Reinforcement Learning based Hierarchical Control for Path Tracking of a Wheeled Bipedal Robot with Sim-to-Real Framework

Wei Zhu, Fahad Raza, and Mitsuhiro Hayashibe

***Abstract*— We propose a reinforcement learning (RL) based hierarchical control framework for path tracking of a wheeled bipedal robot. The framework consists of three control levels. 1) The high-level RL is used to obtain an optimal policy through trial and error in a simulated environment. 2) The middle-level Lyapunov-based non-linear controller is utilized to track a desired line with strong robustness and high stability. 3) The low-level PID-based controller is implemented to simultaneously achieve both balancing and velocity tracking for a physical wheeled bipedal robot in real world. Thanks to the middle-level controller, the offline trained policy in simulation can be directly employed on the physical robot in real time without tuning any parameters. Moreover, the high-level policy network is able to improve optimality and generality for the task of path tracking, as well to avoid the cumbersome process of manually tuning control gains. The experiment results in both simulation and real world demonstrate that the proposed hierarchical control framework can achieve quick, robust, and stable path tracking for a wheeled bipedal robot.**

I. INTRODUCTION

1) Hierarchical Reinforcement Learning (RL): Complex control tasks that are difficult to achieve with only a single-layer RL framework can be divided into multiple simple sub-tasks. Multi-layer RL frameworks are proposed for co-operating to finish all sub-tasks [1]. In addition, the high-level controllers, such as those used to plan the velocities of the robot body [2], can be learned using RL, whereas the low-level non-learning controllers are used to convert the simplified policies into more complex commands, such as the torque or the position of all joint motors [3]. By combining RL frameworks and the existing non-learning controllers, the learning efficiency of RL can be considerably improved because RL algorithms are only used to obtain high-level policies that generally contain simple state and action spaces. In addition, such a combination can aid in transferring the policies learned in the simulation to the real world for two main reasons: the high-level RL methods simply utilize simple kinematics as a simulation engine rather than complex dynamics in some cases; and the low-level non-learning controllers are robust to uncertainties. Therefore, in this work, we utilize a hierarchical control framework combining RL and a non-learning controller for more stable and robust motion.

2) Path Tracking for a Wheeled Bipedal Robot: Most of researchers are focusing on balancing control for wheeled bipedal robot [4][5]. However, the work of path tracking is still limited. There are a tremendous number of works for path tracking of mobile robots. As a widely-used proportional-integral-derivative (PID) controller as well as its variants, it can be theoretically applied for any kinds of mobile robots[6][7]. It is however prohibitively difficult to manually tune well control gains. Another kind of approach is to design stable controllers using Lyapunov techniques [8] which can guarantee stability thus ensuring reliable asymptotic convergence. Similarly, tuning the parameters in Lyapunov-based controllers are time-consuming even dangerous for robot hardware with frequent exploratory manipulation. Thus, we aim at avoiding the adjustment in real world and turn it into simulation where a tremendous amount of training can be quickly and safely generated to update policy networks. Then the optimal parameters obtained in simulation are directly applied in various real-world scenarios without any re-training or re-adjusting. Even though some parameter tuning methods can optimize parameters, such as particle swarm optimization [9] and artificial neural networks [10], the supervised data are generally produced manually or based on system models. Comparatively, RL-based parameter optimization approaches are more automatic and model free because of the learning process of interacting with environments.

In this paper, we implement a reinforcement learning (RL) based hierarchical control framework for path tracking of a wheeled bipedal robot. The robot used in this work is shown

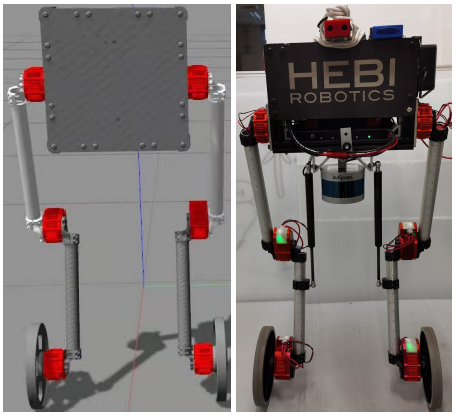

Fig. 1: Wheeled bipedal robot called IGOR. The left figure represents the simulated robot model in Gazebo while the right figure shows the physical robot.

