IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 29, 2016, accepted April 18, 2016, date of publication May 18, 2016, date of current version June 3, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2570255

Gait Balance and Acceleration of a Biped Robot Based on Q-Learning

JIN-LING LIN[1], KAO-SHING HWANG[2], (Senior Member, IEEE), WEI-CHENG JIANG[2], AND YU-JEN CHEN[3]

[1]Department of Information Management, Shih Hsin University, Taipei 116, Taiwan
[2]Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan
[3]Department of Electrical Engineering, National Chung Cheng University, Chiayi 62102, Taiwan

Corresponding author: K.-S. Hwang (carlhwang@gmail.com)

ABSTRACT This paper presents a method for the biped dynamic walking and balance control using reinforcement learning, which learns dynamic walking without *a priori* knowledge about the dynamic model. The learning architecture developed is aimed to solve complex control problems in robotic actuation control by mapping the action space from a discretized domain to a continuous one. It employs the discrete actions to construct a policy for continuous action. The architecture allows for the scaling of the dimensionality of the state space and cardinality of the action set that represents new knowledge, or new requirements for a desired task. The balance learning method utilizing the motion of robot arm and leg to shift the zero moment point on the soles of a robot can maintain the biped robot in a static stable state. This balanced algorithm is applied to biped walking on a flat surface and a seesaw and is making the biped's walks more stable. The simulation shows that the proposed method can allow the robot to learn to improve its behavior in terms of walking speed. Finally, the methods are implemented on a physical biped robot to demonstrate the feasibility and effectiveness of the proposed learning scheme.

INDEX TERMS Reinforcement learning, biped robot, continuous action space, zero moment point.

I. INTRODUCTION

The design of a biped robot primarily involves balance control. A straightforward way to controlling of the balance of a two legged walking robot typically is by mimicking the style of humans' walking locomotion. The structures of biped robots, similar to humans' skeletons, are different from the wheeled robots or industrial robot arms, and are capable of much flexibility and adaptability in do many human-like motions. However, how to make robots walk stably is still an importance issue. Recently many robot's walking controlling methods are concentrated on ways to maintaining the projection of the center of gravity (CoG) of a biped within a support area. Some research has developed mathematical models for biped robots, although the mathematical calculation is a complex task [1]. To reduce the computational load and enhance the robustness of biped walking for robots, various learning methods based on reinforcement learning (RL) have been studied on biped robot for walking behaviors [2]–[4]. In the walking process, many studies based on RL methods calculate the CoG of the biped robots [2], [3], but these methods only consider a discretized state space. In [5], a trajectory planning method is introduced to solve a biped's walking motions. In [6], optimal walking gaits based on Zero Moment Point (ZMP), which needs to stay in a support area criterion all the time, are considered. In [7] and [8], pressure sensors are set on the foot of the biped robots and are used to calculate the ZMP's position by a feedback-force system. In addition, the ZMP's positions are input to a fuzzy controller to decrease the error. The goal is to ensure the position remains in a stable region. The tendency to fall down can be determined based on the ZMP obtained as a cross point of the resultant vector of inertial and gravitational force and the ground [9], [10]. Other studies of biped walking utilizing the ZMP criterion are focused on forcing the computed or sensed ZMP to remain inside the support area by initially planning the reference trajectories of the center of mass of the body [1], [4]. Some studies have proposed standard methods for gait synthesis based on the ZMP to assure the dynamic stability of a biped robot. Basically, these methods consist of two stages. The first stage is designing the desired ZMP trajectory and the second stage is correction of the movement of the torso and pendulum to materialize the defined trajectory in balance. However, because the change in the ZMP due to the movement of the torso is limited, not all desired ZMP trajectories are possible. The ZMP position can be obtained computationally using the

mathematical model of a robot. However, significant error between the real ZMP and the calculated one is possible due to the differences between the physical parameters of the real robot and its approximated mathematical model. Thus, the real ZMP should be measured in order to obtain a steady stable gait. The ability to improve robot behavior, such as balanced walking, through learning is the ultimate challenge of AI and robotics [2], [3], [11], [12]. Learning models of complex tasks can aid in the design of appropriate control laws for robots in unstructured environments with uncertainties. These uncertainties are primarily due to imprecise or unobtainable sensory data, together with unpredictability of the environment; that is, a lack of full knowledge of the environment's characteristics and dynamics. RL agents allow autonomous systems to learn from their experiences instead of exclusively from knowledgeable teachers [13]. Therefore, RL agents are appropriate for practical systems such as robot learning and control. Many researchers have investigated intelligent control methods to solve biped walking problems [2]–[4], [14]. RL is one kind of intelligent learning method for the walking tasks, and RL methods are used to control the ZMP position for walking stability. In this paper, we consider how to improve biped gait using an RL agent with measured ZMP feedback. The learning balance method is applied to the dynamic balance of a biped robot. The learning method has no any priori knowledge about the biped locomotion, so it needs to use measured ZMP feedback to adjust its walking behaviors. To enhance the biped robot's walking behaviors, how to design the action space to fit the biped robot is a difficult task. In RL applications, the action space has two kinds of properties, discretized action space and continuous action space. Most of the RL methods apply one of two kinds of methods for the continuous action space, either a neural network or combinations of some discrete actions chosen by some activated nodes, for example, the tile coding and the ART-based cognitive models. The stochastic real value (SRV), one of the neural network algorithms, provides an idea to create a real value action [15], [16]. However, since it uses the critic network to estimate the immediate reward and update the actor network to obtain the best immediate reward, it is only suitable for greedy problems. Otherwise, it would easily fall into a local minimum. The proposed method with a self-organized state aggregation mechanism was developed to deal with the walking problem in continuous action domains. The goal in this paper proposes a continuous action method to fit the walking behavior problem for a biped robot. It extends the discretized action space to continuous action space. The simulation and experiment results show the proposed method can make the biped robot walk stably in different environments. The remainder of this paper is structured as follows. Section II details the structure of Q-Learning and the RL algorithm. In section III, the proposed method is introduced. In section IV, the simulation and experiment results are reported; a section V concludes this paper.

Algorithm 1 One-Step Q-Learning Algorithm

Initialize $Q(\boldsymbol{s}, a)$ arbitrarily
Repeat(for each episode):
1. Perceive a state $s \in \boldsymbol{s}$, and then select an action $a \in A$ using policy derived from $Q(\boldsymbol{s}, a)$.
2. **Repeat**(for each episode):
3. Take action a, observe next state $\boldsymbol{s}'$ and next reward r.
4. Update the action-value function by $(\boldsymbol{s}, a, \boldsymbol{s}', r)$.
5. $\boldsymbol{s} \leftarrow \boldsymbol{s}'$
6. **until** $\boldsymbol{s}$ is terminal

II. BACKGROUND

Reinforcement learning is used to learn a policy that maximizes the long-term reward for all states [13]. Q-learning is a well-known temporal difference and model-free RL [17], [18], and it can be divided into five parts [19]. During the learning process, the agent selects an action $a \in A(\boldsymbol{s})$ according to its own policy in the state $\boldsymbol{s} \in \boldsymbol{S}$. After taking this action, the agent will transfer to the next state $\boldsymbol{s}'$ and receive an immediate reinforcement signal r, which is returned from the environment. The decision-making is strengthened or weakened according to this reinforcement signal. In other words, when the agent takes an action a in a state $\boldsymbol{s}$, an action-value function $Q(\boldsymbol{s}, a)$ is used to update the policy. Based on these evaluated action-values, the agent can learn how to search for an optimal policy. The Q-value with state s and action a are updated as follows

$$Q(\boldsymbol{s}_t, a_t) = Q(\boldsymbol{s}_t, a_t) + \alpha(r + \gamma \max_{a_{t+1}} Q(\boldsymbol{s}_{t+1}, a_{t+1}) - Q(\boldsymbol{s}_t, a_t)) \quad (1)$$

where $\alpha \in (0, 1]$ is used as the learning rate. The $\gamma \in (0, 1]$ is the temporal discount factor. After many iterations, the Q-values converge to Q^* [18]. The detailed procedure of Q-Learning is listed in **Algorithm 1**.

