Stable and Safe Human-aligned Reinforcement Learning through Neural Ordinary Differential Equations

Liqun Zhao
Department of Engineering Science
University of Oxford
Oxford, UK
liqun.zhao@eng.ox.ac.uk

Keyan Miao
Department of Engineering Science
University of Oxford
Oxford, UK
keyan.miao@eng.ox.ac.uk

Konstantinos Gatsis
School of Electronics and Computer Science
University of Southampton
Southampton, UK
k.gatsis@soton.ac.uk

Antonis Papachristodoulou
Department of Engineering Science
University of Oxford
Oxford, UK
antonis@eng.ox.ac.uk

***Abstract*—Reinforcement learning (RL) excels in applications such as video games, but ensuring safety as well as the ability to achieve the specified goals remains challenging when using RL for real-world problems, such as human-aligned tasks where human safety is paramount. This paper provides safety and stability definitions for such human-aligned tasks, and then proposes an algorithm that leverages neural ordinary differential equations (NODEs) to predict human and robot movements and integrates the control barrier function (CBF) and control Lyapunov function (CLF) with the actor-critic method to help to maintain the safety and stability for human-aligned tasks. Simulation results show that the algorithm helps the controlled robot to reach the desired goal state with fewer safety violations and better sample efficiency compared to other methods in a human-aligned task.**

***Index Terms*—Safety, stability, human-aligned reinforcement learning**

I. Introduction

Safety in robotics has become a focal point in current research activity [1]–[4], and RL approaches have been applied to ensure safety in human-robot scenarios [5], [6]. Recently, an increasing interest has emerged in integrating control approaches with RL to help ensure safety in real-world tasks, and concepts like control barrier function (CBF) [7]–[9] have been used as safety constraints. However, in many studies, the ability of the robot to achieve the designed goal state is not considered, and a physics-based control-affine nominal model of the system is required [7], [10].

Maintaining stability for a control system is also paramount in human-aligned tasks, and concepts like Lyapunov functions are currently used in learning to help guarantee the stability [11]–[14]. However, these model-free algorithms are often data-intensive, and therefore, an algorithm with a higher sample efficiency may be desired for better applications in real-world scenarios when human exists.

Our contributions. 1. We introduce a primary controller that combines the CBF and Control Lyapunov Function (CLF) with the Soft Actor-Critic (SAC) algorithm [15] for human-aligned tasks where human and robot movements are approximated by neural ordinary differential equations (NODEs) [16]. 2. We propose an RL-based backup controller that prioritizes safety constraints when satisfying both safety and stability constraints simultaneously is not possible. 3. We combine the two controllers together to propose a new algorithm and show its better performance on one simulation scenario where human drivers exist compared to other baselines.

II. Background

A. Preliminaries

1) Markov Decision Process: A Markov decision process (MDP) is defined by the tuple $\mathcal{M}$ which is $(\mathcal{X},\mathcal{U},\mathcal{F},r,c,\gamma,\gamma_c)$. $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$ are state and control signal spaces, and $x_{t_k} \in \mathcal{X}$ is the state at timestep t_k, $u_{t_k} \in \mathcal{U}$ is the control signal at timestep t_k. $\mathcal{F}$ denotes the dynamics of the whole system including both human and the controlled robot, and by applying the control signal u_{t_k} between timesteps t_k and t_{k+1}, there is:

$$x_{t_{k+1}} = x_{t_k} + \int_{t_k}^{t_{k+1}} \mathcal{F}(\chi, x_\chi, u_{t_k}) d\chi. \tag{1}$$

The reward and cost are denoted as r and c, and γ and γ_c are the discount factors. The transition probability is defined as $P(x_{t_{k+1}}|x_{t_k}, u_{t_k}) \triangleq I_{\{x_{t_{k+1}} = x_{t_k} + \int_{t_k}^{t_{k+1}} \mathcal{F}(\chi, x_\chi, u_{t_k}) d\chi\}}$, where I is a function that equals 1 if $x_{t_{k+1}}$ satisfies Eq.(1), and 0 otherwise. Following Han et al. [11], the closed-loop transition probability is denoted as $P_\pi(x_{t_{k+1}}|x_{t_k}) \triangleq \int_{\mathcal{U}} \pi(u_{t_k}|x_{t_k}) P(x_{t_{k+1}}|x_{t_k}, u_{t_k}) du_{t_k}$. Additionally, the closed-loop state distribution at timestep t_k is denoted by

$\upsilon(x_{t_k}|\rho,\pi,t_k)$, which is calculated as $\upsilon(x_{t_{k+1}}|\rho,\pi,t_{k+1}) = \int_{\mathcal{X}} P_\pi(x_{t_{k+1}}|x_{t_k})\upsilon(x_{t_k}|\rho,\pi,t_k)dx_{t_k}, \forall t_k \in \mathbb{N}$. The initial state distribution is $\upsilon(x_{t_0}|\rho,\pi,t_0) = \rho$.

