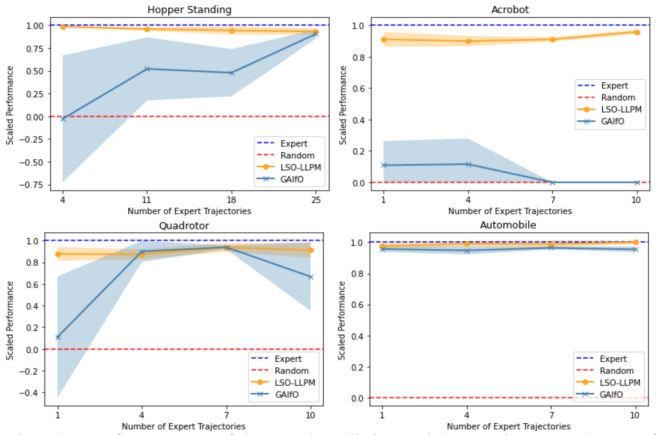

Fig. 2: Performance of learned policies with varying numbers of expert trajectories. The performance is normalized to be between 0 (average reward of random policy) and 1 (average reward of expert policy). The shaded areas show variance over 5 random seeds. We observe the proposed methods perform much better especially in environments that are harder to control, like Hopper Standing, Acrobot, and Quadrotor.

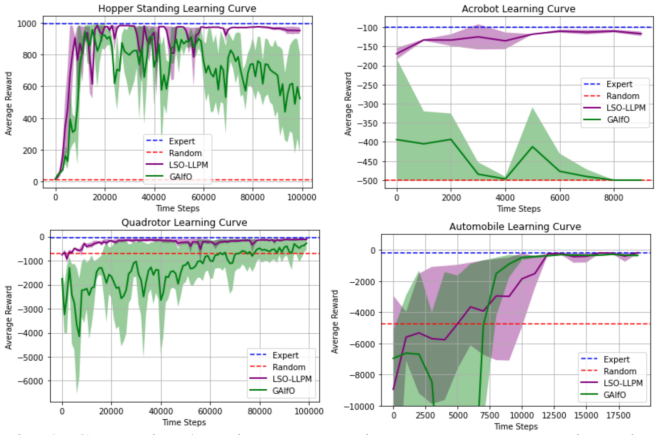

Fig. 3: Comparing learning curves using $10-18$ expert trajectories (fixed number across different methods in each environment) over 5 random seeds. Environments in first row show distribution matching approaches in GAN-based methods often experience difficulty in making consistent learning progress while Lyapunov-like proxy models generate landscapes that are suitable for stabilization tasks and thus achieve stable learning performance.

Environments. The nonlinear control tasks in each environment are specified as follows:

- *Acrobot:* The Acrobot environment consists of two links and two joints. In the initial state, both links are hanging down. The goal is to swing the links up so the tip of the link farthest from the pivot reaches the threshold in the shortest time. The state consists of information on the angles from the joints as well as their angular velocity, and the agent can actuate the joint between the links.

- *Automobile path-tracking control:* Autonomous driving is a control problem in which using speed commands the agent needs to follow a target path. In the environment, the state space is four dimensions, namely the difference between target speed and vehicle speed $V_t - v(t)$, the angular error $\theta_e(t)$, the distance to the path $d_e(t)$, and vehicle speed $v(t)$. The action space has two dimensions, namely the acceleration $a(t)$ and a steering control $\delta(t)$.

- *Quadrotor control:* We also test our algorithm in the 6-degree-of-freedom Quadrotor model. This has four control inputs and twelve state variables. The control inputs Ω_1, Ω_2, Ω_3, and Ω_4 are the angular velocities of each rotor. The state variables are the inertia frame positions (x, y, z), velocities $(\dot{x}, \dot{y}, \dot{z})$, rotation angles (ϕ, θ, ψ), and angular velocities $(\dot{\phi}, \dot{\theta}, \dot{\psi})$. More details regarding the implementation and dynamics of the Quadrotor can be found in [44], [46]. This control problem is known to be hard for policy gradient methods but is solved with learning from demonstrations.