This work was supported by the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research (B) under Grant 18H01399.

The authors are with the Department of Robotics, Graduate School of Engineering, Tohoku University, 980-8579, Sendai, Japan. zhu.wei.r5@dc.tohoku.ac.jp

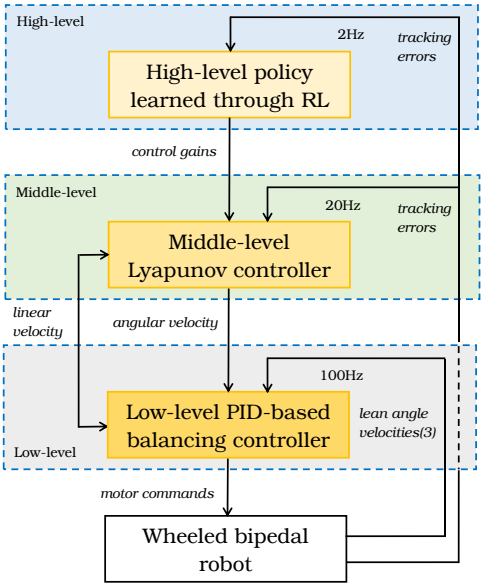

Fig. 2: Hierarchical control framework. The high-level RL is used to update a policy network with the frequency of 2Hz. The middle-level Lyapunov-based controller is utilized to eliminate tracking errors with 20Hz control frequency. The low-level PID-based controller is implemented at 100Hz so as to keep balance and track velocities quickly.

in Fig. 1, which is developed by HEBI Robotics [1]. Such kind of bipedal wheeled robots can be applied in complex and various scenarios because of the high motion flexibility, such as warehouses and outside fields. We first train the robot in simulation to obtain an optimal policy of path tracking, then directly transfer it to the real robot.

We list the main contributions of this work as follows:

- A hierarchical control framework is proposed for path tracking of a wheeled bipedal robot. The high-level RL controller can automatically obtain optimal gains for the middle-level Lyapunov-based control law.
- The sim-to-real transfer is achieved thanks to a Lyapunov-based controller which can prove global stability, thus to guarantee asymptotic convergence.
- The combined learning-based and traditional non-linear controllers improve the overall performance in terms of learning efficiency, optimality, generality, and stability.

II. HIERARCHICAL CONTROL FRAMEWORK

The hierarchical control framework is shown in Fig. 2, where the high-level policy is a neural network trained with the deep deterministic policy gradient (DDPG) algorithm, and the middle level is a Lyapunov-based non-linear controller for eliminating tracking errors and proving global stability, the low-level PID-based controller is used for simultaneously keeping balancing and tracking velocities. The high-level and the middle-level controllers share the same inputs — path tracking errors — with 2Hz and 20Hz respectively. The high-level controller outputs control gains which are the inputs for the middle-level Lyapunov controller. On the one hand, frequently changing control gains would make the system unstable, so we set a low control frequency in the high level. On the other hand, in order to make the path tracking more continuous and accurate while not consuming much of the computing resources, the operation frequency of the middle-level controller is set at 20Hz. According to the path tracking errors as well as the control gains and the constant desired linear velocity, the middle-level controller outputs the desired angular velocity which is the input for the low-level controller. We choose a constant linear velocity to keep the desired lean angle of IGOR as a constant, thereby making the motion more stable. The low-level PID-based controller with high frequency is implemented to track the linear and the angular velocities while keeping the robot lean angle stable. Even though we utilize the IGOR for validation, the proposed high and middle -level frameworks can be generalized to other mobile robots, such as quadruped robots and wheeled vehicles.