2) Definitions of Safety and Stability: If there are m safety constraints (for example, the controlled robot is required to avoid colliding with m human users), the system is safe if $h_i(x_{t_k}) \geq 0, \forall t_k \geq t_0$ holds for $i = 1,\ldots,m$. Here $h_i : \mathbb{R}^n \to \mathbb{R}$ is a function corresponding to the i^{th} safety constraint, and a safe set $\mathcal{C}_{i,0} \subset \mathbb{R}^n$ is defined as follows:

$$\mathcal{C}_{i,0} = \{x_{t_k} \in \mathcal{X} | h_i(x_{t_k}) \geq 0\}. \tag{2}$$

We require the system state to remain within this set $\mathcal{C}_{i,0}$. When the relative degree of the constraint $h_i(x_{t_k}) \geq 0$ is r, the discrete-time CBF can be used to ensure the forward invariance of the safe set, and a list of functions can be defined:

$$\begin{aligned}
&\Phi_{i,0}(x_{t_k}) := h_i(x_{t_k}) \\
&\Phi_{i,1}(x_{t_k}) := \Delta\Phi_{i,0}(x_{t_k},u_{t_k}) + \kappa_{i,1}(\Phi_{i,0}(x_{t_k})) \\
&\qquad\vdots \\
&\Phi_{i,r}(x_{t_k}) := \Delta\Phi_{i,r-1}(x_{t_k},u_{t_k}) + \kappa_{i,r}(\Phi_{i,r-1}(x_{t_k}))
\end{aligned} \tag{3}$$

where $\Delta\Phi_{i,j}(x_{t_k},u_{t_k}) := \Phi_{i,j}(x_{t_{k+1}}) - \Phi_{i,j}(x_{t_k})$, $j = 0,1,\ldots,r-1$, and $\kappa_{i,j}(\cdot)$ are class $\mathcal{K}$ functions. Then the definition of the discrete-time CBF can be given as:

Definition 1 (Discrete-time Control Barrier Function [17]): For the system described by Eq.(1), the function $h_i : \mathbb{R}^n \to \mathbb{R}$ is called a discrete-time CBF of relative degree r if there exists $\Phi_{i,j}(x_{t_k})$, $j = 0,1,\ldots,r$ defined by Eqs.(3) and $\mathcal{C}_{i,j}$ which are defined similarly to Eq.(2), $j = 0,1,\ldots,r-1$ such that for all $x_{t_k} \in \bigcap_{j=0}^{r-1}\mathcal{C}_{i,j}$,

$$\Phi_{i,r}(x_{t_k}) \geq 0. \tag{4}$$

A controller satisfying Ineq.(4) can make the set $\bigcap_{j=0}^{r-1}\mathcal{C}_{i,j}$ forward invariant, and therefore, safety is maintained if there exists a controller such that $\forall i \in [1,m]$, Ineq.(4) holds for all $x_{t_k} \in \bigcap_{i=1}^{m}\bigcap_{j=0}^{r-1}\mathcal{C}_{i,j}$.

In a stabilization task, the system state is required to finally reach a goal state (for example. keeping a specified distance from a human user). Therefore, we define the cost signal as $c(x_{t_k},u_{t_k}) = \left\|x_{t_{k+1}} - x_{\text{desired}}\right\|$ where $x_{t_{k+1}}$ is the next state following Eq.(1), and x_{desired} denotes the goal state (for example, the desired distance). Furthermore, define the cost function under the controller π as $c_\pi(x_{t_k}) = \mathbb{E}_{u_{t_k}\sim\pi}c(x_{t_k},u_{t_k}) = \mathbb{E}_{u_{t_k}\sim\pi}[\left\|x_{t_{k+1}} - x_{\text{desired}}\right\|]$, and naturally, we expect that the value of $c_\pi(x_{t_k})$ decreases as t_k increases, and achieve $c_\pi(x_{t_k}) = 0$ eventually, which means the agent reaches the goal state. Similar to Han et al. [11], we provide the following definition.

Definition 2 (Stability in Mean Cost): Let υ denote the closed-loop state distribution, the goal state is said to be stable in mean cost if there exists a positive constant b such that the condition $\lim_{t_k\to\infty}\mathbb{E}_{x_{t_k}\sim\upsilon}\left[c_\pi(x_{t_k})\right] = 0$ holds for any initial state $x_{t_0} \in \{x_{t_0}|c_\pi(x_{t_0}) < b\}$. If b is arbitrarily large, the goal state is globally stable in mean cost.

