8

Stable Walking Pattern Generation for a Biped Robot Using Reinforcement Learning

Jungho Lee
Department of Mechanical Engineering, KAIST,
335 Gwahangno Yuseong-gu, Daejeon, 305-701, Republic of Korea
Phone: +82-42-869-5223, Fax: +82-42-869-8900
E-mail: jungho77@kaist.ac.kr

Jun Ho Oh
Department of Mechanical Engineering, KAIST,
335 Gwahangno Yuseong-gu, Daejeon, 305-701, Republic of Korea

Abstract

In this research, a stable biped walking pattern is generated by using reinforcement learning. The biped walking pattern for forward direction is chosen as a simple third order polynomial and sinusoidal function is used for sideway direction. To complete the forward walking pattern, four boundary conditions are needed. In order to avoid jerk motion, initial position and velocity and final position and velocity of the joint are selected as boundary conditions. Also desired motion or posture can be achieved by using the initial and final position. The final velocity of the walking pattern is related to the stability but it is hard to choose proper value. So the final velocity of the walking pattern is chosen as a learning parameter. In order to find the proper boundary condition value, a reinforcement learning algorithm is used. For the sideway movement, a sway amount is selected as learning parameter and a reinforcement learning agent finds proper value for sideway movement. To test the algorithm, a three-dimensional simulator that takes into consideration the whole model of the robot and the environment is developed. The algorithm is verified through a simulation.

Keywords: Biped walking; Reinforcement learning; Robot learning; Humanoid robot

1. Introduction

In various research fields involving humanoid robots, this research centers on the mobility. For a humanoid robot, its method of locomotion is critical. Numerous movement methods have been considered, including the use of wheels and caterpillar-type motion, quadruped motion, and hexapod walking methods inspired from the motions of animals and insects. While these methods have respective strengths, it is preferable that humanoid robots to be

used in human society be capable of biped motion using two legs, resembling a human, as our environment is geared to biped walking. By employing biped motion, the humanoid robot will be able to navigate stairs easily and walk along common walkways. In addition, it will be more familiar to humans.

The realization of biped walking is, however, relatively difficult because a biped walking robot is a highly complex system that is inherently unstable. The realization of biped walking started with the idea of static walking. The robot known as WABOT-1, which was developed by Waseda University in the 1970s, required 45 seconds per step but was the first biped walking robot that utilized the concept of static walking [23]. Static walking is characterized by slow biped movement, and the effect of linear or angular momentum of the robot is neglected.

Dynamic walking was considered after the general idea known as ZMP (Zero Moment Point) was introduced to a biped walking robot by Vukobratovic [1]. The ZMP is a point on the ground plane at which the total moments due to ground reaction force becomes zero. If the ZMP is located in a support region, the robot will never fall down when the robot is walking. The first robot to which the idea of ZMP was applied successfully was the WL-10 series from Waseda University. This robot can walk with 1.3 seconds between each step. After the appearance of the first dynamic biped walking robot, researchers developed a variety of working algorithms based on the ZMP.

Since the first successful presentation of biped walking at Waseda University in the 1970s and 1980s, there have been numerous trials to realize stable biped walking robustly and efficiently, with a number of notable results [2][3][4][5][6]. The humanoid robots developed by these research groups can walk steadily on flat or inclined ground and can even run [7][8][9]. Many notable algorithms developed for stable walking and the methods can be categorized into four paradigms [10]: (a) the use of passive walking as a starting point for the design of active walkers; (b) the use of the "zero moment point" control; (c) the use of fixed control architecture and application of a parameter search to find the parameter settings that yield successful walking gaits; and (d) the development of feedback laws based upon insights into balance and locomotion.

HUBO, the first humanoid robot in Korea, was developed by Oh et al. at KAIST in 2004 [2] [11][12][13]. It is a child-sized (125 cm tall) biped walking robot with 41 DOF (Degree Of Freedom). This humanoid robot combines several biped walking methods for stable walking. For the walking strategy, a walking pattern specific to a given environment is initially designed and a ZMP (Zero Moment Point) feedback controller and other sub-controllers are then used to maintain stability for a dynamically changeable environment. Many researchers use only a ZMP feedback controller. While stable walking can be maintained in this manner, however, it is difficult to generate desired motions. Hence, HUBO uses the aforementioned (b) and (c) paradigms to overcome this problem.