III. LEARNING A BALANCE METHOD FOR BIPED ROBOTS

In this section, Continuous-Action Q-Learning (CAQ) is proposed to extend discrete action space to continuous action space. In the original Q-Learning, it is necessary to design the action space to form a discrete action space. The architecture of the Continuous-Action Q-Learning is shown in Fig. 1.

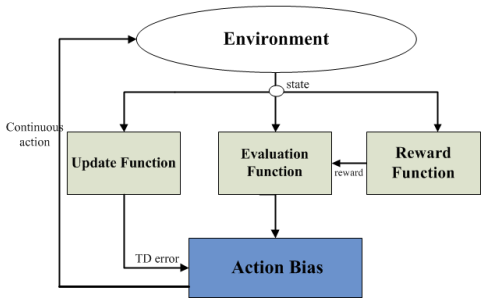


FIGURE 1. Architecture of the continuous-action Q-learning.

The state space maps the sensory input vector from the environment in the learning process, and the reward function is used to evaluate the action that the agent takes. The action bias generates a variety of real actions. The biped robot performs a walking behavior in one cycle, as shown in Fig. 2. In Fig. 2(a), for biped robot walking, the robot is standing with one leg and the ideal ZMP's position is designed into two positions, at the center of a foot and on tiptoe. When the robot is walking, the robot standing with one leg is more unstable than when it stands on both. The robot learns to maintain the ZMP's position in a stable region with one leg. If the position remains in the stable region, the robot will be more stable during walking behavior. In the beginning, one cycle obtains 7 phases to achieve motion straight forward motions and ZMP transferring. Phase 1 is a starting gait pattern, but it will be ignored in the next walking cycle. According to this figure, the robot is using one leg in phases 2, 3 5, 6, and the ZMP's position is transferred to the another leg in phase 4, 7. Q-Learning was used to learn which actions could make walking more stable between two phases, one leg and two legs. Furthermore, the Q-Learning with knowledge about walking on a flat surface is transferred to another environment, the seesaw, a process referred to as Knowledge Transfer Learning (KTL). The learning time could definitely be reduced to accelerate the biped robots' walking motion.

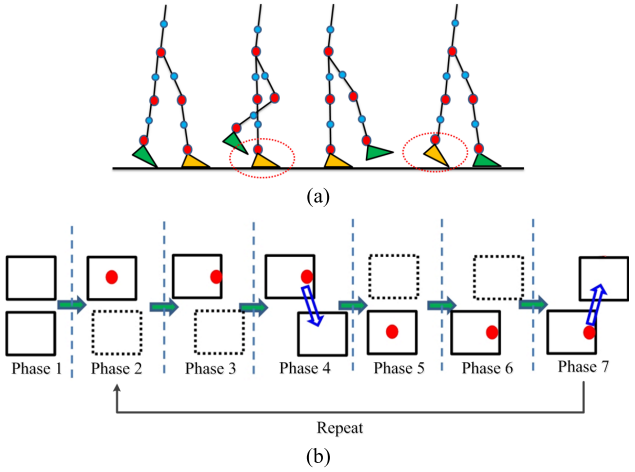


FIGURE 2. Biped robot walking pattern. (a). The biped robot walking pattern (b). ZMP position of biped walking sequence.

A. LEARNING THE WALKING BEHAVIORS BY Q-LEARNING

The gait patterns formed the biped robot's walking phases. So the Q-Learning assigned to each phase is used to adjust appropriate gaits. Finally, the robots in each phase could have a more stable walking motion.

1) STATE SPACE CONSTRUCTION

The agent perceives the information that is called sensory input vectors which are continuous environmental information from the environment. In Fig. 3, when the agent perceives the sensory input vectors from the environment, if the dimensions of the vectors are large, then the curse of

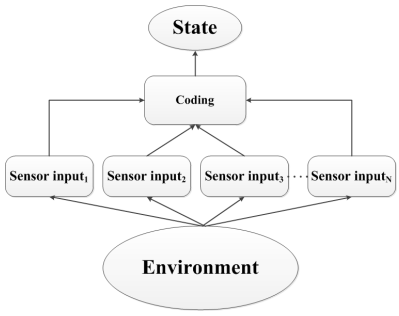


FIGURE 3. The state is derived from the environment.

dimensionality problem occurs [20]–[22]. To decrease the influence of the curse of dimensionality problem, the agent needs to generalize the sensory input vectors to form finite-discrete state space. In Fig. 3, after coding the sensory input vectors, the state of the agent will be formed. In this paper, the finite-discrete state space is shown in Fig. 4. The ZMP coordinates (x, y) of the robot's gravity are projected on the range of the robot's foot. Force sensors, f_1, f_2, f_3 and f_4 are installed at the two feet of the biped robot shown in Fig. 4(a). According to torque equation, the center of the foot is set as the origin point. The ZMP coordinates are calculated as follows.

$$\begin{aligned} x &= W((f_2 + f_4) - (f_1 + f_3))/2(f_1 + f_2 + f_3 + f_4) \\ y &= L((f_1 + f_2) - (f_3 + f_4))/2(f_1 + f_2 + f_3 + f_4) \end{aligned} \quad (2)$$

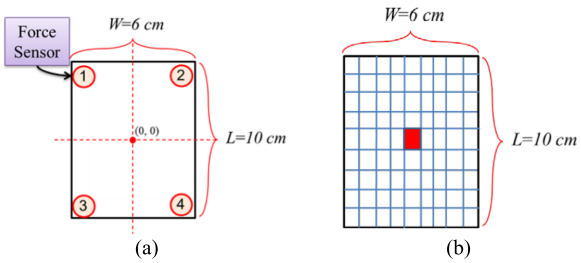


FIGURE 4. The finite-discrete state space. (a) Construction of one robot foot (b) Definition of stability regions and state partition.

After calculating the ZMP coordinates, the finite-discrete state space formed as the subregion at the bottom of the robot's feet and the space is divided into 81 cells as shown in Fig. 4(b). The red cell in the center of the robot's foot means the most stable region. If the ZMP coordinates fall within the 81 cells, it will indicate that the biped robot remain without falling down problem. However, there is a special situation. The robot falls down in that the coordinates don't fall in these cells. So, a new cell needs to be assigned in this situation. Therefore, the total number of elements in state space, $|S|$ includes 82 cells.