3) Introduction to Neural Ordinary Equations (NODEs): According to Eqs.(3), applying CBFs requires the system dynamics to obtain the future states $\{x_{t_{k+1}}, x_{t_{k+2}}, \ldots, x_{t_{k+r}}\}$ of the whole system including both human and the controlled robot. However, in many cases, obtaining system dynamics or even a nominal model directly based on the physics law is difficult. Neural networks are important tools to estimate the dynamics, and there has been a rise in considering neural network hidden layers as states [18]. Chen et al. [16] introduces an ODE to approximate dynamics by neural networks as follows:

$$\begin{cases} \dot{\hat{x}}_{t_k} = \mathcal{F}_\psi\left(t_k, \hat{x}_{t_k}, u_{t_k}\right) \\ \hat{x}_{t_0} = x_{t_0} \end{cases} \tag{5}$$

where $\mathcal{F}_\psi$ is a neural network with parameter ψ. By assuming a constant control signal, i.e., $u_\chi = u_{t_k}, \chi \in [t_k, t_{k+1})$, the inference of NODEs is:

$$\hat{x}_{t_{k+1}} = \hat{x}_{t_k} + \int_{t_k}^{t_{k+1}} \mathcal{F}_\psi\left(\chi, \hat{x}_\chi, u_{t_k}\right) d\chi \tag{6}$$

which is used to estimate Eq.(1). Then we can approximate all future states required for constructing Ineq.(4) by iteratively using Eq.(6).

B. Definition of the Safe and Stable Control Problem

Similar to Dawson et al. [19], here we define:

Problem 1 (Safe and Stable Control Problem): For system described by Eq.(1), given a desired goal state x_{desired}, a set $\mathcal{X}_0$ which is the set of x_{t_0} denoting the initial states, an unsafe set $\mathcal{X}_{\text{unsafe}} \subseteq \mathcal{X}$, and a safe set $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$ such that $x_{\text{desired}} \in \mathcal{X}_{\text{safe}}$ and $\mathcal{X}_{\text{safe}} \cap \mathcal{X}_0 \neq \emptyset$, find a controller π producing sequence $\{u_{t_k}\}_{t_k \geq t_0}$ such that the state sequence $\{x_{t_k}\}_{t_k \geq t_0}$ satisfying Eq.(1) and $x_{t_0} \in \mathcal{X}_{\text{safe}} \cap \mathcal{X}_0$ satisfy: 1. **Safety**: $x_{t_k} \in \mathcal{X}_{\text{safe}}\ \forall t_k \geq t_0$. 2. **Stability in Mean Cost at the Equilibrium:** $\lim_{t_k\to\infty}\mathbb{E}_{x_{t_k}\sim\upsilon}\left[c_\pi(x_{t_k})\right] = 0$.

When an exact expression of the real dynamics based on physics law is not available, we can utilize NODEs to approximate the real dynamics to construct constraints. See Section III for a detailed description.

III. Framework Design

A. Learning the Dynamics via NODEs and Value Function of the Cost

A good approximated system dynamics to predict the human and controlled robot movements is beneficial to the overall training. To obtain the NODE model, we collect state sequences $X = \{x_{t_k}, x_{t_{k+1}}, \ldots, x_{t_{k+h}}\}$ and control sequences $U = \{u_{t_k}, u_{t_{k+1}}, \ldots, u_{t_{k+h-1}}\}$ during the real interaction process following the real dynamics. The model then computes the approximated state sequence based on Eq.(6). Model loss is designed as $\ell = \frac{1}{h}\sum_{i=1}^{h}|x_{t_{k+i}} - \hat{x}_{t_{k+i}}|$, and the training process is illustrated as Algorithm 1.

To help maintain stability for the system, inspired by commonly-used value functions in RL, we define the value function of the cost at the state x_{t_k} as $L_\pi(x_{t_k}) =$

$\mathbb{E}_{\tau\sim\pi}\big[\sum_{i=0}^{\infty}\gamma_c^i c_\pi(x_{t_{k+i}})\big]$, where τ is a trajectory under controller π starting from the initial state x_{t_k}. Based on this definition, $L_\pi(x_{t_k})$ can also be approximated by a neural network, and a natural strategy to achieve stability is to introduce a condition that drives the value of $L_\pi(x_{t_k})$ to decrease along the trajectory τ. Inspired by the concept of CLF, the following condition proposed in Zhao et al. [20] can be utilized:

$$\mathbb{E}_{x_{t_k}\sim\mu_\pi, x_{t_{k+1}}\sim P_\pi}\big[L_\pi(x_{t_{k+1}})-L_\pi(x_{t_k})\big]\leq-\beta\mathbb{E}_{x_{t_k}\sim\mu_\pi}\big[L_\pi(x_{t_k})\big]. \tag{7}$$

$\mu_\pi(x)\triangleq\lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N}\upsilon(x_{t_k}=x|\rho,\pi,t_k)$ is the sampling distribution. Later, we introduce a method to obtain a controller leading to an L_π satisfying Ineq.(7).