The high-level policy network is trained in simulation then directly deployed on the real IGOR robot without any fine tuning thanks to the introduction of the middle-level Lyapunov-based controller which can theoretically guarantee stability and robustness. Besides, since the initial policy is stable, a convergent network can be obtained quickly. In addition to the advantages brought from the middle level, the high-level policy network with one time training can be applied to various scenarios without any re-training due to the generalization ability of deep neural networks. Therefore, no matter what the robot states are within a specific scope, it can move to the desired path as soon as possible. For a wheeled bipedal robot, it's challenging to keep balancing while following a path if directly generating motor commands according to tracking errors. Consequently, we transfer the errors into the angular velocity of the robot through the middle level. Then, the low-level PID-based controller is utilized to track the angular velocity and a pre-defined constant linear velocity, as well as to keep balancing. In this way, the robot can simultaneously follow a path and keep balancing even being pushed with a random force. In sum, the proposed hierarchical control framework can achieve the task of path tracking for a wheeled bipedal robot with optimality, stability, robustness, and generality.

III. HIGH-LEVEL POLICY

The high-level policy network is trained only using the simulated IGOR robot as in Fig. 1, so the dangerous and time-consuming learning on the real robot can be avoided. Moreover, we implement the high-level RL-based controller because it can automatically obtain optimal gains for the middle-level traditional Lyapunov-based control law. Even though such Lyapunov-based controllers can guarantee stability, it is tremendously difficult to tune the gains involved in the controllers. In this work, the policy network is updated using the DDPG algorithm, which can be represented as

[1] https://www.hebirobotics.com/robotic-kits

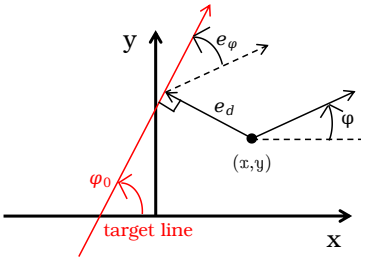

Fig. 3: Robot states on 2D plane. The robot is regarded as a particle with the position (x, y) and the orientation φ. The robot states for path tracking consist of the distance error e_d and the orientation error e_φ which are related to a target straight line with the orientation φ_0.

follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\boldsymbol{s}\sim\mathcal{D}}[\nabla_\theta \mu_\theta(\boldsymbol{s}) \nabla_{\boldsymbol{a}} Q^{\boldsymbol{\mu}}(\boldsymbol{s}, \boldsymbol{a}) \mid_{\boldsymbol{a}=\mu_\theta(\boldsymbol{s})}], \quad (1)$$

where $J(\theta) = \mathbb{E}_{\boldsymbol{s}\sim\rho^\mu}[R(\boldsymbol{s}, \boldsymbol{a})]$ is the objective with R meaning the immediate reward and ρ^μ denoting a discounted state visitation distribution for a policy μ, $\boldsymbol{s}$ represents the robot states, $\boldsymbol{a}$ stands for the control commands, μ_θ denotes the deterministic policy mapping $\boldsymbol{s}$ to $\boldsymbol{a}$ through neural networks with the weights θ, $\mathcal{D}$ is the replay buffer used to store R, $\boldsymbol{s}$ and $\boldsymbol{a}$, as well as to playback them randomly when updating the policy network.

A. Robot States

We assume that the robot can always keep balancing with the lean angle as a constant, therefore, its position and orientation can be simplified as $[x, y, \varphi]$ shown in Fig. 3, where x and y denote the 2D positions, φ stands for the orientation. $[x, y, \varphi]$ can be obtained through ROS-Gazebo APIs in simulation, while it is obtained via fusing the wheel odometer and the inertial measurement units (IMUs) embedded in the hip actuators using extended kalman filter (EKF) in real world. Additionally, the target path is assumed as a straight line for simplification and for that any other curves can be dispersed into sequential connected straight lines. Therefore, the robot states for path tracking can be represented by the distance error e_d and the orientation error e_φ. Moreover, considering the historic angular velocity of the robot may also influence the performance of path tracking, we add the angular velocity of the last time as one part of the robot states, which is defined as $\omega(t-1)$. Therefore, the robot states at the time t can be finally represented as follows:

$$\boldsymbol{s}_t = [e_d(t), e_\varphi(t), \omega(t-1)]. \quad (2)$$

B. Control Commands

A direct idea for path tracking of mobile robots is to plan linear and angular velocities according to tracking errors. However, instead of planning these two velocities in the high level, a more abstract method is proposed to represent the outputs of the policy network, namely control gains $\boldsymbol{\kappa}$. The reasons for this selection include two parts: one is that the stability and robustness can be further enhanced for path tracking; the other is that the training efficiency can be improved significantly because the initial samples are not completely random. In this case, the policy network can be considered as an adaptive regulator to generate optimal $\boldsymbol{\kappa}_t$ based on the robot states $\boldsymbol{s}_t$. According to the analysis in Section IV, the control commands $\boldsymbol{\kappa}$ are chosen as follows:

$$\boldsymbol{\kappa} = [k_1, k_2], \quad (3)$$

where k_1 and k_2 with positive values are the control gains output from the policy network in the high level, which are the inputs for the middle level.

C. Immediate Reward

The target of path tracking is to make e_d and e_φ convergent to zero. Moreover, for the IGOR robot, the lean angle β should be kept as a constant so as for more stable motion and satisfying the assumption of 2D path tracking. Therefore, the immediate reward R at the time t is defined as below:

$$R = -(\gamma_1|e_d(t)| + \gamma_2|e_\varphi(t)| + \gamma_3|\beta(t)|), \quad (4)$$

where γ_1, γ_2, and γ_3 weigh e_d, e_φ, and β respectively.

D. Network Architecture

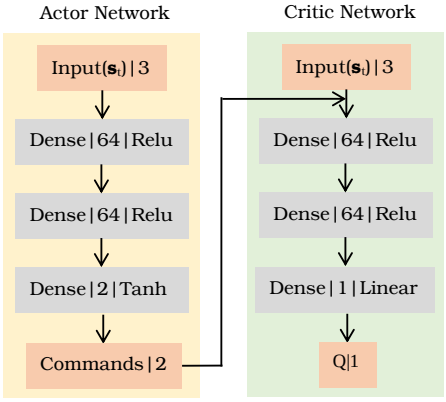

Fig. 4: Network architecture. Every hidden layer is represented by its type, dimension and activation mode. The input and the output layers are described by aforementioned elements as well as their dimensions.

Fig. 4 illustrates the network architecture of the DDPG algorithm, which is based on the actor-critic framework. The actor network, with the robot states $\boldsymbol{s}_t$ as the input, consists of three hidden layers and outputs the commands. For the critic network, the input is composed by the robot states $\boldsymbol{s}_t$ and the output from the actor network with three hidden layers. The Q-value, as the output of the critic network, is predicted through three hidden layers.

IV. MIDDLE-LEVEL LYAPUNOV-BASED CONTROLLER

One of the contributions of this work is that we combine a traditional Lyapunov-based controller and the RL algorithm so as to make the performance of RL more stable, thereby solving the problem that the policy learned in simulation can not be directly transferred to the real world with high success rate.

A. Design of Controller

For path tracking shown in Fig. 3, the widely-used approaches include PID [6], model predictive control (MPC) [11], etc.. They are either unable to theoretically prove global stability or consume a tremendous amount of computing resources thus deteriorating real-time performance. Therefore, we introduce a Lyapunov-based controller which can guarantee stability while ensuring real-time performance. The controller is given as follows [8]:

$$\omega = -k_1 v e_d \frac{sin(e_\varphi)}{e_\varphi} - k_2 |v| e_\varphi, \tag{5}$$

where $k_1 > 0$ and $k_2 > 0$ are control gains output from the high level, ω and v are angular and linear velocities respectively. In this work, v is chosen as a constant value for avoiding the translation acceleration which would tilt the robot, thereby satisfying the assumption of 2D motion.

