Hierarchical Control for Bipedal Locomotion using Central Pattern Generators and Neural Networks

Sayantan Auddy
Intelligent and Interactive Systems
University of Innsbruck
Innsbruck, Austria
sayantan.auddy@uibk.ac.at

Sven Magg
Knowledge Technology
University of Hamburg
Hamburg, Germany
magg@informatik.uni-hamburg.de

Stefan Wermter
Knowledge Technology
University of Hamburg
Hamburg, Germany
wermter@informatik.uni-hamburg.de

Abstract—The complexity of bipedal locomotion may be attributed to the difficulty in synchronizing joint movements while at the same time achieving high-level objectives such as walking in a particular direction. Artificial central pattern generators (CPGs) can produce synchronized joint movements and have been used in the past for bipedal locomotion. However, most existing CPG-based approaches do not address the problem of high-level control explicitly. We propose a novel hierarchical control mechanism for bipedal locomotion where an optimized CPG network is used for joint control and a neural network acts as a high-level controller for modulating the CPG network. By separating motion generation from motion modulation, the high-level controller does not need to control individual joints directly but instead can develop to achieve a higher goal using a low-dimensional control signal. The feasibility of the hierarchical controller is demonstrated through simulation experiments using the Neuro-Inspired Companion (NICO) robot. Experimental results demonstrate the controller's ability to function even without the availability of an exact robot model.

Index Terms—Gait development, hierarchical neural architecture, reinforcement learning

I. INTRODUCTION

In nature, central pattern generators (CPGs) exist as neural circuits in the spinal cord of vertebrates and have been found to be responsible for the rhythmic movements of animals [1]–[5]. Artificial CPGs mimic the behavior of natural CPGs and have a number of properties suitable for robot locomotion. CPGs can produce rhythmic output autonomously, they exhibit stable limit cycle behavior, and they can achieve mutual entrainment with the natural dynamics of a robot [6].

CPGs such as the Matsuoka oscillator [7], [8] have been used in the past for bipedal locomotion [9]–[12] by employing a network of interconnected CPGs, each of which controls an individual robot joint. Feedback pathways are incorporated to allow for some degree of adaptation and the CPG parameters are tuned by an optimization algorithm. The fitness criterion of the optimization process is designed to yield basic gait characteristics such as stability or speed. The adaptive capability of such an approach is restricted because the feedback mechanisms, which are often implemented using linear coupling terms in the CPG differential equations, have a limited capability of reacting to environmental changes.

In order to improve the adaptability and performance of a CPG-based bipedal controller, we propose a novel hierarchical architecture, in which a trainable high-level controller is added on top of a CPG network. The lower level of the controller, consisting of a CPG network (based on [11]) and its feedback pathways, is used for generating target angles for the robot joints. A generalized Matsuoka oscillator [13] is used for modeling the CPGs and the CPG network parameters are found by an optimization algorithm. The high-level controller is implemented as a feedforward neural network trained through deep reinforcement learning, whose outputs are used to regulate selected CPG network parameters. This enables the high-level controller to modulate the CPG network's behavior using a low-dimensional feedback signal without needing to explicitly control every single joint. On purpose, we used a robot model with minor asymmetries between the left and right sides. Together with slippage on the ground, this causes the robot to deviate laterally when only the CPG network is used for walking. As an example of simple high-level control, we demonstrate how the neural network can learn to adjust to the robot model's inaccuracies and make it walk straight.

II. RELATED WORK

Methods based on zero moment point (ZMP) [14]–[17] have been a common choice for bipedal locomotion in robots. A criticism of such methods is that the reference trajectories generated using simplified models constrain the robot's movements, resulting in an unnatural gait with bent knees [18] and high energy consumption [19]. McGeer's [20] passive walking machines and Collins et al.'s [19] minimally actuated robots showed that if the body's natural dynamics is utilized, bipedal locomotion is possible even without complicated approaches.