Algorithm 1 NODE-based Dynamics Learning for Human and Controlled Robot Movements Prediction

Input: Learning rate η_1

Collect trajectories X and control variable sequences U
Initialize dynamics model $\mathcal{F}_\psi$.
for $i=1$ **to** h **do**
  $\hat{x}_{t_{k+i}}=\hat{x}_{t_{k+i-1}}+\int_{t_{k+i-1}}^{t_{k+i}}\mathcal{F}_\psi\left(\chi,\hat{x}_\chi,u_{t_{k+i-1}}\right)d\chi$
end for
$\psi\leftarrow\psi-\eta_1\nabla_\psi\ell$

Output: ψ

B. Augmented Lagrangian Method for Parameter Updating

We denote the parameters of the RL-based primary controller and two action-value networks Q^{π_p} by θ_p and ϕ_i, where $i=1,2$, respectively. Additionally, we employ L_ν, referred to as the Lyapunov network, to approximate L_{π_p} with parameters ν. According to the previous sections, by calculating expected values with predicted states, the RL problem with CBF and CLF constraints can be formed as follows:

$$\begin{aligned}&\min_{\theta_p}\ -V^{\theta_p}\\ &s.t.\ \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[-\Phi_{i,r}(x_{t_k})\big]\leq 0\qquad\forall i\in[1,m],\\ &\quad\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[L_\nu(\hat{x}_{t_{k+1}})-L_\nu(x_{t_k})+\beta L_\nu(x_{t_k})\big]\leq 0,\end{aligned} \tag{8}$$

where $\mu_{\pi_p}(x)$ is the sampling distribution under the controller π_p, $-V^{\theta_p}$ is the objective function commonly used in SAC. Loss functions for updating the action-value networks $Q_{\phi_i}^{\pi_p}, i=1,2$, coefficient α_p, and Lyapunov network L_ν are:

$$J_{Q^{\pi_p}}(Q_{\phi_i}^{\pi_p})=\mathbb{E}_{(x_{t_k},u_{t_k},r_{t_k},x_{t_{k+1}})\sim\mathcal{D},\xi\sim\mathcal{N}}\Big[\big(r_{t_k}+\gamma\big(\min_{j=1,2}Q_{targ,\phi_j}^{\pi_p}(x_{t_{k+1}},\tilde{u}_{\theta_p}(x_{t_{k+1}},\xi))-\alpha_p\log\pi_{\theta_p}(\tilde{u}_{\theta_p}(x_{t_{k+1}},\xi)|x_{t_{k+1}})\big)-Q_{\phi_i}^{\pi_p}(x_{t_k},u_{t_k})\big)^2\Big], \tag{9}$$

$$J_{\alpha_p}(\alpha_p)=-\alpha_p\times\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},\xi\sim\mathcal{N}}\big[\log\pi_{\theta_p}(\tilde{u}_{\theta_p}(x_{t_k},\xi)|x_{t_k})+\mathcal{H}\big], \tag{10}$$

$$J_L(L_\nu)=\mathbb{E}_{(x_{t_k},c_{t_k},x_{t_{k+1}})\sim\mathcal{D}}\Big[\big[c_{t_k}+\gamma_c L_{targ,\nu}(x_{t_{k+1}})-L_\nu(x_{t_k})\big]^2\Big], \tag{11}$$

where $Q_{targ,\phi_i}, i=1,2$ are the target action-value networks, $\mathcal{H}$ is a predefined threshold, and $\mathcal{D}$ denotes the batch of transitions following π_p. By introducing a vector of additional variables $\boldsymbol{z_p}=(z_{1,p}^2,z_{2,p}^2,\cdots,z_{m,p}^2,z_{m+1,p}^2)$, we can convert the Problem (8) to the following problem:

$$\begin{aligned}&\min_{\theta_p,\boldsymbol{z_p}}\ -V^{\theta_p}\\ &s.t.\ \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[-\Phi_{i,r}(x_{t_k})\big]+z_{i,p}^2=0\ \ \forall i\in[1,m],\\ &\quad\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[L_\nu(\hat{x}_{t_{k+1}})-L_\nu(x_{t_k})+\beta L_\nu(x_{t_k})\big]+z_{m+1,p}^2=0.\end{aligned} \tag{12}$$

Denote the Lagrangian multipliers for CBF and CLF constraints for this primary controller as $\lambda_{i,p}$ and ζ, respectively, and the penalty parameter as c_p. Furthermore, define $f_{i,p}(\theta_p)\triangleq\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[-\Phi_{i,r}(x_{t_k})\big]$, $\forall i\in[1,m]$, and $g(\theta_p)\triangleq\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[L_\nu(\hat{x}_{t_{k+1}})-L_\nu(x_{t_k})+\beta L_\nu(x_{t_k})\big]$, the augmented Lagrangian function is then $\mathcal{L}_{c_p}^p(\theta_p,\lambda_{i,p},\zeta)=-V^{\theta_p}+\sum_{i=1}^m\lambda_{i,p}\times(f_{i,p}(\theta_p)+z_{i,p}^2)+\sum_{i=1}^m\frac{c_p}{2}\times(f_{i,p}(\theta_p)+z_{i,p}^2)^2+\zeta\times(g(\theta_p)+z_{m+1,p}^2)+\frac{c_p}{2}\times(g(\theta_p)+z_{m+1,p}^2)^2$. The updates of parameters are in Algorithm 2.