- *Hopper and Walker Standing:* We use MuJoCo Hopper and Walker and change task reward to formulate the stabilization control problem of standing in upright position. In Hopper Standing, there is one leg with 3 joints. In Walker Standing, there are two legs with 6 joints. In both environments, the agent's goal is to maintain standing state without falling down (losing balance) for the longest time. We show graph results for Hopper Standing as the results are similar.

A. *Overall Performance*

Results: LfO Performance. We examine the performance of the policies trained by LSO-LLPM and GAIfO with different number of expert trajectories in Figure 2. LSO-LLPM reaches at least 85% of the expert performance in all environments for all number of expert trajectories tested. As shown in the figure, LSO-LLPM consistently performs better than GAIfO for all environments and number of expert trajectories. In particular, in Hopper Standing and Acrobot, LSO-LLPM is able to perform well, even when GAIfO performs not much better than the random policy baseline. This difference illustrates the important use of Lyapunov-like proxy models that capture challenging control problems in these environments due to underlying nonlinear dynamics. In these cases, GAN-based approaches performing distribution matching are often not sufficient for finding good control policies for stabilization.

Results: Learning Efficiency. The learning curves of LSO-LLPM and GAIfO in each environment are shown in Figure 3. In each environment, we use the number of expert trajectories (ranging between 10 to 18) that corresponds to the high performance cases in Figure 2. We see that across all environments, the LSO-LLPM methods can generally achieve fast learning progress with small variance across different random seeds. In comparison, GAIfO can learn well in simple control tasks such as simulated Automobile, but struggles to make progress in harder environments such as Hopper Standing and Acrobot. Note that this difficulty is not only determined by the dimensionality of the action space but also by how difficult it is to find control policies that can allow the learner to imitate the expert trajectories, and the LfO setting further increases that difficulty. For instance, although Acrobot is low-dimensional, the stabilization control problem is still very challenging. Overall, we observe Lyapunov-like proxy models can generate landscapes that