2) ACTION SPACE

After perceiving the environmental information to form the state, the robot needs to select an action by estimating the

appropriate policy. A huge action space will influence the learning efficiency, because the robot has to explore many times to find a good action in the state. Because the biped robot includes 18 degrees of freedom, they are mapped to the 7 elements $\{hip, leg_r, leg_l, arm_r, arm_l, shoulder_r, shoulder_l\}$, as shown in Fig. 5. The element *hip* is to control the angle of hip joint; leg_r and leg_l are to adjust the swing angles of corresponding left and right foot which can lift forward or backward. The elements arm_r and arm_l are to adjust the swing angles of the corresponding left and right arms which can move up and down. The elements $shoulder_r$ and $shoulder_l$ use the same control method as arms. The following section will explain how to make the action space discrete by these seven joints to form the joint action vector $\boldsymbol{\theta}_b = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7\}$.

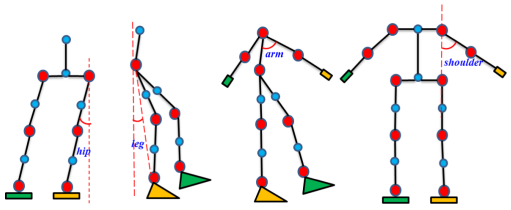


FIGURE 5. Control elements.

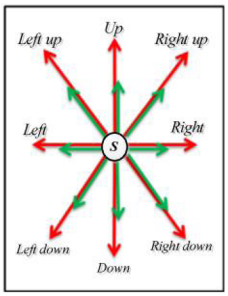


FIGURE 6. Motion vectors of ZMP by 16 join actions.

a: DISCRETE ACTION SPACE

The action space in this balance method is used to control the ZMP's movement directions. There are 8 directions of movement for the ZMP's directions, as shown in Fig. 6. One direction includes two kinds of movement distances, the green line and the red line. The green line can make the ZMP move a short distance and the red line can make it move a long distance. So, the total number of actions is 16. These movement directions are controlled by the joint action vectors, which are composed of 7 angle elements, $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7\}$ with the variations of angles, $\boldsymbol{\Delta\theta} = \{\Delta\theta_1, \Delta\theta_2, \Delta\theta_3, \Delta\theta_4, \Delta\theta_5, \Delta\theta_6, \Delta\theta_7\}$. Thus, the discrete action is shown as follows.

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_\mathbf{b} + \Delta\boldsymbol{\theta}_k \tag{3}$$

where $k = 16$ indicates the cardinality of a discrete action space.

For example, the ZMP's directions " ↗" can be expressed as $\{\theta_1 - 3.9375, \theta_2 + 0, \theta_3 - 4.875, \theta_4 + 7.5, \theta_5 + 0, \theta_6 + 9.375, \theta_7 + 0\}$, and this control makes the ZMP's directions move to the expected position by increasing or decreasing each parameter. In the same way, the ZMP's directions " ↗" can be expressed as $\{\theta_1 + 8.625, \theta_2 + 0, \theta_3 + 9, \theta_4 - 18.75, \theta_5 + 0, \theta_6 - 15, \theta_7 + 0\}$.

b: CONTINUOUS ACTION SPACE

One of the objectives of this work is to extend conventional Q-Learning to generate continuous effort from a set of actions. The continuous state problem is always an issue in reinforcement learning. The architecture of the proposed algorithm can be divided into two layers; the first of which is similar to conventional Q-Learning. One important issue is that if the discrete action space is designed to be low resolution, so the biped robot cannot adjust its motion accurately. On the other hand, if the space is high resolution; the robot needs to spend much time to learn policy. However, on the second layer, the concept of the stochastic action generation method is applied to reinforcement learning. In addition, the reward returned from the environment is based on the maximal Q-value referred to in the first layer. In other words, this algorithm uses the Gaussian distribution to produce a stochastic and real-valued output, and tries to record the maximum expected future reward and the action under this policy. The algorithm adjusts the mean and the variance of the Gaussian distribution so as to increase the probability of producing an optimal real-valued output for each input pattern. This proposed method expands the robot's discrete action into a continuous action. Before using this method, it is necessary to first have a discrete action space. The action bias extends an equation that is for discrete action to form continuous action. As illustrated in Fig. 7, each paradigm action has a corresponding action bias mean, $\boldsymbol{B}_m$, and an action bias variance, $\boldsymbol{B}_v$. The items, $\boldsymbol{\Delta\theta}$ are used as the paradigm action and the corresponding action bias mean. The items, $\boldsymbol{B}_v$ are set as the upper and lower bounds of one direction, as shown in Fig. 6. One direction has two kinds of moving distances. In the same direction, the red line is set as the upper bound and the green line is set as the lower bound. One moving distance needs to control 7 angle elements. While the agent obtains the state from the environment and evaluates an action, the algorithm utilizes the normal distribution to

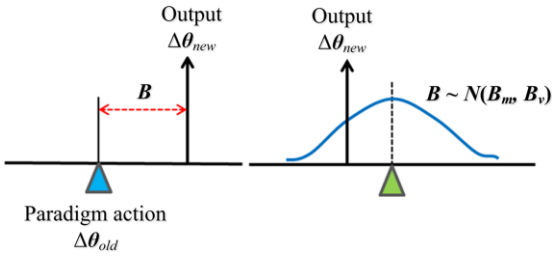


FIGURE 7. Illustration of action bias.

find the action bias, $\boldsymbol{B}\sim\boldsymbol{N}(\boldsymbol{B}_m, \boldsymbol{B}_v)$ in each angle element. $\boldsymbol{B}_m$ and $\boldsymbol{B}_v$ are the action bias mean and action bias variance to the selected action.

$$\Delta\theta_{i,new} = \Delta\theta_{i,old} + B_i \tag{4}$$

where $i = 1, 2, \ldots, 7$, $\Delta\boldsymbol{\theta}_{i,new} \in \Delta\boldsymbol{\theta}_k$, $B_i \in B$.

The action bias creates a suitable action as the output of the agent. Then, the value of the action bias mean and action bias variance are updated according to the following method.

(i) ACTION BIAS MEAN: The action bias mean is used to slightly adjust the quantized action. In addition to tailoring conventional Q-learning for the application of biped robot walking learning, one of the objectives of this paper is to expand the discreet action space to a continuous one to make actions more dexterous and improve the performance. The action bias mean is updated according to the estimation error of the sate-action value rather than the immediate reward. The estimating error of the state action value is obtained as follows.

$$\Delta Q(s, a) = (r + \gamma \cdot Max(Q(s', a')) - Q(s, a)_{old}) \tag{5}$$

After obtaining the estimating error, the action bias mean is updated as follows.

$$B_m \leftarrow B_m + \beta_m |\Delta Q| [B - B_m] \tag{6}$$

where β_m is a learning rate.

The explorative direction demonstrates the difference between the action bias and the action bias mean. The Q-function is utilized to calculate the estimated error, shown in equation (5), and to criticize the explorative performance rather than the estimated error of the immediate reward. If ΔQ is negative, it means that B is worse than B_m, but it does not indicate what the suitable bias is. The learning rate might be a smaller value. On the other hand, when ΔQ is positive, it means that B is better than B_m, and thus the learning rate can be a large value to speed up the learning. For that reasons, β_m can be set as different values according to the estimated error as follows.

$$\beta_m = \begin{cases} \beta_p & if\ \Delta Q \geq 0 \\ \beta_n & if\ \Delta Q < 0 \end{cases} \quad \text{where } \beta_p > \beta_n > 0. \tag{7}$$

where β_m represents a learning rate using in action bias mean.