CPG-based bipedal methods seek to utilize the ability of CPGs to entrain with the body dynamics of the robot. Taga et al. [21] used a network of coupled Matsuoka oscillators to control a 5-link planar biped in simulation. They showed that stable and flexible locomotion could be generated by entrainment between the rhythmic activities of the coupled CPGs and the movements of the mechanical structure. Ishiguro et al. [9] used a network of Matsuoka oscillators to control a 3D bipedal robot in simulation. An artificial neuromodulation mechanism was used for modulating the CPGs and the parameters were set
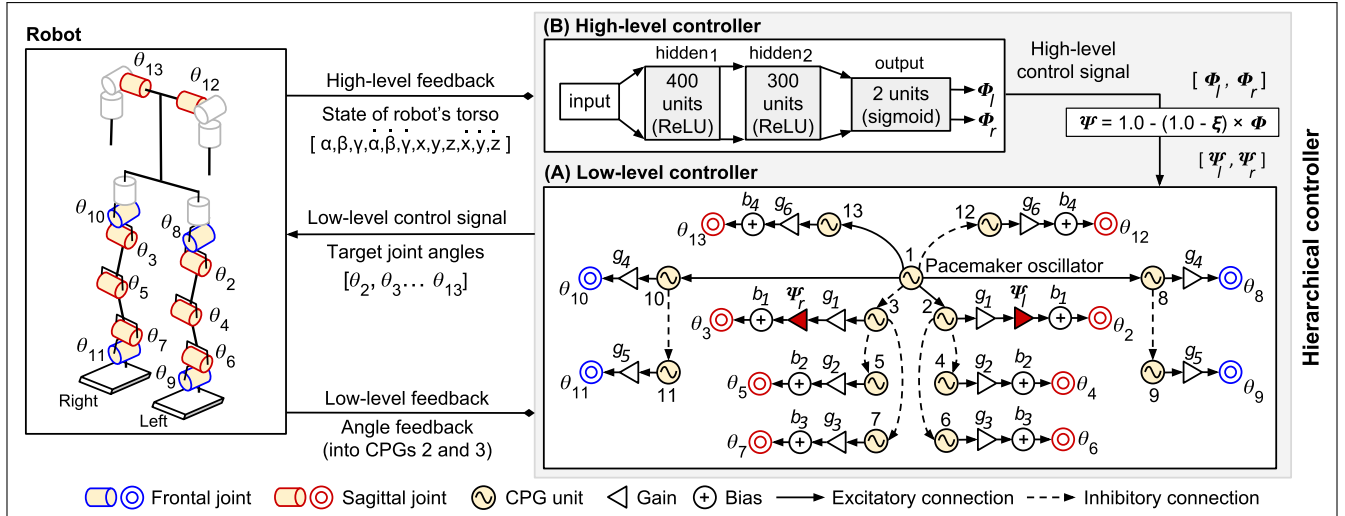
Fig. 1. Detailed architecture. Left: Robot joints are controlled by the CPG network. Joint θ_i is controlled by CPG i. Right: (A) Low-level controller CPG network with 13 CPGs and their gains and biases (modified from [11]). Apart from CPG-1 (pacemaker oscillator), each CPG controls a single joint. (B) High-level controller neural network, whose outputs (Φ_l, Φ_r) are converted to (Ψ_l, Ψ_r) for modulating the CPG network.

by a genetic algorithm. Endo et al. [22] used a CPG-based controller to achieve bipedal locomotion in a physical humanoid robot. The walking motion was broken up into a stepping-in-place motion and a propulsive motion. The feedback pathways for the propulsive motion were learned using a policy-gradient based method. The developed feedback controller showed stable walking behavior in a physical environment. Cristiano et al. [11] implemented a CPG network, composed of Matsuoka oscillators, for controlling the Nao robot's walking behavior in simulation as well as in the real world. One of the CPGs, designated as the pacemaker oscillator was used for generating the master signal for driving the CPG network. CPG network parameters were optimized by a genetic algorithm.

Most of these CPG-based methods use some form of feedback control, but the feedback parameters are fixed after optimization. Introducing a trainable high-level controller, which can modulate the CPG network, will lead to greater generalization abilities and a more effective form of control, which motivates the approach followed in this paper.

III. ARCHITECTURE

The architecture of the hierarchical controller (Fig. 1) depicts three distinct components: the robot, the low-level controller (CPG network) and the high-level controller (neural network). A simulated NICO humanoid robot [23] has been used to develop our bipedal locomotion approach. The NICO is 101cm tall, weighs 7kg and has 30 DoF (degrees-of-freedom) in total. Each leg has 3 DoF in the hip, 1 DoF in the knee and 2 DoF in the ankle. The proposed controller controls 10 sagittal and frontal joints in the legs and 2 sagittal joints in the shoulders (since arm swing increases walking stability).