B. Proof of Stability

The general kinematics of mobile robots moving on 2D plane is given as follows:

$$\begin{aligned} \dot{x} &= v\cos(\phi), \\ \dot{y} &= v\sin(\phi), \\ \dot{\phi} &= \omega, \end{aligned}$$

As presented in Fig. 3, via coordinate transformation, ϕ is equal to e_φ and y can be replaced by e_d. The Lyapunov function candidate is selected as below:

$$V(t) = \frac{1}{2} k_1 e_d^2 + \frac{1}{2} e_\varphi^2.$$

Differentiating V yields

$$\dot{V} = k_1 e_d \dot{e}_d + e_\varphi \dot{e}_\varphi = k_1 e_d v \sin e_\varphi + e_\varphi \omega.$$

Substituting the controller (5) into the above equation yields

$$\dot{V} = -k_2 |v| e_\varphi^2 \leq 0,$$

implying that the controller guarantees global asymptotic convergence of e_d and e_φ.

V. LOW-LEVEL PID-BASED BALANCING CONTROLLER

Most of the previous research works focus on the position and orientation control for two-wheeled inverted mobile robots while keeping balancing. In our work, however, the robot is required to simultaneously keep balancing and track desired angular and linear velocities so as to stably follow a straight line. Given a constant linear velocity and a dynamic angular velocity output from the middle level, the low-level controller needs to track them while keeping the lean angle of the IGOR robot as a constant.

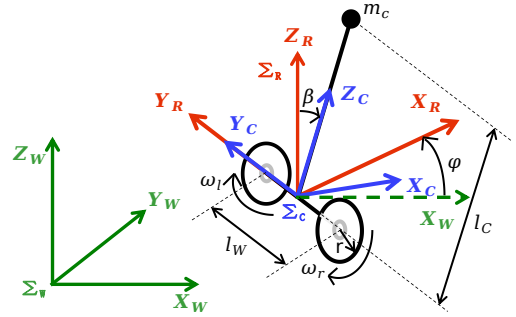

Fig. 5: Model of IGOR robot. Σ_W, Σ_R, and Σ_C represent the world frame, robot frame, and CoM frame respectively. When doing the task of path following, the robot states $\boldsymbol{s}_t$ are relative to Σ_W without Z_W. If simultaneously tracking desired velocities and keeping balancing, the lean angle β and the translation velocity of CoM are relative to Σ_R.

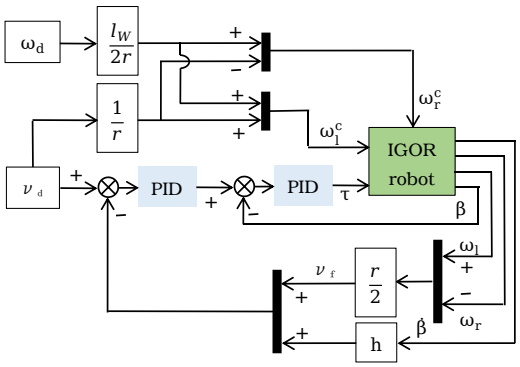

Fig. 6: Framework of the PID-based controller. The angular and linear velocity tracking is achieved through a feedforward controller while the balancing is done via two coupled PID-based feedback controllers.

A. Model of IGOR Robot

Fig. 5 describes the model of IGOR robot. In this work, the hip and the knee joints are fixed for simplification, therefore, the distance l_C from the center of mass (CoM) to the center of wheel axis is a constant value. The lean angle of the robot is defined as β, the height of CoM is thus given as below:

$$h = l_C cos(\beta).$$

The rotation velocities of two wheels are ω_l and ω_r respectively. l_W represents the distance between two wheels and the radius of wheel is r. Therefore, ω and v can be expressed as follows:

$$\begin{aligned} \omega &= -r(\omega_r + \omega_l)/l_W, \\ v &= r(\omega_l - \omega_r)/2. \end{aligned} \tag{6}$$

B. PID-based Controller

The framework of the PID-based Controller is illustrated in Fig. 6, where the angular and linear velocity tracking is achieved by a feedforward controller while the balance keeping includes two PID-based feedback controllers. According to the velocity model (6), the commands of wheel rotation speed ω_l^c and ω_r^c can be calculated, which are then tracked through the speed control mode of wheel motors. It's however complex for the control of balancing because the linear velocity and the lean angle are highly coupled. The feedback of the linear velocity consists of two parts: one is calculated by the wheel rotation speeds (ω_l and ω_r) while the other is computed via the velocity of the lean angle ($\dot{\beta}$). The outer-loop PID controller is used to generate the desired lean angle for the inner-loop PID controller, which outputs the commands of torque (τ) for wheel motors.