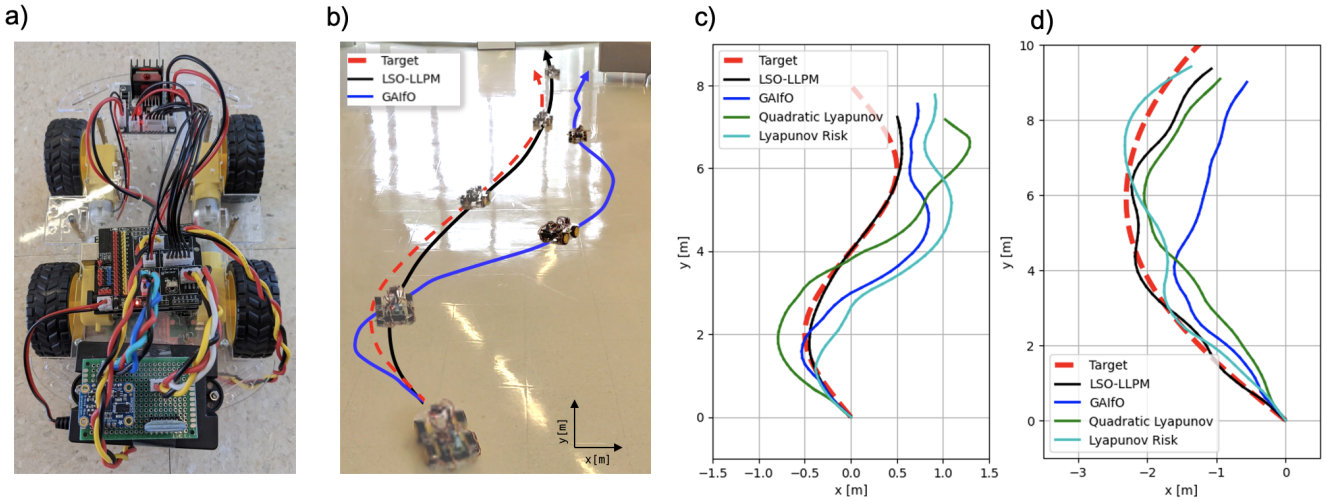

Fig. 4: (a) Car robot used for the hardware experiments. (b) Tracking target path I. The car controlled by LSO-LLPM (black line) tracked the target path (red dash line) better than GAIfO (blue line). (c), (d) All trajectories driven by LSO-LLPM, GAIfO, Quadratic Lyapunov, and Lyapunov Risk. Our proposed method tracked both target paths fairly well compared with the other methods. (c) shows the result of tracking target path I and (d) shows the result of tracking target path II.

are particularly suitable for stabilization tasks for efficient training of the learner in hard control tasks.

VI. Experiments in Hardware Environments

We present the evaluation of performance of automobile path-tracking control driven by LSO-LLPM, GAIfO, Quadratic Lyapunov, and Lyapunov Risk.

Quadratic Lyapunov In Lyapunov stability theory, a typical choice is the quadratic form (sum of squared differences between state features and their ideal locations/orientations) as default Lyapunov landscape. This is provably optimal for linear systems and often works well for mildly nonlinear systems. We consider whether a quadratic model can guide the learner without learning a Lyapunov-like proxy model.

Lyapunov Risk We also consider learning the Lyapunov Function with the generic Lyapunov Risk Loss as proposed in [48] instead of LSO-LLPM's loss expression in Eq. 4. This would demonstrate the importance of enforcing constant Lie derivative on expert trajectories in guiding the learner.

Hardware settings We tested the control task using the car robot shown in Fig. 4 (a). One IMU sensor and two photoelectric encoders are on it to calculate its velocity $v(t)$ and orientation $\theta(t)$ each time step. With these values, we obtain the difference between target speed and vehicle speed $V_t - v(t)$, the angular error $\theta_e(t)$, and the distance to the path $d_e(t)$, which are used as each policy's input. Given these input values, the policies determine acceleration $a(t)$ and steering control $\delta(t)$ at the next time step. In this hardware experiment, we set $\Delta t = 0.5$ [s], target speed $V_t = 0.3$ [m/s]. The policies are trained under a simulator with these parameter settings, and they are directly deployed to the robot car for testing.

Results We tested the performance of automobile path-tracking with two different target paths, Target path I and II. Fig. 4 (b) and (c) are the results of tracking target path I, and Fig. 4 (d) shows the result of tracking target path II. The red dot lines are the target paths given in advance, and the closer the trajectories are to them, the better their control policies. As seen from these figures, the car driven by LSO-LLPM (black lines) tracked the target paths fairly well compared with the other methods. In particular, GAIfO (blue lines) showed low path tracking performance as the trajectories controlled by it gradually deviated from the target paths. The main strength of our approach is obtaining stable control policies through LfO training process by utilizing Lyapunov-like proxy models. This enables more stable tracking performances even for the hardware experiments, in which are various disturbances such as sensor noises and discrepancies between simulator and real hardware modeling.

VII. Discussion and Conclusion

We have introduced a novel model-free Lyapunov-based method to accelerate Learning from Observations for stabilization control problems by introducing intermediate proxy models between the expert and the agent policy based on Lyapunov stability theory. Our LfO training process first learns a Lyapunov landscape model from the expert state sequences and then transforms the learned model to guide the training of the learner's policy. We showed the proposed methods can capture stabilization control behaviors that take into account underlying dynamics so the learner's agent can successfully recover stable control policies through policy optimization. We evaluated the proposed methods in various real and simulated nonlinear stabilization control environments and observed better learning efficiency compared to the state-of-the-art GAN-based approaches.

We primarily focused on stabilization to a fixed equilibrium. To handle more general control tasks, it is possible to extend our approach by incorporating time in the Lyapunov-like proxy models. Such extension may work on more control environments with general locomotion tasks.

REFERENCES

[1] Daniel Dewey. Reinforcement learning and the reward engineering principle. In *2014 AAAI Spring Symposium Series*, 2014.