A. Low-level Controller

For the low-level controller, we adapted the CPG network from [11]. The CPG network consists of 13 generalized

Matsuoka oscillators [13], interconnected using either an excitatory connection (weight = +1) causing the connected CPGs to oscillate in-phase, or an inhibitory connection (weight = −1) causing anti-phase oscillations. One pacemaker oscillator is responsible for driving the other CPGs and maintaining the phase relationships between them. The other 12 CPGs control separate robot joints. Empirically chosen, identical parameters are used for all the CPGs. Each CPG output is varied by multiplication with a gain term (for changing the amplitude) and addition with a bias term (for changing the mean position). Gains and biases are used symmetrically across the left and right sides of the CPG network, to produce symmetrical movement. Biases are omitted for frontal joints since their mean position should be zero. The configuration of CPG connections (shown in Fig. 1) results in limb movements similar to that in humans, e.g. contralateral sagittal hip joints move in anti-phase but each sagittal hip joint moves in-phase with the contralateral sagittal shoulder joint.

We modified the CPG network of [11] by using a different feedback mechanism. Instead of the phase reset feedback, wherein the pacemaker oscillator's phase is reset based on the pattern of foot-ground contact, we integrated the feedback mechanism from [9] where the actual angular position of each sagittal hip joint is multiplied by a weight k and then fed back into the corresponding CPG. In experiments in which we tested each feedback mechanism for 300 trials each (Sec. 5.2 in [24]), we found that using the angular feedback mechanism resulted in a better gait in terms of distance and stability.

To enable the robot to turn, we introduced two additional gains Ψ_l and Ψ_r (red triangles in Fig. 1), for the left and right sagittal hip CPGs respectively. By using unequal values for Ψ_l and Ψ_r, the amplitude of the two sagittal hip joints can differ, causing unequal stride lengths resulting in a small turn. The values of Ψ_l and Ψ_r are set by the high-level controller.

Each CPG unit is a Matsuoka oscillator consisting of an

extensor and a flexor neuron whose behaviors are governed by (1) and (2), respectively (generalized Matsuoka equations [11], [13]). Subscript i denotes a CPG unit, and superscripts e and f denote the extensor and flexor neurons respectively.

$$\tau_0\kappa\dot{u}_i^e = -u_i^e - w_0y_i^f - \beta v_i^e + u_t + f_i^e + s_i^e$$
$$\tau_0'\kappa\dot{v}_i^e = -v_i^e + y_i^e \tag{1}$$
$$\text{where } y_i^e = max(0, u_i^e) \text{ and } i = 1, ..., num$$

$$\tau_0\kappa\dot{u}_i^f = -u_i^f - w_0y_i^e - \beta v_i^f + u_t + f_i^f + s_i^f$$
$$\tau_0'\kappa\dot{v}_i^f = -v_i^f + y_i^f \tag{2}$$
$$\text{where } y_i^f = max(0, u_i^f) \text{ and } i = 1, ..., num$$

State variables u_i^e, u_i^f control the discharge rate and v_i^e, v_i^f control the self-inhibition of the extensor and flexor neurons respectively. τ_0 and τ_0' are the time constants for the rate of discharge and adaptation, respectively. κ is a parameter which modulates the frequency of the CPG. y_i^e and y_i^f are the activations of the extensor and flexor neurons. β and w_0 are the constants of self- and mutual-inhibition respectively. u_t is the tonic input and num is the number of CPG units in the network. f_i^e and f_i^f are feedback terms, which are non-zero only for the sagittal hip oscillators, for which $f_i^e = k\theta_i'$ and $f_i^f = -k\theta_i'$, where θ_i' is the actual angle of the joint controlled by CPG i and k is the feedback weight. $s_i^e = w_{ij}u_j^e$, $s_i^f = w_{ij}u_j^f$ represent the interaction between connected CPGs i and j (w_{ij} is the connection weight). Output o_i of CPG i is obtained by $o_i = -y_i^e + y_i^f$, which is multiplied by a gain and added with a bias (for sagittal CPGs), as shown in Fig. 1.

B. High-level Controller

The CPG network is designed to produce symmetrical joint movement across the left and right sides of the robot. However, the presence of structural inconsistencies in the legs, and slippage with the ground cause the robot to deviate from a straight trajectory when the low-level controller is used in isolation. In this paper, we use the high-level controller to rectify the turning behavior, but its function can be easily extended to other high-level objectives as well. Any lateral deviation is minimized by adjusting the stride lengths of the left and right feet (by varying the gain parameters Ψ_l, Ψ_r). The target angles θ_2 and θ_3 for the left and right sagittal hip joints are given by (3), where o_2 and o_3 are the outputs of the corresponding CPGs, and g_1 and b_1 are the gain and bias applied to the CPG outputs.