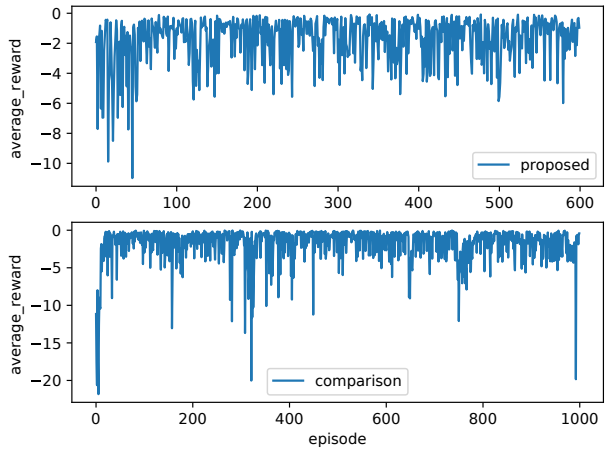

Fig. 7: Average episode reward for two different policy networks. The upper figure shows the reward of the proposed hierarchical control framework while the bottom one demonstrates the result of the comparison policy without the middle level. The oscillation is more distinct in the bottom figure as the vertical axis scale is much smaller than that of the upper.

VI. EXPERIMENTS

One of the contributions in this work is a proposed method that enables the simulated policy to be deployed on the physical robot without any fine tuning. Through the comparison experiments, we demonstrate the middle-level Lyapunov-based controller plays a significant role for successful sim-to-real transfer and fast convergence of RL training. The simulation and real experiments can be referred to the attached video [2].

A. Simulation

We trained two policy networks, which have same inputs but different outputs for comparison. The initial robot states are randomly set for each episode, with the scope $-1 < e_d < 1$ m, $-\pi < e_\varphi < \pi$ rad, $\omega(0) = 0$. For the comparison policy, the output is angular velocity with high frequency (20Hz). For the proposed framework, the outputs are control gains with low speed (2Hz), which are further processed through the middle-level controller and then transferred to the angular velocity. Based on the updating frequency of these two policy networks, the numbers of steps in each training episode are chosen as 1000 and 100 respectively. The learning results are shown in Fig. 7 which describes the average episode reward during training. The initial average reward of the proposed policy (about -10) is two times as high as that of the comparison policy (about -20), even though both of these two policies have same average reward (about -0.2) in the end of training. Therefore, the required number of training steps for the convergence of network is much fewer when using the proposed learning framework compared with the reference framework. Although we train the robot in simulation, such a policy network can be also updated using the physical robot because the proposed RL framework behaves well in the beginning of training and consumes less computing resource.

After training, we tested these two policy networks, with the results of path tracking showing in TABLE I, comprehensively analyzing the performance from ten tests. For the proposed and the comparison policy networks in each test, the robot starts from the same states $e_\varphi(0)$ and $e_d(0)$. t_a^1 and t_a^2 are the convergence time corresponding to the proposed and the comparison methods respectively. The steady-state errors corresponding to t_a^1 and t_a^2 are $[e_\varphi^1(t_a^1), e_d^1(t_a^1)]$ and $[e_\varphi^2(t_a^2), e_d^2(t_a^2)]$. Through comparing the average convergence time, it's demonstrated that the proposed method can achieve faster convergence of path tracking. Moreover, the proposed method can guarantee smaller steady-state orientation error. When comparing the standard deviation (STD), our method can achieve more stable performance in terms of all indexes. Even though there is no significant difference in path tracking between these two methods in simulation, the proposed framework possesses outstanding advantages in real world as demonstrated in the next section.

B. Real Experiments

Thanks to the middle-level Lyapunov-based controller, the simulated policy can be implemented on the physical robot while the comparison method without this controller fails. Moreover, we fine-tuned the parameters of the simulated robot according to those of the real robot, such as the mass of links and actuators, torque limitations. Besides, we utilized the technique of domain randomization in simulation to enhance the sim-to-real reliability. Fig. 8 shows the results of one test, with the left three figures describing the performance of the proposed method and the right illustrating the comparison method's. Obviously, our method with the middle-level controller can stably and robustly achieve path tracking while the reference framework is unable to accurately follow the path and makes the robot vibrate violently.