When the ΔQ is greater than or equal to zero, it means the learning action bias mean is effective, and the correct explorative direction should be adjusted quickly. On the other hand, if the ΔQ is smaller than zero, the explorative direction should be adjusted slowly.

(ii) ACTION BIAS VARIANCE: The action bias variance performs exploration. Since larger variance leads to more exploration, the variance should be decreased with learning. The action bias variance should be small when the action bias mean indicates the proper value, which is not known in advance. This only guarantees that the action bias mean is updated toward the suitable value. It also indirectly implies that if ΔQ is approximately 0, the action bias mean is inclined toward the suitable value. Therefore, the action bias variance is updated as follows.

$$Bv \leftarrow Bv + \beta_v(|\Delta Q| - Bv) \tag{8}$$

where β_v is a learning rate.

3) REWARD FUNCTION

In the learning process, after an agent passes a decision action to the environment, the environment returns a reward to the agent. The reward function provides a learning guideline for the agent. When the environment returns a good reward to the agent, it will strengthen the decision action; on the other hand, when the agent gets a bad reward, it will weaken the decision action. In other words, the agent can use the reward function to update the learning policy. In this learning task, the reward function gives the biped robot positive rewards for moving forwards with good balance, and negative rewards for moving unstably or falling over. The reward function is designed by the ZMP's position. In Fig 4(b), the stable region is set in the middle of the sole of a foot. When the ZMP's position can be moved to this region, the controlled action is viewed as good decision-making. In Fig. 8, the red point in the center represents the ZMP's desired position. The reward function is established as follows:

$$R(\mathbf{s}) = \begin{cases} -10 & if\ fallingdown \\ 0 & if\ fallingstableregion \\ -\lambda\sqrt{(x-x_0)^2+(y-y_0)^2} & Otherwise \end{cases} \tag{9}$$

where $\lambda \in [0, 0.5]$ is a predefined constant.

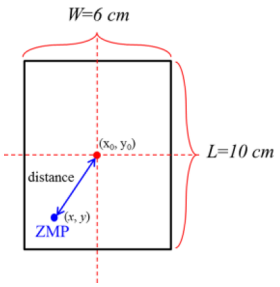


FIGURE 8. Setting of ideal ZMP position and calculation of the reward.

If a robot walks stably, the transient ZMP must reside in the stable region and the reward is set to 0. In the worst case, the robot falls down, the episode ends, and it is penalized by a -10 reward value. On the other hand, if the current state is out of that region, the robot obtains a penalty of distance between the center point and ZMP's coordinate. This reward function allows the robot learn stable walking. The policy selects a proper action according to the current state. The goal of training the policy is to find an optimal action that can maximize the Q-value associated with that state. The learning rule used to adjust the Q-value can be described as in equation (1). The algorithm of the proposed method is briefly

Algorithm 2 Continuous-Action Q-Learning Algorithm

Initialize $Q(\boldsymbol{s}, a)$ arbitrarily for all state $\boldsymbol{s}$ and action a

Repeat(for each episode):

1. Perceive the current sensory input vector from the environment to form state $\boldsymbol{s}$.
2. Select an action a according to the action-value function and generate a real-valued action depends on (4).
3. Take the real-valued action, receive an immediate reward r and observe the next state $\boldsymbol{s}'$.
4. Update the action-value function as equation (1).
5. Update the action bias mean by (6).
6. Update the action bias variance by (8).
7. $\boldsymbol{s} \leftarrow \boldsymbol{s}$'
8. **until** $\boldsymbol{s}$ is terminal

summarized in **Algorithm 2**. The action bias mean and action bias variance will be updated by the reward because it can be used to adjust the ΔQ. After learning iteration, action bias will correct suitable value for controlling the robot action. The action space is extended from Fig. 6 to Fig. 9.

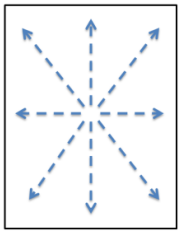


FIGURE 9. Moving distance controlled by the continuous action.

B. KNOWLEDGE TRANSFER LEARNING BASED ON CONTINUOUS-ACTION Q-LEARNING

The concepts of knowledge transfer learning are derived from human beings. For example, when a human learns how to walk on flat ground and climb slope, they can realize to walk on the seesaw. To apply these concepts for biped robot, the CAQs are implemented to this idea, named knowledge transfer learning (KTL). Using the KTL can be divided into two parts. The first part implements CAQs to solve some learning problems. In this paper, there are three learning tasks, walking on flat ground, uphill and downhill. These learning tasks are solved using three CAQ methods. To merge three CAQ methods to apply for another learning task, the seesaw, another Q-learning is used to learn how to switch three knowledgeable CAQs, which can be fitted to different situations. For example, the seesaw task is divided into three stages as shown in Fig. 10. If the current situation in this task

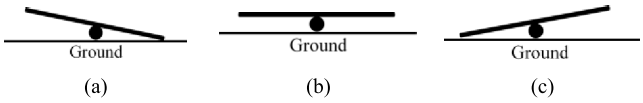


FIGURE 10. The seesaw task. (a). First Stage (b). Second Stage (c). Third Stage.

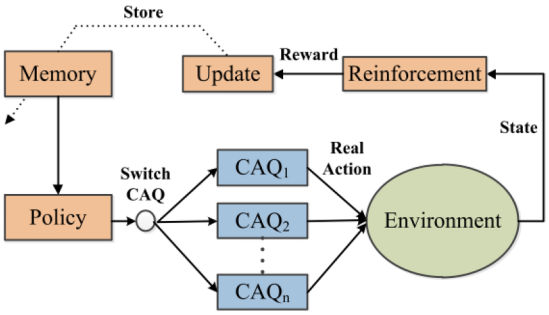


FIGURE 11. Architecture of KTL.

is the first stage, the robot needs to use its uphill experience. If the robot walks to the middle seesaw, the robot should switch to the walking on flat ground experience. Finally, the robot has to use its downhill experience. The architecture of KTL is shown in Fig. 11. The KTL method uses the seesaw's board as the state space, and the action space includes some actions, which are used to switch the CAQ's experience by the perceiving states. In the process of KTL, the learning tasks are executed based on existing experience (CAQ_1, CAQ_2,…,CAQ_n). The reward function of KTL is set as follows.

$$R(\mathbf{s}) = \begin{cases} -10 & \textit{if fallingdown} \\ e^{-c} - 1 & \textit{Otherwise} \end{cases} \tag{10}$$

where c is the number of the state transition between $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$. If the value of c is small, then the environment will return a good reward. On the other hand, if the c is large, then the environment will return a bad reward. When the robot falls down, then the environment will return the worst reward. The reward function makes the robot select an appropriate experience to let the robot walk on the seesaw stably.

IV. SIMULATIONS AND EXPERIMENTS

To demonstrate the proposed methods, the CAQ and KTL approaches are verified in the simulator, Webots and in the actual environment. The learning results derived from the simulator are mounted on physical robots, called Bioloid robots, in the actual environment. The simulated and experimental results verify that the Bioloid can walk stably on the different walking environments.