$$\theta_2 = o_2\Psi_lg_1 + b_1$$
$$\theta_3 = o_3\Psi_rg_1 + b_1 \tag{3}$$

The high-level controller is implemented as a fully connected, feedforward neural network with two hidden layers containing 400 and 300 ReLU units respectively (the structure of the hidden layers is based on the actor network used in [25]). The output layer consists of 2 sigmoid units. The input to the network is the vector $[\alpha, \beta, \gamma, \dot{\alpha}, \dot{\beta}, \dot{\gamma}, x, y, z, \dot{x}, \dot{y}, \dot{z}]$, consisting of the angular position (α, β, γ) and velocity $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$, and the Cartesian position (x, y, z) and velocity $(\dot{x}, \dot{y}, \dot{z})$ of

the robot's torso in the three dimensions. The outputs of the network (Φ_l, Φ_r) are used to derive the gains (Ψ_l, Ψ_r), according to (4), where $\xi \in [0.0, 1.0]$ is a parameter which controls how much influence the high-level controller can exert over the low-level controller.

$$\Psi_l = 1.0 - (1.0 - \xi)\Phi_l$$
$$\Psi_r = 1.0 - (1.0 - \xi)\Phi_r \tag{4}$$

When $\xi = 1.0$, Ψ_l and Ψ_r evaluate to 1.0, irrespective of the neural network's output (Φ_l and Φ_r). In this case, the high-level controller has no influence over the CPG network's behavior because the angles for the sagittal hip joints (3), computed by the CPG network, remain unaffected. When $\xi = 0.0$, Ψ_l and Ψ_r will be fully dependent on Φ_l and Φ_r respectively, and hence, the high-level controller can influence the CPG network to a great extent.

The inverse relationship between (Ψ_l, Ψ_r) and (Φ_l, Φ_r) in (4) is necessary because the initialization of the final layer weights and biases of the neural network (described in Section IV-B) is such that the network outputs are near zero in the initial stages of training. If Φ_l and Φ_r (both having near-zero values) are used in place of Ψ_l and Ψ_r in (3), the angles θ_2 and θ_3 would not show any oscillatory behavior about the bias position. Hence the robot would not be exhibiting any forward motion and training the neural network would not be possible.

Using this setup, the low-level controller can be optimized to produce a basic stable gait without bothering about the lateral deviation. The high-level controller can then be trained to minimize the lateral deviation using the parameters Ψ_l and Ψ_r, without having to deal with the 12 joints individually. Controlling 12 joints would have required 12 outputs from the high-level neural network controller which would have significantly increased the complexity of the training process.

IV. EXPERIMENTS AND RESULTS

The hierarchical controller was constructed in two phases. In the first phase, the neural network was omitted, and the CPG network was optimized by a genetic algorithm (GA). In the second phase, the neural network was added on top of the optimized CPG network and trained using the deep deterministic policy gradient (DDPG) algorithm [25]. After training, the entire controller was tested. All experiments were conducted using an Intel Core i5-6500 CPU and 16GB RAM. Simulations were carried out using the V-REP simulator.

A. Low-level Controller Optimization

Instead of optimizing the internal parameters (described in section III-A) of all the CPGs, first, the pacemaker oscillator's parameters were set empirically and replicated in the other CPGs. Then the gains, biases, the frequency controlling parameter κ and the low-level feedback weight k were optimized. A real-valued chromosome with 12 elements was used: $[\kappa, g_1, g_2, g_3, g_4, g_5, g_6, b_1, b_2, b_3, b_4, k]$. The gains $g_1, .., g_6$ and biases $b_1, .., b_4$ were used symmetrically across the left and right sides of the CPG network as shown in Fig.

1. This significantly reduces the optimization search space but does not allow the CPG network to address asymmetries through errors in the robot model, which are instead handled by the high-level controller. To bound the search space, the limits for the chromosome values were set as: $\kappa \in [0.2, 1.0]$, $g_1, .., g_6 \in [0.01, 1.0]$, $b_1 \in [-0.06, 0.0]$, $b_2 \in [0.0, 0.5]$, $b_3 \in [-0.5, 0.0]$, $b_4 \in [0.0, 1.0]$, $k \in [-2.5, 2.5]$. The bias limits were set so that the mean position of the sagittal joints resulted in a stable pose. The other limits were set empirically.