But the key challenge with the existing method used by HUBO is the determination of the proper parameters for designing or generating a stable walking pattern. It is difficult to find proper parameters as they are influenced by many factors such as the posture of the robot and the ground conditions. The existing robot HUBO determines these parameters through many experiments along with an analysis of walking data using a real system. This process is, however, difficult and time-consuming. Furthermore, only an expert can tune these parameters because an unconfirmed walking pattern is tested using a real robot, there is an inherent risk of accidents. This is the starting point of the present research.

In order to overcome these problems, a HUBO simulator and an algorithm that automatically determines an appropriate walking pattern were developed. The HUBO simulator describes the dynamics of the entire system using a physics engine and includes interactions between the robot and its environment, such as reaction forces and collision analysis. The function of this simulator is to test walking patterns. Also reinforcement learning is used to find suitable walking pattern parameters automatically in order to ensure stable walking and tested on this simulator. Reinforcement learning is a learning method that mimics the human learning process (i.e., learning from experience). Furthermore, this control method is usable if the information or the model of the given system is unclear. With the exception of reinforcement learning, many other methods such as those utilizing a neural oscillator, neural network or fuzzy logic can be used to solve this problem. However, these methods are complex compared to reinforcement learning and require an expert or reliable data. Thus, reinforcement learning is used for the generation of stable walking patterns in this study.

Earlier research on the subject of biped walking using reinforcement learning focused primarily on stable walking. However, the posture of a robot is as important as stable walking. For example, posture is particularly important when the robot is climbing stairs or walking across stepping stones. In these cases, foot placement by the robot is very important. Each foot should be placed precisely or the robot can collapse. Thus, the main goal of this research is to determine a walking pattern that satisfies both stable walking and the required posture (foot placement) using reinforcement learning. Particularly, the Q-leaning algorithm is used as the learning method and CMAC (Cerebellar Model Articulation Controller) serves as the generalization method. The Q-learning algorithm is easy to implement and its convergence is not affected by the learning policy. Hence, it has been used in many applications.

2. Related work

Former studies concerning the realization of stable biped walking using reinforcement learning are categorized below.

(a) The use of reinforcement learning as a sub-controller to support the main controller
(b) The use of reinforcement learning as a main controller or a reference generator
In Case (a), reinforcement learning is normally used as a gain tuner of the main controller

or as a peripheral controller for stable walking. In Case (b), reinforcement learning is used directly to generate a stable walking pattern or as the main controller for stable walking.

Chew and Pratt [14][54] simulated their biped walking robot, Spring Flamingo (Fig. 2-1), in the planar plane (two-dimensional simulation). This seven-link planar bipedal robot weighed 12 kg, was 0.88m in height and had bird-like legs. A reinforcement leaning system was used as the main controller. The following states were chosen as follows: (a) velocity of the hip in the forward direction (x-coordinate); (b) the x-coordinate of an earlier swing ankle measured with reference to the hip; and (c) the step length. The goal was to enable the robot to walk with constant speed; thus, the learning system received '0' when the robot walked within the boundary speed or was given a negative value as a reward. The position of the swing foot was used as the action. Additionally, a torque controller in each ankle was used to control the ankle joint torque. The same type of torque controller was also used to maintain the velocity of the body. The ankle joint torque was limited to a certain stable value; hence, the robot could walk stably without considering the ZMP. However, because the goal was to realize walking with constant speed, the posture of the robot was not considered.

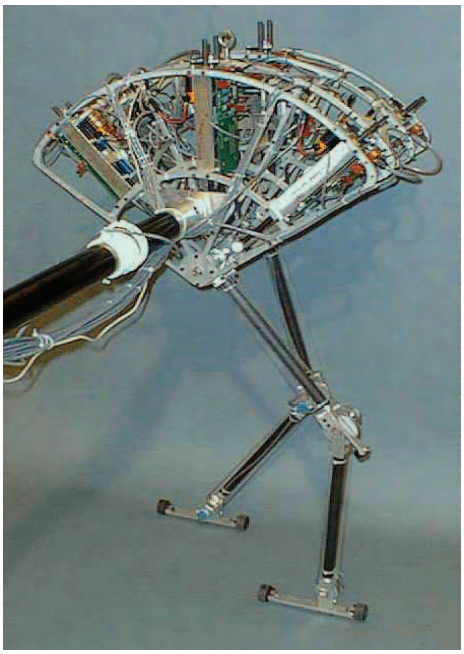

Fig. 1. Spring Flamingo

Benbrahim and Franklin [15] used a reinforcement learning system as both the main and sub-controllers. To achieve dynamic walking with their planar robot, central and other

peripheral controllers were used. The central controller used the experience of the peripheral controllers to learn an average control policy. Using several peripheral controllers, it was possible to generate various stable walking patterns. The main controller activated specific peripheral controllers in an approach that was suitable for specific situations. However, the architecture of the controller was complex, and this approach required many learning trials and a lengthy convergence time.