We also introduce some tricks used in implementations. Firstly, a different timescale method is applied similar to Yu et al. [22], and furthermore, we set n_m, n_L, and n_b to update the NODEs model, Lagrangian multipliers and the backup controller with delayed update tricks [23]. Also, in practical implementation, we sample transition pairs from the replay buffer $\mathcal{B}$ for updating parameters, and before calculating the expected values for the constraints in Problem (8), we first apply the ReLU function to CBF and CLF constraints at each sampled x_{t_k}.

C. Backup Controller Design

Due to the existence of multiple constraints, the feasibility of the Problem (8) becomes a crucial problem. Priority is given to safety constraints when both constraints cannot be satisfied simultaneously, therefore, we design an RL-based backup controller π_b parameterized by θ_b by formulating an additional constrained optimization problem as follows:

$$\begin{aligned}&\min_{\theta_b}\ -V^{\theta_b}\\ &s.t.\ \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_b}\big[-\Phi_{i,r}(x_{t_k})\big]\leq 0\ \ \forall i\in[1,m],\end{aligned} \tag{13}$$

and the predicted states are now following the backup controller π_b. The objective function used for the backup controller is used to maximize the cumulative discounted reward if we use the backup controller π_b for one step.

For simplicity, we define $f_{i,b}(\theta_b)\triangleq\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_b}\big[-\Phi_{i,r}(x_{t_k})\big]$, $\forall i\in[1,m]$. Therefore, the augmented Lagrangian function for updating the backup controller is given as $\mathcal{L}_{c_b}^b(\theta_b,\lambda_{i,b})=-V^{\theta_b}+\sum_{i=1}^m\lambda_{i,b}\times(f_{i,b}(\theta_b)+z_{i,b}^2)+\sum_{i=1}^m\frac{c_b}{2}\times(f_{i,b}(\theta_b)+z_{i,b}^2)^2$, where $\lambda_{i,b}$, c_b and $\boldsymbol{z_b}=(z_{1,b}^2,z_{2,b}^2,\cdots,z_{m,b}^2)$ are Lagrangian multipliers for CBF constraints, the penalty parameter, and additional variables. Also, in real implementation, we sample the data from the replay buffer $\mathcal{B}$, and apply the ReLU function to CBF constraints at each sampled x_{t_k}. The framework combining the primary and backup controllers can be summarized as Algorithm 2.

Algorithm 2 Neural Ordinary Differential Equations-based Lyapunov-Barrier Actor-Critic

Input: Number of steps $N = 0$, an initialized NODE model parameterized by φ, action-value networks $Q_{\phi_i}^{\pi_p}$, Lyapunov network L_ν, primary controller network π_{θ_p}, coefficient α_p and Lagrange multipliers $\lambda_{i,p}$ and ζ for the primary controller, RL-based backup controller network π_{θ_b}, coefficient α_b and Lagrange multipliers $\lambda_{i,b}$ for the backup controller, replay buffer $\mathcal{B}$, coefficients of quadratic terms c_p and c_b, quadratic term coefficient factor $\rho_c \in (1, \infty)$, learning rates η_1, η_2, η_3. n_m, n_L, and n_b which are delay steps for NODE model, Lagrangian multipliers, and backup controller updates, respectively.

for each episode **do**
  for each step **do**
    $N \leftarrow N + 1$
    if $N \bmod n_m = 0$ **then**
      Update the NODEs model with data collected during the controller learning process:
      $\psi \leftarrow \psi - \eta_1 \nabla_\psi \ell$
    end if
    Construct CBF and CLF constraints with the NODE model and transition pairs from $\mathcal{B}$
    Update the Lyapunov network and action-value networks:
    $\nu \leftarrow \nu - \eta_2 \nabla_\nu J_L(L_\nu)$; $\phi_i \leftarrow \phi_i - \eta_2 \nabla_{\phi_i} J_{Q^{\pi_p}}(Q_{\phi_i}^{\pi_p})$ for $i \in \{1, 2\}$
    Update the primary controller network and its coefficient α_p:
    $\theta_p \leftarrow \theta_p - \eta_3 \nabla_{\theta_p} \mathcal{L}_{c_p}^p(\theta_p, \lambda_{i,p}, \zeta)$; $\alpha_p \leftarrow \alpha_p - \eta_3 \nabla_{\alpha_p} J_{\alpha_p}(\alpha_p)$
    $c_p \leftarrow \rho_c \times c_p$
    if $N \bmod n_L = 0$ **then**
      Update the Lagrangian multipliers $\lambda_{i,p}$ and ζ according to Bertsekas et al. [21]:
      $\lambda_{i,p} \leftarrow \max\{0, \lambda_{i,p} + c_p f_{i,p}(\theta_p)\}$;
      $\zeta \leftarrow \max\{0, \zeta + c_p g(\theta_p)\}$
    end if
    if $N \bmod n_b = 0$ **then**
      Update the backup controller network and its coefficient α_b:
      $\theta_b \leftarrow \theta_b - \eta_3 \nabla_{\theta_b} \mathcal{L}_{c_b}^b(\theta_b, \lambda_{i,b})$;
      $\alpha_b \leftarrow \alpha_b - \eta_3 \nabla_{\alpha_b} J_{\alpha_b}(\alpha_b)$
      $c_b \leftarrow \rho_c \times c_b$
      if $N \bmod (n_b \times n_L) = 0$ **then**
        Update the Lagrangian multipliers $\lambda_{i,b}$:
        $\lambda_{i,b} \leftarrow \max\{0, \lambda_{i,b} + c_b f_{i,b}(\theta_b)\}$
      end if
    end if
    if Backup controller should be used according to the condition specific to the task **then**
      $u_{t_k} \sim \pi_{\theta_b}(u_{t_k}|x_{t_k})$ and apply control signal u_{t_k}
    else
      $u_{t_k} \sim \pi_{\theta_p}(u_{t_k}|x_{t_k})$ and apply control signal u_{t_k}
      Store the transition pair $(x_{t_k}, u_{t_k}, r_{t_k}, c_{t_k}, x_{t_{k+1}})$ in $\mathcal{B}$
    end if
  end for
end for