A. SIMULATED AND EXPERIMENTAL ENVIRONMENT

The simulation environment is shown in Fig. 12. In Fig. 12(a), the size of this field with physics properties (gravity and friction) is 6(m) × 8(m). These properties will increase the difficulty in the learning process. In Figs. 12(b) and 12(c), the biped robot learns to walk up and down a slope with the degree of slope of about 5. The seesaw is set to verify the KTL method, as shown in Fig. 12(a). In the experiment, the walking environment is similar to the simulated environment.

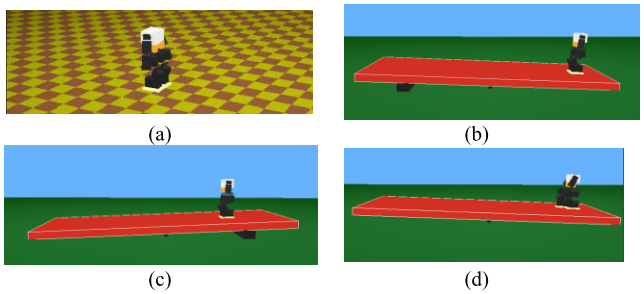


FIGURE 12. Walking environments in the simulation. (a) Walking on flat ground. (b) Uphill environment. (c) Downhill environment. (d) Seesaw.

B. SIMULATED ROBOT AND REAL BIPED ROBOT

The biped robots are the same and illustrate their three-dimensional link structure in the simulator and real environment, as shown in Fig. 13. In Fig. 13(a), the biped robot is built in the Webots simulator. The real robot is shown in Fig. 13(b). The robots are actuated by 18 RC servo motors with a mass of about 1.91 kg in total. In other words, the robots include 18 degrees of freedom (DOF). Each arm has three degrees of freedom (DOF) and each leg has six active DOFs; three DOFs are available at the hip and there is one DOF at the knee and two DOFs at the ankle joint. Each foot is equipped with four force sensors to provide the contact forces between the feet and the ground, as shown in Fig. 13(a) and Fig. 13(b). In the experiment, four ASAKUSA force sensors (AS-FS) are equipped on the bottom of each foot. Because the simulation environment is similar a real environment, the learning experience is valuable for implementing on the real robot.

C. SIMULATION RESULTS

The simulation results include two parts. The first part examines the CAQ method in three learning tasks. After the learning experience, three CAQs with different experience are merged by the KTL method and implemented in the seesaw task. In the CAQ method, when the robot takes an action, this paper introduces a policy listing as follows.

$\varepsilon = 1/\left(1 + e^{-E/T}\right)$
$\delta \leftarrow$random(0, 1)
if $\delta > \varepsilon$
  The robot selects a random action.
else
  The robot selects an action with max Q-value.

where E is the number of episodes and T is a decay parameter.

The reward function, which is returned from the environment, is given according to the ZMP of the biped robot. If the ZMP remains in the stable region, which is described as a simple walking state, then the environment feeds back a reward of 0. These results demonstrate the task of the balance control for the biped robots.

1) LEARNING TASKS BY USING CAQ

The first learning task is to walk on flat ground, as shown in Fig. 12(a). In one training round, the robot walks in a straight line for each episode. The results are compared with another method that was proposed in [23], where a discrete-action Q-Learning (DAQ) is introduced to control the biped robot. In the learning process, when the number of learning steps reaches 300 or the robot falls down, an episode is terminated. The robot in each simulation runs 2500 episodes per round. The parameter settings in the learning problem are listed in Table 1. Fig. 14 shows the average number of learning steps required for the biped robot to walk on flat ground for each episode. The x-axis represents the episodes and the y-axis represents the learning steps. The two curves represent DAQ and CAQ. It can be seen that the proposed CAQ shows clear improvement in learning speed. This method makes the robot walk without falling down. Although the CAQ lets the biped robot walk stably, but the continuous action space in the CAQ moves the ZMP shortly in the former stage as

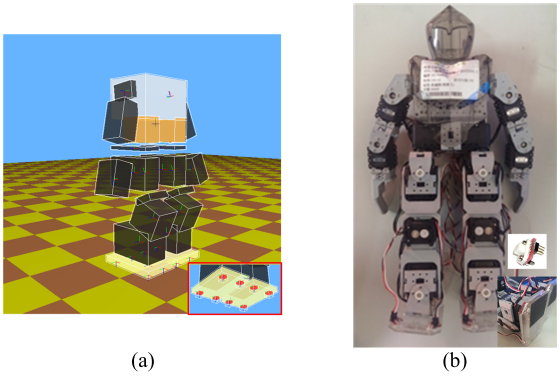


FIGURE 13. Biped robots. (a) Simulated robot. (b) Real robot.

TABLE 1. Simulation parameters.

Parameters	Value
Learning rate(α)	0.9
Discount rate(γ)	0.9
β_p	0.6
β_n	0.4

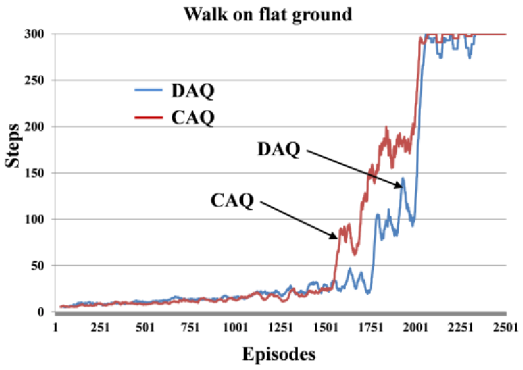


FIGURE 14. Comparison of learning steps between CAQ and DAQ.

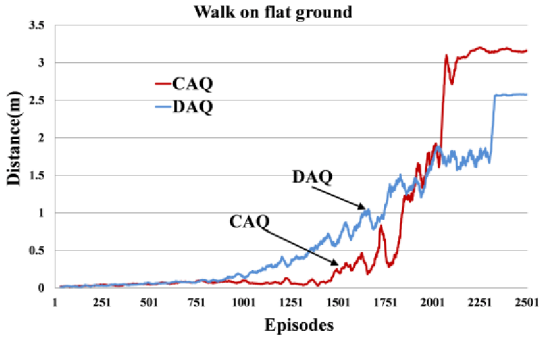


FIGURE 15. Comparison of walking distances between CAQ and DAQ.

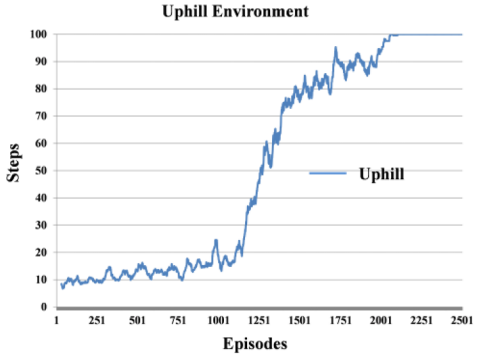


FIGURE 16. Learning curve for the robot walking uphill.

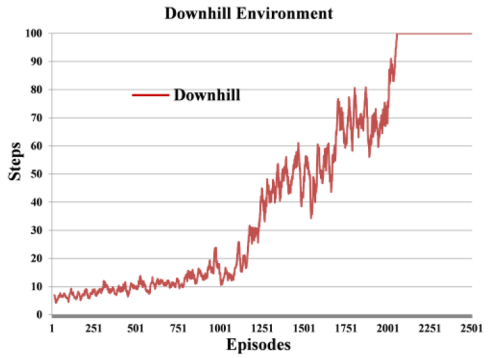


FIGURE 17. Learning curve for the robot walking downhill.