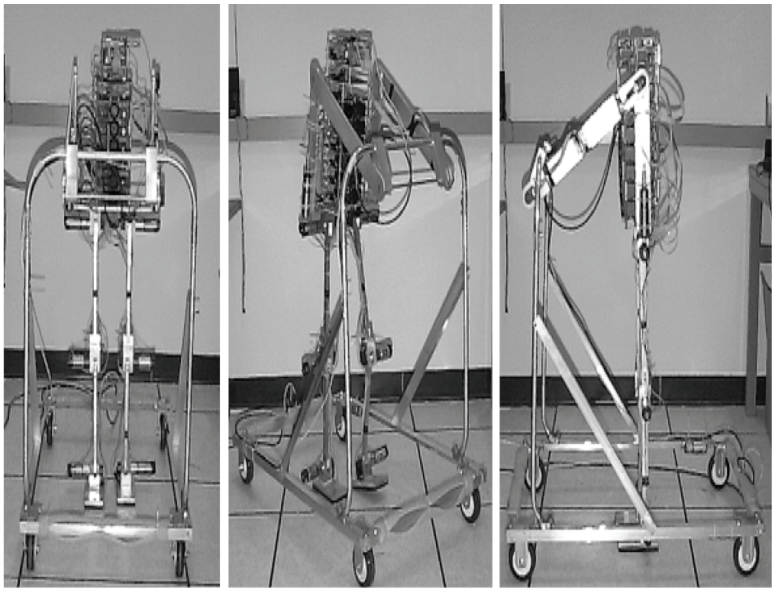

Fig. 2. Benbrahim's biped planar walking robot

Morimoto, Cheng, Atkeson, and Zeglin [16][60] (Fig. 2-3) used a simple five-link planar biped robot to test their reinforcement learning algorithm. The foot of each leg had a shape resembling a 'U', and no joints were used in the ankle. Consequently, it moved in the manner of a passive walker. The goal of the learning system was to walk with constant speed and the states were as follows: (a) velocity of the hip in the forward direction; and (b) forward direction distance between the hip and ankle. The reward was simply falling down or remaining upright, and the action was the angle of the knee joint. The hip joint trajectory was fixed but the step period could vary. If the swing leg touched the ground before the current step period, the next step period was decreased. In addition, if the swing leg touched the ground after the current step period, the step period was increased. This work concentrated only on stable walking; the posture of the robot was not considered.

Schuitema et al. [17] also used reinforcement learning to simulate their planar robot. Their robot, termed Meta, is a passive dynamic walking robot with two hip active joints. The goal of their learning system was to have the robot walk successfully for more than 16 steps.

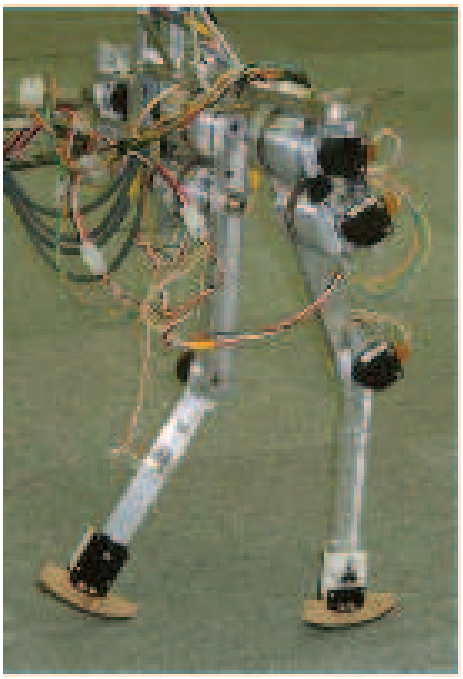

Fig. 3. Morimoto, Cheng, Atkeson, and Zeglin's five-link planar robot

The state space consisted of six dimensions: the angle and angular velocity of the upper stance leg, the upper swing leg, and the lower swing leg. To avoid conditions in which the system would not learn the same thing twice, symmetry between the left and right leg was implemented by mirroring left and right leg state information when the stance leg changed. There was one action dimension, the torque that was applied to the hip joint, which was given a range between -8 and 8 Nm. If the robot walked forward successfully, the learning system received a reward. Additionally, if the body of the robot moves backward, the learning system was penalized. Various experiments were simulated under various constraints and compared the results of each experiment.