The previous experiments are all about following a straight line, we therefore extend our method to follow more complex curves in order to further validate the generality of

[2]https://youtu.be/B01vbc-Lx1Q

TABLE I: COMPREHENSIVE ANALYSIS FOR THE PERFORMANCE OF TEN TESTS.

test	1	2	3	4	5	6	7	8	9	10	average	STD
$e_\varphi(0)$	0.777	-1.968	-2.364	0.525	2.487	0.082	0.219	1.370	3.000	-1.730	—	—
$e_d(0)$	-0.796	-0.325	0.119	0.327	0.151	0.711	-0.813	0.026	0.231	-0.736	—	—
t_a^1	22.894	30.321	27.306	16.057	29.980	19.876	24.010	17.667	33.698	33.772	**25.558**	**6.110**
t_a^2	32.086	39.300	27.297	10.889	27.809	18.103	32.155	12.376	33.612	45.099	27.873	10.586
$e_\varphi^1(t_a^1)$	0.071	0.072	0.070	-0.100	-0.097	-0.098	0.070	-0.097	-0.100	0.071	**0.085**	**0.014**
$e_\varphi^2(t_a^2)$	0.082	0.085	0.087	-0.409	-0.401	-0.399	0.085	-0.400	-0.403	0.085	0.244	0.159
$e_d^1(t_a^1)$	-0.050	-0.050	-0.050	0.050	0.050	0.050	-0.050	0.050	0.050	-0.050	**0.050**	**0.000**
$e_d^2(t_a^2)$	-0.012	-0.012	-0.013	0.049	0.049	0.049	-0.011	0.049	0.050	-0.012	0.031	0.019

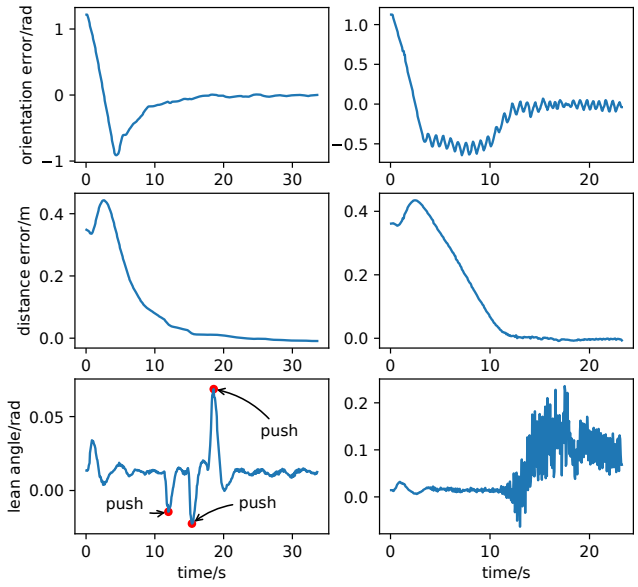

Fig. 8: Results with real robot test. The left three figures show the results of the proposed method while the right display the comparison method's case without the middle-level controller. The upper figure is orientation error, the middle represents the distance error, and the bottom stands for the lean angle. For the proposed method, we pushed the robot three times as presented in the left bottom figure, where the positive peak angle means we push the robot backward while the other two negative represent the forward pushing.

the proposed control framework. Considering a complicated curve can be discretized as a series of straight lines, we chose a circle with the radius 1m and then sliced it into 32 line segments. The tracking results in real world are illustrated as in Fig. 9, which testifies the effectiveness of the proposed method even when it is applied for tracking a complex curve. Due to the frequently changing situation for the direction of the line segments, the steady-state error is unavoidable as presented in Fig. 9, which would be solved in our future work. This level of tracking accuracy is acceptable considering the precision required for mobility auto-navigation purpose.