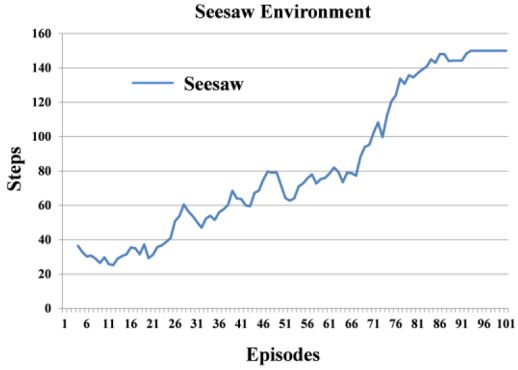


FIGURE 18. Using Knowledge Transfer Learning for walking on the seesaw.

shown in Fig. 15. The x-axis represents the episodes and the y-axis represents the walking distance. In the latter stage, the action bias can be adjusted to an appropriate value, so the biped robot can walk stably and quickly and the walking distance will overtake the DAQ in the 2000 episode. The longest distance using CAQ is approximately 3.3m. Details of the walking distance in the last five episodes are shown in Table 2 where the CAQ makes the robot walk on flat ground farther than DAQ. To demonstrate CAQ, uphill and downhill learning tasks are introduced; there are the same tasks. In the uphill task, the robot starts on the top of the slope, as depicted in Fig. 12(b). In the downhill task, the robot starts on the bottom of the slope, as shown in Fig. 12(c). The degree of the slope is about 2.5. In the two simulations, the two CAQs are used to control the biped robot to learn to walk on these two environments. In these simulations, the robot runs for 2500 episodes and 100 steps in an episode. In each episode, if the number of steps is greater than 100 and the robot does not fall down, the episode is terminated. The parameters for the simulation are the same as Table 1. Figs. 16 and 17 show the learning curves of the robot walking uphill and downhill, respectively. These two figures indicate that the robot frequently meets the falling down problem in the former stages. The robot reaches 30 learning steps in episode 1200 and episode 1255, and then the robot will walk more and more stably. The uphill curve shows that the robot reaches 100 steps in 2100 learning episode. The downhill curve shows that the robot reaches 100 steps in the 2105 learning episode. By using the uphill and downhill simulations; it is verified that the CAQ can be applied for various learning tasks.

2) LEARNING ON THE SEESAW BY USING KTL

Humans can use their experience to successfully execute different tasks. For example, when humans have experience moving uphill and downhill, they can apply their experience to move up and down stairs. Similarly, KTL is used to achieve these concepts, and this method allows the robot to transfer existing experience to accomplish the tasks. Therefore, a demonstrating task, walking on the seesaw is designed to test the KTL, and the architecture is shown in Fig. 11. In this learning demonstration, the robot first develops experience from different learning experience. In this paper, the different types of experience are established from three tasks, walking on flat ground, uphill and downhill. After establishing these types of experiences, another Q-Learning cycle learns how to switch these experiences. In this simulation, the board of seesaw tilts its position based on the robot's position. Three kinds of position are listed in Fig. 10, and the moving position makes the robot walk on the board unstably. The simulation result is depicted in Fig. 18, according to which, the robot walks on the seesaw stably by using 100 episodes. The established experience can speed up the robot's learning. To verify the efficiency of KTL, the four experiences directly used in

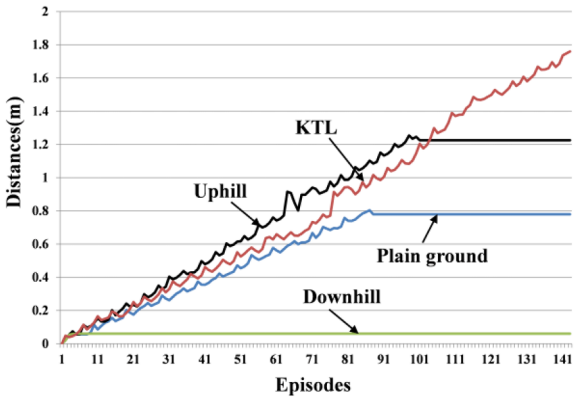


FIGURE 19. Using different learning experiences for walking on the seesaw.

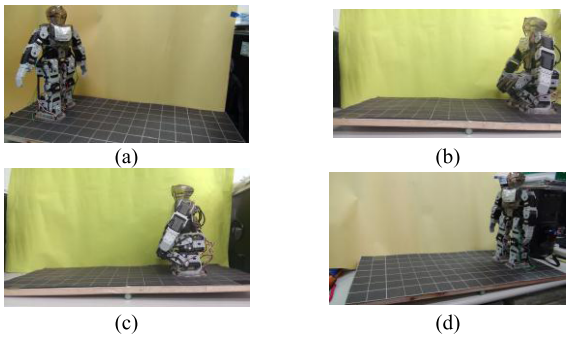


FIGURE 20. Walking environments in the experiment. (a) Walking on flat ground. (b) Uphill environment. (c) Downhill environment. (d) Seesaw.

TABLE 2. Detailed comparisons of DAQ and CAQ.

Episodes	Distance(CAQ)	Distance(DAQ)
2496	3.1309(m)	2.5724(m)
2497	3.2305(m)	2.5996(m)
2498	3.2682(m)	2.5727(m)
2499	3.1559(m)	2.5725(m)
2500	3.2442(m)	2.5728(m)

the seesaw task and the results are listed in Fig. 19. The x-axis indicates the episode and the y-axis indicates the walking distance on the seesaw. The curve of green line represents using downhill experience for walking on the seesaw. The curve of the blue line represents using the experience of walking on flat ground; the curve of the black line represents using uphill experience, and the curve of red line represents using KTL experience. In this figure, the downhill experience is not sufficient for the robot to walk on the seesaw stably. Because the seesaw in the initial stage is similar to the uphill task, as shown in Fig. 10(a), the experience cannot be used for walking on the seesaw. The uphill experience is the best for walking on the seesaw. Then robot walks to the latter part of the seesaw's board, as shown in Fig. 10(c). The walking distance of uphill is 1.22(m) and the walking distance of flat ground is 0.77(m). Thus the KTL method can use the downhill experience to walk stably on the seesaw. However, the KTL method consumes some time to switch the experience, so it is slower than uphill experience. But the KTL is the only method that can finish the seesaw task.