Kim et al. [18] used a reinforcement learning system for ZMP compensation. In their research, two-mode Q-learning was used as ZMP compensation against the external distribution in a standing posture. The performance of the Q-learning system was improved using the failure experience of the learning system more effectively along with successful experiences. The roll angle and the roll angular velocity of the ankle were selected as the states. For the action, ankle rolling was given three discrete levels (±0.5°, 0°) during a period of 20 ms. If selecting an action in the opposite direction of the external force, the agent received a reward. If the angle and ZMP constraints were exceeded, the agent was penalized.

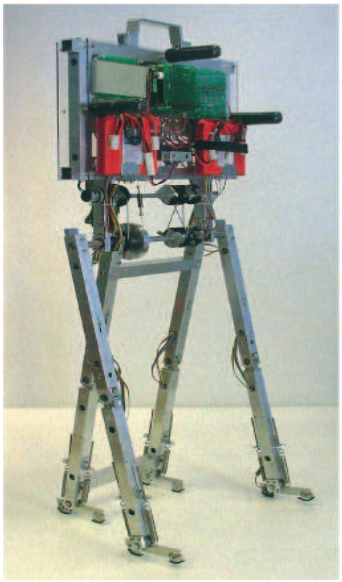

Fig. 4. Meta passive dynamic walking robot

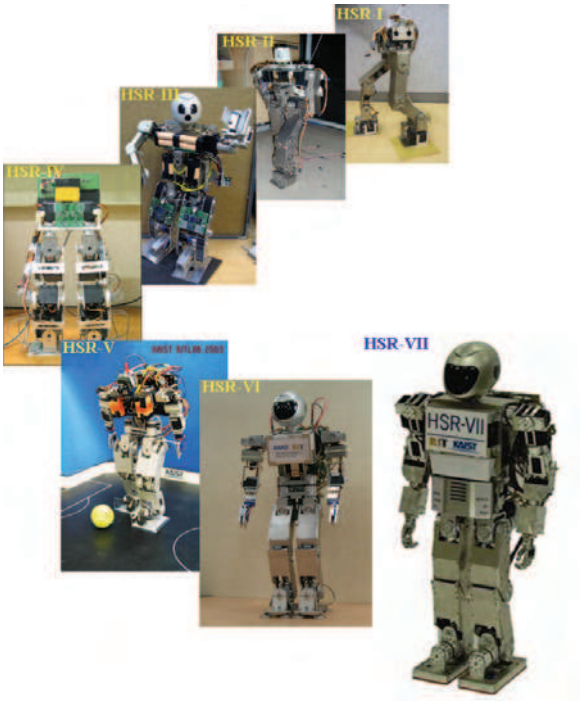

Fig. 5. HanSaRam Series

Other researchers have also proposed learning approaches for stable biped walking [19][20][21]. However, existing research in this area in which reinforcement learning is utilized concerns only stable walking while the posture of the robot is not considered. These researchers normally utilize the position of the swing leg for stable walking. It is, however, difficult to locate the foot in a desired position using only the swing leg. Moreover, it was necessary to use another controller or additional devices for the motion of the support leg. Therefore, this research focuses on the control of the support leg as opposed to control of the swing leg for stable and desired motion walking.

3. Walking pattern

3.1 Sagittal plane

There are several methods of designing a stable walking pattern. But recent researches can be categorized into two groups [22]. The first approach is the 'inverted pendulum model control method' [46][47]. In this method, a simple inverted pendulum model is used as a biped walking model. Based on this model, a proper ZMP reference is generated and a ZMP feedback controller is designed to follow this reference. As this method uses a simple inverted pendulum model, its control structure is very simple. Furthermore, because it follows the ZMP reference for stable walking, stability is always guaranteed. However, it requires a proper ZMP reference and it is difficult to define the relationship between the ZMP reference and the posture of the biped walking robot clearly and accurately. Therefore, it is difficult to select the proper ZMP reference if the posture of the biped walking robot and its walking stability is important. A pattern generator, which translates the ZMP reference to a walking pattern, is also required. Fig. 3-1 shows a block diagram of the 'inverted pendulum model control method'.