Output: π_{θ_p}, π_{θ_b}, $Q_{\phi_i}^{\pi_p}$, $i = 1, 2$, and L_ν

IV. Simulations

Currently, there are many studies applying machine learning algorithms to solve problems in the field of transportation [24]–[30], and therefore, we conduct experiments on the task called "Simulated Car Following", and we use SAC-RCBF [31], MBPPO-Lagrangian [32], LAC [11], CPO [33], PPO-Lagrangian and TRPO-Lagrangian [34] as baselines. The environment "Simulated Car Following" is modified (but different) from Zhao et al. [20]. This task involves a chain of five cars, four of which are driven by human drivers while the 4^{th} one is the robot car controlled by an RL-based controller. These five cars are following each other on a straight road, and the goal is to control the acceleration of the 4^{th} car to maintain a desired distance from the 3^{rd} car while avoiding collisions with other cars, namely achieving the goal state while maintaining safety when human drivers exist. The movements of cars driven by human drivers are given by:

$$\dot{x}_{t_k,i} = \begin{bmatrix} v_{t_k,i} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 + d_i \end{bmatrix} a_{t_k,i} \qquad \forall i \in \{1, 2, 3, 5\}.$$

Each state of the system is denoted as $x_{t_k,i} = [p_{t_k,i}, v_{t_k,i}]^T$, indicating the position $p_{t_k,i}$ and velocity $v_{t_k,i}$ of the i^{th} car at the timestep t_k, $d_i = 0.1$. The time interval used in this experiment is 0.02s. The predefined velocity of the 1^{st} car is $v_s - 4\sin(t_k)$, where $v_s = 3.0$. Its acceleration is given as $a_{t_k,1} = k_v(v_s - 4\sin(t_k) - v_{t_k,1})$ where $k_v = 4.0$. The human drivers of the following cars decide the accelerations of the 2^{nd}, 3^{rd}, and 5^{th} car, and can change their decisions sharply. Accelerations of Car 2 and 3 are given by:

$$a_{t_k,i} = \begin{cases} k_v(v_s - v_{t_k,i}) - k_b(p_{t_k,i-1} - p_{t_k,i}) & if\ |p_{t_k,i-1} - p_{t_k,i}| < 6.5 \\ k_v(v_s - v_{t_k,i}) & otherwise, \end{cases}$$

where $k_b = 20.0$ and $i = 2, 3$. The acceleration of the 5^{th} car is:

$$a_{t_k,5} = \begin{cases} k_v(v_s - v_{t_k,5}) - k_b(p_{t_k,3} - p_{t_k,5}) & if\ |p_{t_k,3} - p_{t_k,5}| < 13.0 \\ k_v(v_s - v_{t_k,5}) & otherwise, \end{cases}$$

The model of the 4^{th} car is as follows:

$$\dot{x}_{t_k,4} = \begin{bmatrix} v_{t_k,4} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1.0 \end{bmatrix} u_{t_k},$$

where u_{t_k} is the acceleration of the 4^{th} car (robot), and also the control signal generated by the RL-based controller at the timestep t_k. The reward signal is defined to minimize the overall control effort, and an additional reward of 2.0 is granted during timesteps when $d_{t_k} = p_{t_k,3} - p_{t_k,4}$, which is the distance between the 3^{rd} and 4^{th} car, falls within $[9.0, 10.0]$. This range is defined as the desired region for d_{t_k}, and as this task rewards the system when it tries to maintain the desired state (distance), we employ cumulative reward as a metric, and a higher cumulative reward indicates better convergence to the desired goal state. The cost signal is determined as $\left\|d_{t_{k+1}} - d_{\text{desired}}\right\|$, where $d_{\text{desired}} = 9.5$. CBFs are defined as $h_1(x_{t_k}) = p_{t_k,3} - p_{t_k,4} - \delta$ and $h_2(x_{t_k}) = p_{t_k,4} - p_{t_k,5} - \delta$,