The internal CPG parameters were set according to [11] ($\tau_0 = 0.28$, $\tau_0' = 0.4977$, $\beta = 2.5$, $w_0 = 2.2829$, $u_t = 0.4111$). A GA with a population size of 200 was run for 30 generations. We used tournament selection with a tournament size of 3, and a 2-point crossover with a probability of 80% with random crossing points. Mutation to a gene value was performed by adding a small number, drawn randomly from a Gaussian distribution (mean=0.0, variance=10^{-4}) with probability 10% for a chromosome and 5% for a gene. For evaluating fitness, the CPG network was initialized with the values in the chromosome and the robot was allowed to walk for a maximum of 20 seconds or until it fell. Afterwards, the forward distance (d_x) and the time for which the robot was upright (t_{up}) were used to calculate the fitness using (5).

$$fitness = d_x + 0.5 \times t_{up} \tag{5}$$

We used a simple fitness function with no penalty for lateral deviation to simplify the optimization process. The genetic algorithm was executed multiple times, with similar results. The best solution obtained for a stable walk was [$\kappa = 0.3178$, $g_1 = 0.3777$, $g_2 = 0.0234$, $g_3 = 0.0132$, $g_4 = 0.4567$, $g_5 = 0.2019$, $g_6 = 0.3309$, $b_1 = $ -0.0519, $b_2 = 0.0963$, $b_3 = $ -0.1156, $b_4 = 0.4814$, $k = 1.5364$].

B. High-level Controller Training and Testing

The neural network (high-level controller) was trained based on the DDPG algorithm [25] which is an off-policy, model-free, actor-critic reinforcement learning algorithm, capable of handling continuous state and action spaces. In the reinforcement learning setup, the high-level controller was the actor, responsible for implementing the policy function. At each timestep t, the actor network took as input a state $s_t \in \mathbb{R}^{12}$ (high-level feedback $[\alpha, \beta, \gamma, \dot{\alpha}, \dot{\beta}, \dot{\gamma}, x, y, z, \dot{x}, \dot{y}, \dot{z}]$ in Fig. 1) and produced an action $a_t \in \mathbb{R}^2$ (high-level control signal $[\Phi_l, \Phi_r]$). As a consequence, a reward $r_t \in \mathbb{R}$ and a new state s_{t+1} were generated. The transition (s_t, a_t, r_t, s_{t+1}) was stored in a first-in-first-out cache called the *replay buffer*. A separate critic network, shown in Fig. 2, estimated the action-values (Q-values). DDPG makes use of *target actor* and *critic networks* for improving the stability of the learning process. Minibatches from the replay buffer and the outputs of the target networks were used for updating the critic network. The outputs of the critic network were then used for sampling the *policy gradient*, which was used for updating the weights and biases of the actor network.

During training, the high-level neural network modulated the optimized CPG network to make the robot walk. In each

training episode, the robot started at the origin of the world frame, facing in the x-direction. Each episode lasted for 40 seconds or until the robot fell down. After every second during an episode, the reward r_t was calculated using (6).

$$r_t = \begin{cases} -100 & \text{if robot fell} \\ -\zeta_{dev} \cdot |d_y| + \zeta_{dist} \cdot d_x - \zeta_{\gamma} \cdot |\gamma| & \text{otherwise} \end{cases} \tag{6}$$

The reward function was designed to favor a long forward distance (d_x) and to penalize any lateral deviation (d_y) or change in torso orientation (for the ideal straight walk, the torso orientation γ about the world z-axis would always be 0 radians). The terms $\zeta_{dev}, \zeta_{dist}$ and ζ_{γ} were weights assigned to the deviation, distance and orientation components, respectively. Together with ξ, the factor controlling the influence of the high-level controller (used in (4)), $\zeta_{dev}, \zeta_{dist}$ and ζ_{γ} formed the 4 hyperparameters specific to the high-level controller. Different values for these hyperparameters were set to create the setups shown in Table I.

TABLE I
HYPERPARAMETERS FOR TRAINING THE HIGH-LEVEL CONTROLLER

Setup	ζ_{dev}	ζ_{dist}	ζ_{γ}	ξ
$S1$	1.0	0.5	1.0	0.1
$S2$	1.0	0.5	1.0	0.4
$S3$	1.0	0.3	1.0	0.1
$S4$	1.0	0.3	1.0	0.4

We used sigmoid and linear activations in the output layer of the actor and critic networks, respectively, and used a replay buffer size of 10^5. All other DDPG-specific hyperparameters were set as per [25]. The final layer weights and biases of the actor and critic networks were initialized from a uniform distribution $[-3 \times 10^{-3}, 3 \times 10^{-3}]$, so that the initial outputs of both networks would be close to zero. The weights and biases of all the other layers were initialized from the uniform distribution $[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}]$, where f was the fan-in of the layer.