D. EXPERIMENTAL RESULTS

This section implements the simulation results to the real robot in different kinds of learning environment, including flat ground, uphill, downhill, and seesaw, as shown in Fig. 20. The settings of the experiment are the same as the simulations. The AS-FS single-point force sensors are used to calculate the ZMP's position. The perceived values of force sensors are analog signals. An AGB65-ADC converter that is equipped on the back of the robot transforms the signals to digital signals. These signals are sent to the computer system using a Bluetooth module that is installed on the robot. Then the computer will send a controlling action to the robot after receiving these sensor values. In Fig. 20, the size of the wooden board is 35(cm) × 80(cm) and is marked 5 (cm) × 5 (cm) grids. To decrease the amount of time needed, the simulation results are directly implemented to the actual robot. Fig. 21 shows that the robot can walk on flat ground. Fig. 22 shows that the robot learns to move uphill, and Fig. 23 shows it learns to move downhill. The three figures, Fig. 21(a), Fig. 22(a), and Fig. 23(a) show that the robot starts in the initial position. The robot takes an action in a state by using the sensor values. The three figures, Fig. 21(b), Fig. 22(b), and Fig. 23(b) show the robot walking in the

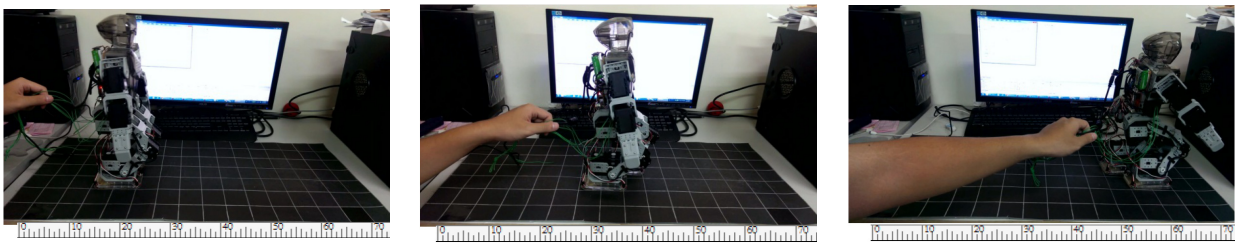


FIGURE 21. The experiment result for flat ground. (a) Initial stage. (b) Middle stage. (c) Final stage.

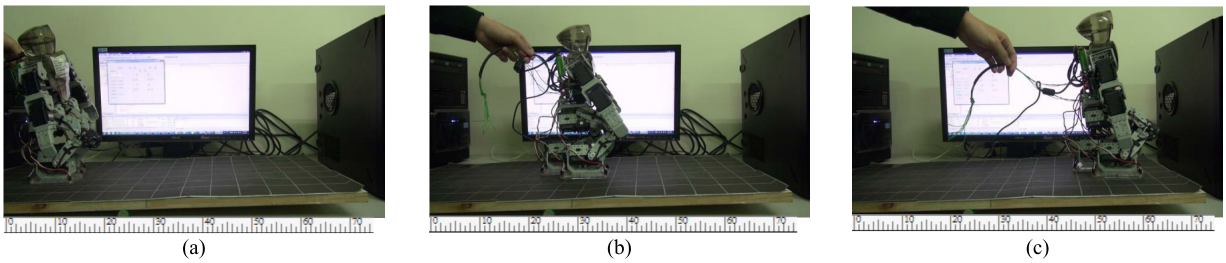


FIGURE 22. Experimental results of uphill. (a) Initial stage. (b) Middle stage. (c) Final stage.

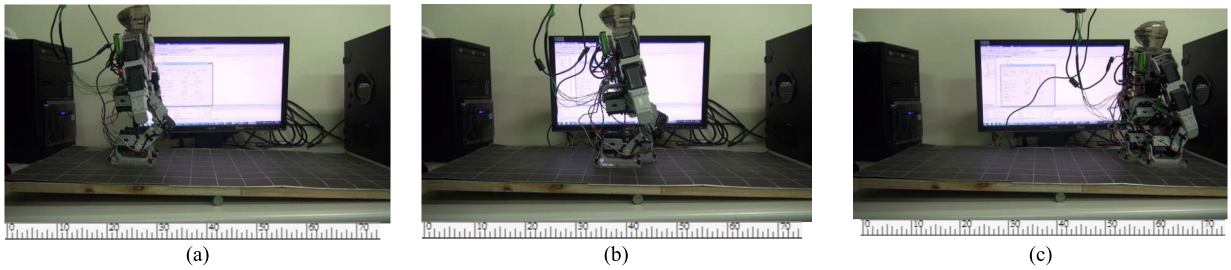


FIGURE 23. Experimental results of downhill. (a) Initial stage. (b) Middle stage. (c) Final stage.

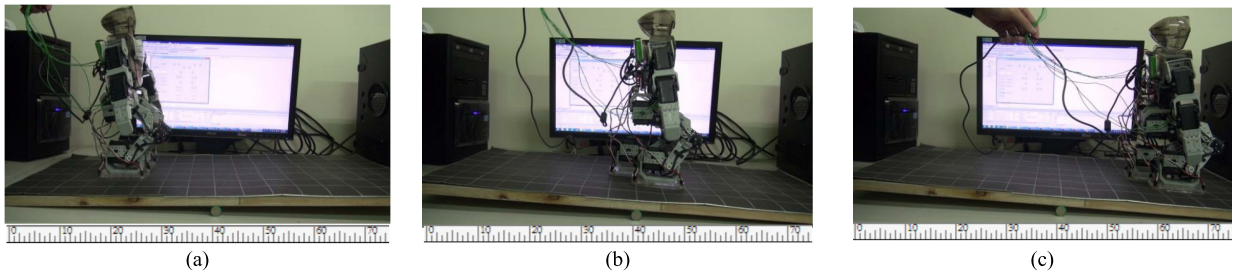


FIGURE 24. Experimental result of the seesaw. (a) Initial stage. (b) Middle stage. (c) Final stage.

middle stage of the tasks. Then the robot finishes the learning tasks as shown in Fig. 21(c), Fig. 22(c), and Fig. 23(c). After finishing the three tasks, the learning experiences which were collected in advance are merged to apply for the seesaw task and Fig. 24 shows the learning results. As shown in Fig. 24(a), the robot stays in the initial stage and the uphill experience can be implemented to walk on the seesaw. In the middle stage, the KTL switches the plain ground and downhill experience to walk on the seesaw, as shown in Fig. 24 (b). In the final stage, the downhill experience is used to finish the task. According to the experiment results, the robot can accomplish the goal. The video recording the scenarios, plain ground, uphill, downhill, and seesaw, as seen in the final results of the experiments, can be accessed at [23].

V. CONCLUSION

Controlling a biped robot walking in real time using the dynamic model of ZMP is complicated and difficult. The proposed Q-Learning architecture allows for a real-time control of robots. This learning method is based on the control of the ZMP's location acquired through force sensors placed on the soles. To relieve the loading in mapping the sensory receptions to the state space, a simple state aggregation method is applied in the architecture of the Q-Learning. In addition, the RL approach in this paper utilizes a simple Q-Learning architecture though it has the drawback suffering from a limited number of action options. To tackle this problem, the Continuous-Action Q-Learning is proposed. Based on the proposed method, a continuous action policy is constructed and shown to be more stable and robust after learning in a reasonable elapsed time. In the continuous policy, an action bias generates a smooth action to fit the requirements of applications. In learning processes, three experienced CAQs are combined to form the KTL. The proposed KTL architecture with three CAQs is used to complete a new task. To demonstrate the effectiveness of the proposed method, simulations and experiments on some learning tasks have been conducted. The simulation and experimental analysis demonstrate that it is possible for a robot to learn, without a dynamic model initially, walking gaits and improve its balance capability by the proposed method.

REFERENCES

[1] H. Lingyun and S. Zengqi, "Reinforcement learning method-based stable gait synthesis for biped robot," in *Proc. 8th Int. Conf. Control, Autom., Robot. Vis.*, vol. 2. Dec. 2004, pp. 1017–1022.