with δ being the minimum required distance between the cars. Hence, the relative degree is 2 and the planning horizon for making predictions using NODEs is 2. When a safety constraint is violated if two types of constraints cannot be satisfied simultaneously, the 4^{th} car might be in close proximity to the 5^{th} human driver in order to make d_{t_k} be within $[9.0, 10.0]$. In such cases, the backup controller is activated. The primary controller is reactivated when the 4^{th} car moves beyond the dangerous area, namely out of the vicinity of the 5^{th} car, or when the predetermined time threshold for utilizing the backup controller is exceeded.

Noted that when we use NODEs to model this system, the input of the network $\mathcal{F}$ which is structured by MLP is $(t_k, \hat{x}_{t_k}, u_{t_k})$ with the dimension of 12, and the output dimension is 10. As illustrated in Fig. 1, the proposed method consistently achieves the highest cumulative reward. This outcome indicates its exceptional capability in effectively regulating the distance d_{t_k} within the desired range $[9.0, 10.0]$, which is the desired goal. Moreover, the cumulative number of safety violations caused by the proposed method is considerably smaller than those of others after some episodes and finally decreases to almost 0. Even compared to the SAC-RCBF algorithm where a good nominal model is used and Gaussian processes (GPs) are applied to approximate the differences between the real and nominal dynamics, our method which does not require prior knowledge (for example, a nominal model) of the system dynamics always achieves higher reward during the whole training, and comparatively good performance in maintaining zero safety violations in the latter part of the training process. This shows that our method can achieve and maintain both safety and stability in fewer iterations of training in a human-robot scenario compared to the model-based and model-free baselines, which makes this proposed method appropriate to be utilized in real-world applications where algorithms should be sample efficient. It is also noteworthy that methods using GPs to help approximate the dynamics [20] face the potential problem of having a larger computational burden since GP is a non-parametric method that scales poorly with the size of the data collected, while NODE does not have this problem.

V. CONCLUSIONS

This paper introduces a method that helps guarantee both the safety and stability for human-robot scenarios where the movements of both human and robot are approximated by NODEs, and the experiments show higher cumulative rewards and fewer safety violations with better sample efficiency.

However, this method has limitations. For example, although good performance is achieved in our task, the difference between the real dynamics and NODE-based models is still unknown and can be large in human-aligned tasks which are more complex, and this may harm the performance of the proposed method. We believe that addressing current limitations could be interesting future directions.

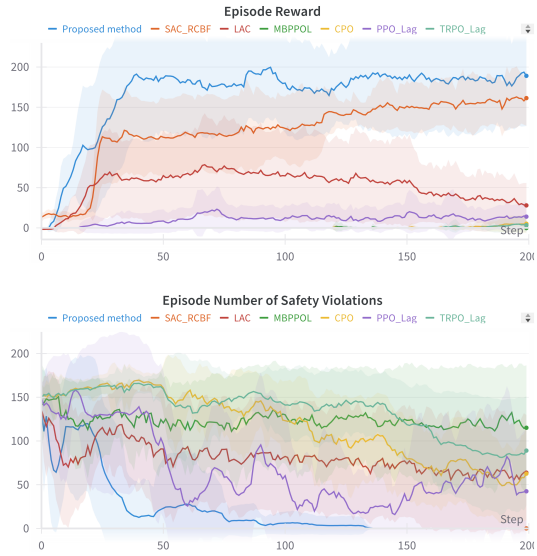

Fig. 1. The cumulative reward and cumulative number of safety violations of each episode in the simulated car following environment setting are compared between the proposed method (drawn in blue) and baselines. Each curve illustrates the average across ten experiments employing different random seeds, with the shaded area denoting the standard deviation. The safety violation of the SAC-RCBF algorithm keeps being 0 and therefore its graph is coincident with the X-axis.

REFERENCES

[1] Y. Wang and D. Boyle, "Trustworthy reinforcement learning for quadrotor uav tracking control systems," *arXiv preprint arXiv:2302.11694*, 2023.

[2] H. Cao, Y. Mao, L. Sha, and M. Caccamo, "Physical deep reinforcement learning towards safety guarantee," *arXiv preprint arXiv:2303.16860*, 2023.

[3] Y. Wang, J. O'Keeffe, Q. Qian, and D. Boyle, "Quadue-ccm: Interpretable distributional reinforcement learning using uncertain contraction metrics for precise quadrotor trajectory tracking," in *Conference on Robot Learning*. PMLR, 2023, pp. 2306–2316.