For each setup in Table I, the high-level controller was trained for 1000 episodes. After every 10 episodes, the performance of the networks was evaluated. During these test episodes, the exploratory noise used in DDPG was switched off and network updates were not made. The reward, distance, deviation, and torso orientation were recorded. These results are shown in Fig. 3. It can be seen that by the 1000th episode the high-level controller for all the setups learned to maximize the distance and minimize the deviation and change in orientation. Towards the end, the robot was also more stable, since large negative rewards rarely occurred. Overall, $S3$'s results were the best since the distance stabilized around the 6m mark and the deviation and orientation stayed very close to 0 from episode 600 onwards.

Once the training was complete, the trained high-level neural network for each setup was tested for 100 episodes, each of 40s duration. The distance, deviation and orientation were measured at the end of each episode. A control setup $S0$ was created by using only the optimized CPG network
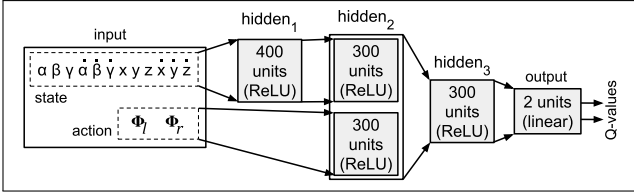
Fig. 2. Critic network. The 2 blocks in hidden$_2$ are merged by summation.

Fig. 4. Results for 100 test episodes, each with a maximum duration of 40s. Top: distance, middle: deviation, bottom: orientation.

without any high-level controller. To perform a comprehensive comparison, we also created 2 additional setups ($L1$, and $L2$) in which a simple linear feedback controller was used to modulate the CPG network by tracking the deviation of the robot from a straight trajectory. For modulating the CPG network, the linear controller used (7) for setting the values of Φ_l and Φ_r.

$$\left(\Phi_l, \Phi_r\right) = \begin{cases} \left(bound(\mathcal{G} \times |d_y|), 0\right) & d_y > 0 \\ \left(0, bound(\mathcal{G} \times |d_y|)\right) & d_y < 0 \end{cases} \quad (7)$$

In (7), $\mathcal{G}$ is the gain of the linear controller and $bound(x)$ restricts x within the range $[0, 1]$. Since the linear controller acts as the high-level controller, (4) is used to convert $\Phi_{l|r}$ to $\Psi_{l|r}$. The hyperparameters for $L1$ were $\xi = 0.1, \mathcal{G} = 0.2$ and those for $L2$ were $\xi = 0.1, \mathcal{G} = 0.4$. These values were empirically determined. The control setup $S0$ and the linear controller setups $L1$ and $L2$ were also tested for 100 episodes each. The performance of all the setups is shown in Fig. 4.