[2] K.-S. Hwang, J.-L. Lin, and K.-H. Yeh, "Learning to adjust and refine gait patterns for a biped robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 12, pp. 1481–1490, Dec. 2015.

[3] K.-S. Hwang, J.-L. Lin, T.-C. Huang, and H.-J. Hsu, "Humanoid robot gait imitation," in *Proc. SICE Annu. Conf. (SICE)*, Sep. 2014, pp. 2124–2128.

[4] T.-H. S. Li, Y.-T. Su, S.-W. Lai, and J.-J. Hu, "Walking motion generation, synthesis, and control for biped robot by using PGRL, LPI, and fuzzy logic," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 3, pp. 736–748, Jun. 2011.

[5] T. Suzuki, T. Tsuji, and K. Ohnishi, "Trajectory planning of biped robot for running motion," in *Proc. 32nd IEEE Conf. Ind. Electron. Soc.*, Nov. 2005, pp. 1815–1820.

[6] D. Tlalolini, C. Chevallereau, and Y. Aoustin, "Human-like walking: Optimal motion of a bipedal robot with toe-rotation motion," *IEEE/ASME Trans. Mechatronics*, vol. 16, no. 2, pp. 310–320, Apr. 2011.

[7] K.-C. Choi, H.-J. Lee, and M. C. Lee, "Fuzzy posture control for biped walking robot based on force sensor for ZMP," in *Proc. SICE-ICASE Int. Joint Conf.*, Oct. 2006, pp. 1185–1189.

[8] K. Suwanratchatamanee, M. Matsumoto, and S. Hashimoto, "Balance control of robot and human-robot interaction with haptic sensing foots," in *Proc. 2nd Conf. Human Syst. Interact. (HSI)*, May 2009, pp. 68–74.

[9] S. N. Napoleon, S. Nakaura, and M. Sampei, "Balance control analysis of humanoid robot based on ZMP feedback control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3. 2002, pp. 2437–2442.

[10] T. Sato, H. Ono, and K. Ohnishi, "Gyroscope integrated environmental mode compliance control for biped robot," in *Proc. IEEE Int. Workshop Adv. Motion Control*, Mar. 2012, pp. 1–6.

[11] L. Wang, Z. Liu, C. L. P. Chen, Y. Zhang, S. Lee, and X. Chen, "A UKF-based predictable SVR learning controller for biped walking," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 43, no. 6, pp. 1440–1450, Nov. 2011.

[12] Y. Yoshida, K. Takeuchi, Y. Miyamoto, D. Sato, and D. Nenchev, "Postural balance strategies in response to disturbances in the frontal plane and their implementation with a humanoid robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 6, pp. 692–704, Jun. 2014.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[14] T.-H. S. Li, P.-H. Kuo, Y.-F. Ho, M.-C. Kao, and L.-H. Tai, "A biped gait learning algorithm for humanoid robots based on environmental impact assessed artificial bee colony," *IEEE Access*, vol. 3, pp. 13–26, 2015.

[15] V. Gullapalli, "A stochastic reinforcement learning algorithm for learning real-valued functions," *Neural Netw.*, vol. 3, no. 6, pp. 671–692, 1990.

[16] V. Gullapalli, "Associative reinforcement learning of real-valued functions," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, vol. 3. Charlottesville, VA, USA, Oct. 1991, pp. 1453–1458.

[17] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Dept. Psychol., Univ. Cambridge, Cambridge, U.K., 1989.

[18] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, May 1992.

[19] C. F. Touzet, "Q-learning for robot," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 2003, pp. 934–937.

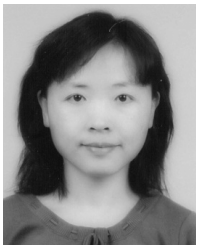
[20] K.-S. Hwang, H.-Y. Lin, Y.-P. Hsu, and H.-H. Yu, "Self-organizing state aggregation for architecture design of Q-learning," *Inf. Sci.*, vol. 181, no. 13, pp. 2813–2822, Jul. 2011.

[21] W. T. B. Uther and M. M. Veloso, "Tree based discretization for continuous state space reinforcement learning," in *Proc. 15th Nat. Conf. Artif. Intell. (AAAI)*, Madison, WI, USA, 1998, pp. 769–774.

[22] L. D. Pyeatt and A. E. Howe, "Decision tree function approximation in reinforcement learning," in *Proc. 3rd Int. Symp. Adapt. Syst., Evol. Comput. Probabilistic Graph. Models*, 1998, pp. 70–77.

[23] K.-S. Hwang, J.-L. Lin, and J.-S. Li, "Biped balance control by reinforcement learning," *J. Inf. Sci. Eng.*, to be published.

[24] IRIS. (Feb. 11, 2014). *Biped Robot Walk and Experience Transfer Demo. IRIS Lab.*, National Sun Yat-sen University, Kaohsiung, Taiwan, accessed Nov. 16, 2015. [Online]. Available: https://www.youtube.com/watch?v=mVahCHBFWyo



JIN-LING LIN received the master's and Ph.D. degrees in computer science from the University of Oklahoma, Norman, OK, USA, in 1989 and 1993, respectively. She is currently a Professor with the Department of Information Management, Shih Hsin University, Taipei, Taiwan. Her current research interests include object-oriented programming, data science, data mining, information retrieval, intelligent system, intelligent network routing, and path planning in logistics.

KAO-SHING HWANG (M'93–SM'09) is currently a Professor with the Electrical Engineering Department, National Sun Yat-sen University, and an Adjunct Professor with the Department of Healthcare Administration and Medical Informatic, Kaohsiung Medical University, Taiwan. He received the M.M.E. and Ph.D. degrees in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 1989 and 1993, respectively. He had been with National Chung Cheng University in Taiwan from 1993 to 2011. He was the Deputy Director of the Computer Center (1998-1999), the Chairman of the Electrical Engineering Department (2003-2006), and the Director of the Opti-Mechatronics Institute (2010-2011). He has been a fellow of the Institution of Engineering and Technology. His research interest includes methodologies and analysis for various intelligent robot systems, machine learning, embedded system design, and ASIC design for robotic applications.

WEI-CHENG JIANG received the B.S. degree in computer science and information engineering and the M.S. degree in electro-optical and materials science from National Formosa University, Yunlin, Taiwan, in 2007 and 2009, respectively, and the Ph.D. degree in electric engineering from National Chung Cheng University, Chiayi, Taiwan, in 2013. He holds a post-doctoral position with the Electrical Engineering Department, National Sun Yat-sen University, Kaohsiung, Taiwan. His research interests include machine learning, neural networks, and intelligent control.

YU-JEN CHEN received the B.S. degree in electrical engineering from the Tatung Institute of Technology, Taipei, Taiwan, in 1994, and the M.S. and Ph.D. degrees in electrical engineering from National Chung Cheng University, Chiayi, Taiwan, in 1997 and 2009, respectively. From 2004 to 2009, he was an Adjunct Lecturer with the Center for General Education, National Chung Cheng University. Since 2010, he has been an Assistant Professor with the Electrical Engineering Department, National Chung Cheng University. His current research interests include machine learning, robotics, neural networks, and embedded systems.