[4] S. He, Y. Wang, S. Han, S. Zou, and F. Miao, "A robust and constrained multi-agent reinforcement learning framework for electric vehicle amod systems," *arXiv preprint arXiv:2209.08230*, 2022.

[5] A. Zacharaki, I. Kostavelis, A. Gasteratos, and I. Dokas, "Safety bounds in human robot interaction: A survey," *Safety science*, vol. 127, p. 104667, 2020.

[6] Q. Liu, Z. Liu, B. Xiong, W. Xu, and Y. Liu, "Deep reinforcement learning-based safe interaction for industrial human-robot collaboration using intrinsic reward function," *Advanced Engineering Informatics*, vol. 49, p. 101360, 2021.

[7] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[8] H. Wang, K. Margellos, and A. Papachristodoulou, "Safety verification and controller synthesis for systems with input constraints," *arXiv preprint arXiv:2204.09386*, 2022.

[9] A. A. do Nascimento, A. Papachristodoulou, and K. Margellos, "A game theoretic approach for safe and distributed control of unmanned aerial vehicles," 2023.

[10] Y. Emam, P. Glotfelter, and M. Egerstedt, "Robust barrier functions for a fully autonomous, remotely accessible swarm-robotics testbed," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 3984–3990.

[11] M. Han, L. Zhang, J. Wang, and W. Pan, "Actor-critic reinforcement learning for control with stability guarantee," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6217–6224, 2020.

[12] H. Cao, Y. Mao, L. Sha, and M. Caccamo, "Physical deep reinforcement learning: Safety and unknown unknowns," *arXiv preprint arXiv:2305.16614*, 2023.

[13] K. Miao and K. Gatsis, "Towards optimal network depths: Control-inspired acceleration of training and inference in neural ODEs," in *The Symbiosis of Deep Learning and Differential Equations III*, 2023. [Online]. Available: https://openreview.net/forum?id=wErWEsPY8g

[14] H. Wang, Z. Xiong, L. Zhao, and A. Papachristodoulou, "Model-free verification for neural network controlled systems," *arXiv preprint arXiv:2312.08293*, 2023.

[15] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[16] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[17] Y. Xiong, D.-H. Zhai, M. Tavakoli, and Y. Xia, "Discrete-time control barrier function: High-order case and adaptive case," *IEEE Transactions on Cybernetics*, 2022.

[18] K. Miao and K. Gatsis, "Learning robust state observers using neural odes," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 208–219.

[19] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[20] L. Zhao, K. Gatsis, and A. Papachristodoulou, "A barrier-lyapunov actor-critic reinforcement learning approach for safe and stable control," *arXiv preprint arXiv:2304.04066*, 2023.

[21] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic Press, 1982.

[22] D. Yu, H. Ma, S. Li, and J. Chen, "Reachability constrained reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 25 636–25 655.

[23] H. Ma, Y. Guan, S. E. Li, X. Zhang, S. Zheng, and J. Chen, "Feasible actor-critic: Constrained reinforcement learning for ensuring statewise safety," *arXiv preprint arXiv:2105.10682*, 2021.

[24] X. Ma, "Traffic performance evaluation using statistical and machine learning methods," Ph.D. dissertation, The University of Arizona, 2022.

[25] X. Wang and Y. Jin, "Work process transfer reinforcement learning: Feature extraction and finetuning in ship collision avoidance," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 86212. American Society of Mechanical Engineers, 2022, p. V002T02A069.

[26] X. Ma, A. Karimpour, and Y.-J. Wu, "On-ramp and off-ramp traffic flows estimation based on a data-driven transfer learning framework," *arXiv preprint arXiv:2308.03538*, 2023.

[27] Y. Miao, I. Armeni, M. Pollefeys, and D. Barath, "Volumetric semantically consistent 3d panoptic mapping," *arXiv preprint arXiv:2309.14737*, 2023.

[28] X. Wang and Y. Jin, "Transfer reinforcement learning: Feature transferability in ship collision avoidance," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 87318. American Society of Mechanical Engineers, 2023, p. V03BT03A071.

[29] S. Su, Y. Li, S. He, S. Han, C. Feng, C. Ding, and F. Miao, "Uncertainty quantification of collaborative detection for self-driving," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5588–5594.

[30] Y. Lin, C. Li, M. Ding, M. Tomizuka, W. Zhan, and M. Althoff, "Drplanner: Diagnosis and repair of motion planners using large language models," *arXiv preprint arXiv:2403.07470*, 2024.

[31] Y. Emam, P. Glotfelter, Z. Kira, and M. Egerstedt, "Safe model-based reinforcement learning using robust control barrier functions," *arXiv preprint arXiv:2110.05415*, 2021.

[32] A. K. Jayant and S. Bhatnagar, "Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 432–24 445, 2022.

[33] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.

[34] A. Ray, J. Achiam, and D. Amodei, "Benchmarking Safe Exploration in Deep Reinforcement Learning," 2019.