Fig. 4 (top) shows that the trained high-level controller was able to increase the forward distance walked by the robot, especially for the setups $S3$ and $S4$, for which the median distances are much higher than for the control setup $S0$. For $L1$ and $L2$ the median distance was better than $S0$ but considerably less than $S3$ and $S4$. The effect of the neural network-based high-level controller is also evident in Fig. 4 (middle) and (bottom), where the median deviation and orientation for $S3$ and $S4$ are very close to the ideal position of zero. The results for $S1$ and $S2$ are also better than $S0$ but not as good as $S3$ and $S4$. The linear feedback
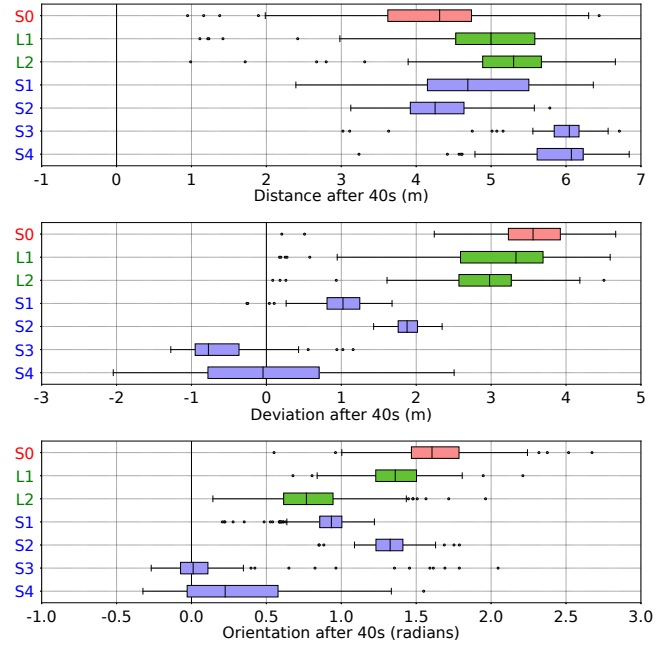
controller setups $L1$ and $L2$ also performed better than the standalone CPG ($S0$) but were not as effective in reducing the deviation and torso rotation as the neural network-based controller. It can be seen from Fig. 5, that for $S3$ and $S4$, the robot's trajectory was straighter and hence also longer than the other setups. The superior performance of $S3$ and $S4$ may be attributed to the fact that compared to the other neural network-based setups, $S3$ and $S4$ assigned a lower weight to the forward distance in the reward calculation, and thereby learned to pay more attention to the undesirable behaviors of lateral deviation and change in orientation. Additionally, for $S3$, the influence of the high-level controller was more (due to a lower value of ξ than $S4$), which may have contributed to the low variability in its performance compared to $S4$.
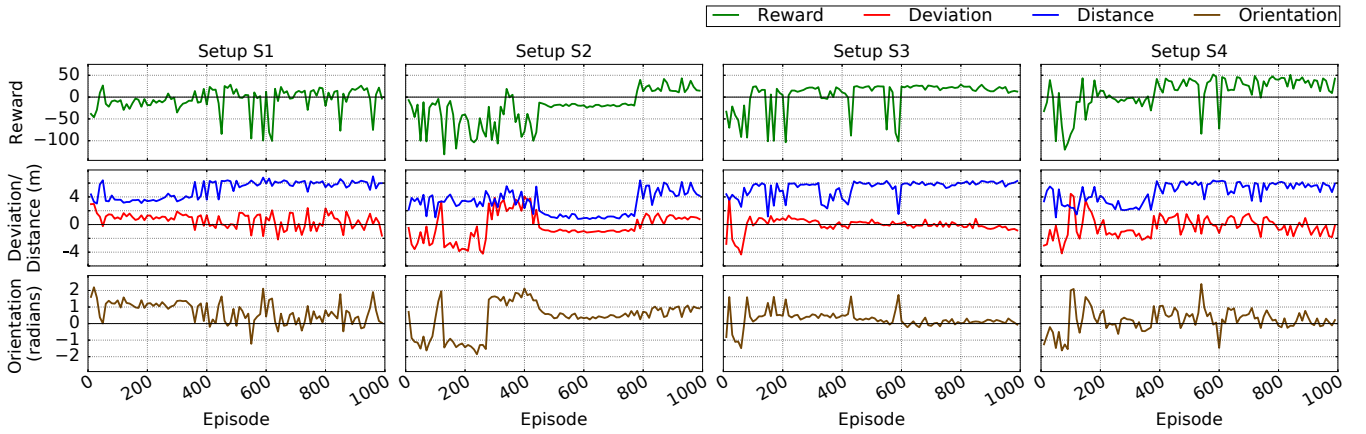

Fig. 3. Results of training the high-level controller using the setups S1-S4.

From Fig. 4, it is also evident that the simple linear equation used in (7) was inadequate in dealing with the asymmetry of the robot and the ground slippage, and was not as capable as the neural network in achieving the high-level objective. The robot's walk for the different setups can be viewed at `https://youtu.be/4c64rKhj72E`.
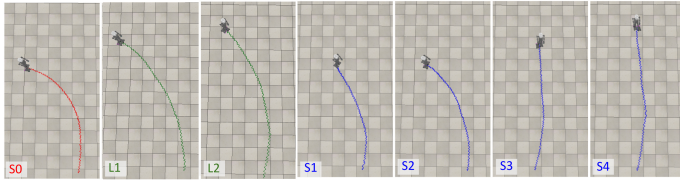

Fig. 5. Example trajectories for the different setups: S0, L1, L2, S1, S2, S3, S4 (left to right).

V. CONCLUSIONS

Although the optimized CPG network can produce a stable walk, errors in the robot model or noise in the actuation or the environment can always lead to deviations. The neural network-based high-level controller is able to deal with this problem effectively by using a simple 2-dimensional control signal and without having to explicitly control every individual joint. By using a relatively simple high-level objective of walking straight, we showed that a high-level neural network controller can be used to improve the performance of a CPG network for bipedal locomotion. The same approach can be extended in the future for more complex high-level goals, such as maintaining balance while walking on uneven or sloped surfaces. This can be achieved by letting the high-level controller modulate the bias position of joints, which would affect the tilt of the robot's body. The general approach can be implemented using a different robot, or by using different CPG network configurations or neural network architectures for the high-level controller. Also it is possible to train the high-level controller for multiple high-level objectives together by designing the reward function accordingly. We will also investigate the effects of using a simpler state representation and the effects of sensory noise and feedback delays. Confirming the advantages of our approach on the real NICO robot is our next step, but the presented results already show the promise of this biologically-inspired, developmental approach for achieving bipedal walking in uncertain environments.