[2] Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 79–86, 2017.

[3] Petar Kormushev, Sylvain Calinon, and Darwin Gordon Caldwell. Robot motor skill coordination with em-based reinforcement learning. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3232–3237, 2010.

[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[5] Brenna Argall, S. Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics Auton. Syst.*, 57:469–483, 2009.

[6] Ahmed Hussein, Mohamed Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 50, 04 2017.

[7] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *CoRR*, abs/1905.13566, 2019.

[8] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *CoRR*, abs/1807.06158, 2018.

[9] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[11] Wassim Haddad and Vijaysekhar Chellaboina. Nonlinear dynamical systems and control: A lyapunov-based approach. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*, 01 2008.

[12] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *ICLR*, 2018.

[13] Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *AAAI*, 2018.

[14] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *ICLR*, 2019.

[15] Jiankai Sun, Lantao Yu, Pinqian Dong, Bo Lu, and Bolei Zhou. Adversarial inverse reinforcement learning with self-attention dynamics model. *IEEE Robotics and Automation Letters*, 6(2):1880–1886, 2021.

[16] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995.

[17] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1988.

[18] Pieter Abbeel and Andrew Ng. Apprenticeship learning via inverse reinforcement learning. *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, 09 2004.

[19] Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*, 05 2000.

[20] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *ICLR*, 2019.

[21] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *CoRR*, abs/1805.01954, 2018.

[22] Michael Laskey, Sam Staszak, Wesley Yu-Shu Hsieh, Jeffrey Mahler, Florian T. Pokorny, Anca D. Dragan, and Ken Goldberg. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 462–469, 2016.

[23] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[24] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010.

[25] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12402–12413. Curran Associates, Inc., 2020.

[26] Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[27] Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. Provably efficient imitation learning from observation alone. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6036–6045. PMLR, 09–15 Jun 2019.

[28] Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1755–1763. PMLR, 09–15 Jun 2019.

[29] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8092–8101. Curran Associates, Inc., 2018.

[30] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Mohammad Ghavamzadeh, and Edgar A. Duéñez-Guzmán. Lyapunov-based safe policy optimization for continuous control. *CoRR*, abs/1901.10031, 2019.

[31] Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters and IROS*, 2020.

[32] Ming Jin and Javad Lavaei. Stability-certified reinforcement learning: A control-theoretic perspective. *IEEE Access*, 8:229086–229100, 2020.

[33] Ya-Chien Chang and Sicun Gao. Stabilizing neural control using self-learned almost lyapunov critics. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1803–1809, 2021.

[34] He Yin, Peter Seiler, Ming Jin, and Murat Arcak. Imitation learning with stability and safety guarantees. *IEEE Control Systems Letters*, 6:409–414, 01 2022.

[35] S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[36] S. Mohammad Khansari-Zadeh and Aude Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765, 2014.

[37] Yonadav Shavit, Nadia Figueroa, Seyed Sina Mirrazavi Salehian, and Aude Billard. Learning augmented joint-space task-oriented dynamical systems: A linear parameter varying and synergetic control approach. *IEEE Robotics and Automation Letters*, 3(3):2718–2725, 2018.

[38] Nadia Figueroa and Aude Billard. A physically-consistent bayesian non-parametric mixture model for dynamical system learning. In *CoRL*, 2018.

[39] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994.

[40] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and

Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[42] Jarrod M. Snider. Automatic steering methods for autonomous automobile path tracking. 2009.

[43] Alborz Geramifard, Christoph Dann, Robert H. Klein, William Dabney, and Jonathan P. How. Rlpy: A value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16(46):1573–1578, 2015.

[44] Bartomeu Rubí, Ramon Pérez, and Bernardo Morcego. A survey of path following control strategies for uavs focused on quadrotors. *Journal of Intelligent & Robotic Systems*, 98, 05 2020.

[45] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[46] Atsushi Sakai, Daniel Ingram, Joseph Dinius, Karan Chawla, Antonin Raffin, and Alexis Paques. Pythonrobotics: a python code collection of robotics algorithms, 2018.

[47] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[48] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.