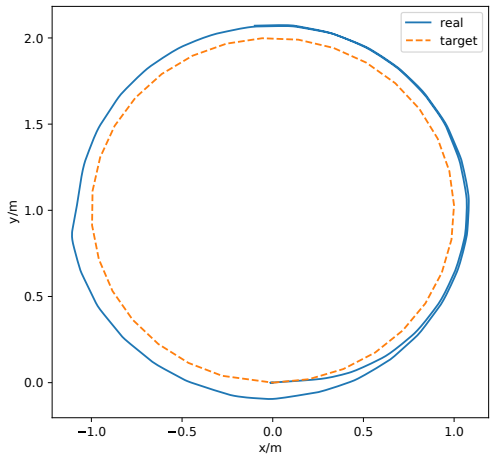

Fig. 9: Results of circle path tracking of the real robot. The dotted curve is the target circle while the solid one indicates the actual trajectory of the robot.

VII. CONCLUSIONS

We proposed a reinforcement learning-based hierarchical control for path tracking of a wheeled bipedal robot, wherein the high-level policy network is used to generate optimal gains for the middle-level Lyapunov-based non-linear controller, while the low-level PID-based controller receives the velocity signals from the middle level and then output motor commands for speed tracking and balance keeping. For a wheeled bipedal robot auto-navigation, we face a multi tasking problem for speed tracking and balance keeping for unstable system. This paper demonstrated that a reinforcement learning-based hierarchical control by employing the middle-level non-linear controller is useful for managing multi-tasking function. The high-level RL controller avoids the cumbersome process of manually tuning control gains for Lyapunov-based controllers. Moreover, thanks to the stability and robustness of the middle-level controller, the simulated policy can be deployed on the physical robot without fine tuning for realizing sim-to-real transfer. The sufficient experiment results both for simulation and for real robot demonstrated the optimality, stability, and generality of the proposed control framework. This work can be one contribution toward auto-navigation for wheeled bipedal systems in real world, supported by deep learning framework.

For next step, there are several opportunities for future work including eliminating steady-state errors through introducing integral factor (this work only considers the proportional factor), fusing the data from the Lidar sensor installed under the robot body for more accurate localization, navigating and exploring under unknown environments.

REFERENCES

[1] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[2] K. Tsunekawa, F. Leiva and J. Ruiz, "Visual navigation for biped humanoid robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3247-3254, 2018.

[3] X. Zhang, X. Guo, Y. Fang and W. Zhu, "Reinforcement learning-based hierarchical control for path following of a salamander-like robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6077-6083, 2020.

[4] R. Fahad, and M. Hayashibe. "Towards robust wheel-legged biped robot system: Combining feedforward and feedback control," in *IEEE/SICE International Symposium on System Integration (SII)*, 2021.

[5] R. Fahad, W. Zhu, and M. Hayashibe. "Balance stability augmentation for wheel-legged biped robot through arm acceleration control," *IEEE Access*, vol. 9, pp. 54022-54031, 2021.

[6] A. Hernandez, C. Copot, R. De Keyser, T. Vlas and I. Nascu, "Identification and path following control of an AR.Drone quadrotor," in *International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 583-588, 2013.

[7] J. Rico, I. Alcala, J. Ortega, and E. Camacho, "Mobile robot path tracking using a robust PID controller," *Control Engineering Practice*, vol. 9, no. 11, pp. 1209-1214, 2001.

[8] A. Morro, A. Sgorbissa and R. Zaccaria, "Path following for unicycle robots with an arbitrary path curvature," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 1016-1023, 2011.

[9] T. Kim, I. Maruta, and T. Sugie, "Robust PID controller tuning based on the constrained particle swarm optimization," *Automatica*, vol. 44, no. 4, pp. 1104-1110, 2008.

[10] G. Han, W. Fu, W. Wang, and Z. Wu, "The lateral tracking control for the intelligent vehicle based on adaptive PID neural network," *Sensors*, vol. 17, no. 6, pp. 1244-1258, 2017.

[11] G. Klancar, and I. Skrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robotics and autonomous systems*, vol. 55, no.6, pp. 460-469, 2007.