REFERENCES

[1] S. Grillner, "Neural control of vertebrate locomotion-central mechanisms and reflex interaction with special reference to the cat," in *Feedback and motor control in invertebrates and vertebrates*. Springer, 1985, pp. 35–56.

[2] P. S. Stein, D. G. Stuart, S. Grillner, and A. I. Selverston, *Neurons, networks, and motor behavior*. MIT press, 1999.

[3] F. Delcomyn, "Neural basis of rhythmic behavior in animals," *Science*, vol. 210, no. 4469, pp. 492–498, 1980.

[4] P. Arena, "The central pattern generator: a paradigm for artificial locomotion," *Soft Computing*, vol. 4, no. 4, pp. 251–266, Dec 2000. [Online]. Available: https://doi.org/10.1007/s005000000051

[5] G. M. Shepherd, *Neurobiology*. Oxford University Press, 1994.

[6] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[7] K. Matsuoka, "Sustained oscillations generated by mutually inhibiting neurons with adaptation." *Biological cybernetics*, vol. 52, no. 6, pp. 367–376, 1985.

[8] ——, "Mechanisms of frequency and pattern control in the neural rhythm generators," *Biological Cybernetics*, vol. 56, no. 5-6, pp. 345–353, 1987.

[9] A. Ishiguro, A. Fujii, and P. Eggenberger Hotz, "Neuromodulated Control of Bipedal Locomotion Using a Polymorphic CPG Circuit," *Adaptive Behavior*, vol. 11, no. 1, p. 7, 2003. [Online]. Available: http://adb.sagepub.com/cgi/content/abstract/11/1/7

[10] J. Cristiano, D. Puig, and M. A. Garcıa, "Locomotion control of biped robots on uneven terrain through a feedback cpg network," in *XIV Workshop of Physical Agents*, 2013, pp. 1–6.

[11] J. Cristiano, D. Puig, and M. A. García, "Locomotion control of a biped robot through a feedback CPG network," in *ROBOT2013: First Iberian Robotics Conference*. Springer, 2014, pp. 527–540.

[12] ——, "Generation and control of locomotion patterns for biped robots by using central pattern generators," *Journal of Physical Agents (JoPHA)*, vol. 8, No. 1, pp. 40–47, 2017.

[13] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Automatic locomotion design and experiments for a modular robotic system," *IEEE/ASME Transactions on mechatronics*, vol. 10, no. 3, pp. 314–325, 2005.

[14] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2. IEEE, 2003, pp. 1620–1626.

[15] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, vol. 1, 2001, pp. 239–246.

[16] Y. Kamogawa, K. Yamada, H. Masuta, and H. O. Lim, "Stability control and pattern generation for biped humanoid robot," in *13th International Conference on Control, Automation and Systems (ICCAS)*, Oct 2013, pp. 910–915.

[17] Z. Yu, M. Zheng, Q. Zhou, X. Chen, L. Meng, W. Zhang, A. Ming, and Q. Huang, "Disturbance rejection controller for biped walking using real-time ZMP regulation," in *ROMANSY 21-Robot Design, Dynamics and Control*. Springer, 2016, pp. 179–188.

[18] M. Missura and S. Behnke, "Gradient-driven online learning of bipedal push recovery," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 387–392.

[19] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.

[20] T. McGeer, "Passive dynamic walking," *The International Journal of Robotics Research*, vol. 9, no. 2, pp. 62–82, 1990.

[21] G. Taga, Y. Yamaguchi, and H. Shimizu, "Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment," *Biological Cybernetics*, vol. 65, no. 3, pp. 147–159, 1991.

[22] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[23] M. Kerzel, E. Strahl, S. Magg, N. Navarro-Guerrero, S. Heinrich, and S. Wermter, "Nico - neuro-inspired companion: A developmental humanoid robot platform for multimodal interaction," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2017, pp. 113–120. [Online]. Available: https://www2.informatik.uni-hamburg.de/wtm/publications/2017/KSMNHW17/

[24] S. Auddy, "Hierarchical control for bipedal locomotion using central pattern generators and neural networks," Master's thesis, Universität Hamburg, Hamburg, Germany, Dec 2017. [Online]. Available: http://edoc.sub.uni-hamburg.de/informatik/volltexte/2018/237/

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.