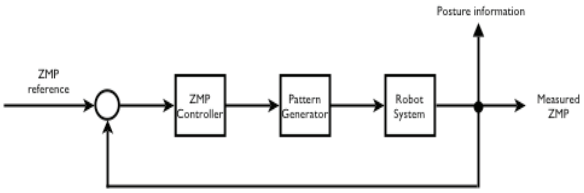

Fig. 6. Inverted pendulum model control method

A second method is known as the 'accuracy model method' [44][45][48][49]. This model requires an accurate model of the biped walking robot and its environment. In this method, a stable walking pattern is generated in advance based on the abovementioned accurate model and the biped walking robot follows this walking pattern without a ZMP feedback controller. One advantage of this method is that it allows control of the biped walking robot with a desired posture. Additionally, it does not require a ZMP controller. However, the generated walking pattern is not a generally functional walking pattern. For example, the walking pattern that is generated for flat ground is not suitable for inclined ground.

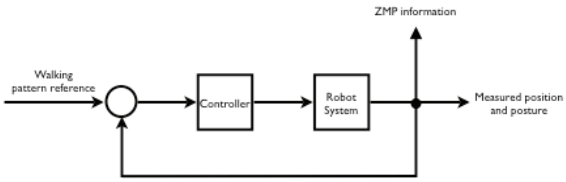

Fig. 7. Accuracy model method

Therefore, with different conditions (e.g., different ground conditions, step lengths, or step periods), new walking patterns should be generated. Fig. 3-2 shows the process of the 'accuracy model method'.

Compared to the 'inverted pendulum model control method', the 'accuracy model method' does not guarantee stability against disturbances; however, it has its own strengths. First, it is possible to control the motion of the biped robot using this method. The 'inverted pendulum model control method' only guarantees stability if the ZMP reference is correct. And it is not possible to control the motion. Second, this method is more intuitive compared to the 'inverted pendulum model control method'. Thus, it is easy to imply physical intuitions using this method. Third, a ZMP controller is not required. Hence, the overall control architecture is simpler with this method compared to the 'inverted pendulum model control method'.

However, an additional problem with the 'accuracy model method' involves difficulty in obtaining an accurate model of the robot and its environment, including such factors as the influence of the posture of the robot, the reaction force from the ground, and so on. Consequently, the generated walking pattern should be tuned by experiments. The generated walking pattern for a specific environment is sensitive to external forces, as this method does not include a ZMP controller. However, when the precise posture of the biped walking robot is required, for example, when moving upstairs or through a doorsill, the 'accuracy model method' is very powerful [9].

In an effort to address the aforementioned issues, the algorithm generating walking patterns based on the 'accuracy model method' was developed using reinforcement learning. To generate a walking pattern, initially, the structure of the walking pattern should be carefully selected. Selection of the type of structure is made based on such factors as polynomial equations and sine curves according to the requirements. The structure of the walking pattern is selected based on the following four considerations [24].

(a) The robot must be easy to operate. There should be minimal input from the operator in terms of the step time, stride, and mode (e.g. forward/backward, left/right) as well as commands such as start and stop.
(b) The walking patterns must have a simple form, must be smooth, and must have a continuum property. It is important that the walking patterns be clear and simple. The

trajectory of the walking patterns should have a simple analytic form and should be differentiable due to the velocity continuum. After the walking patterns are formulated, the parameters for every step are updated.
(c) The calculation must be easy to implement in an actual system. The calculation burden and memory usage should be small and the pattern modification process should be flexible.
(d) The number of factors and parameters that are to be tuned must be small. The complexity of the learning process for the walking patterns is increased exponentially as the number of factors and parameters is increased.

In this research, based on these considerations, a third-order polynomial pattern for the support leg was designed as the walking pattern. This pattern starts from the moment one foot touches the ground and ends the moment the other foot touches the ground (Fig. 3-3).

$$\begin{aligned} z(t) &= Z \\ x(t) &= at^3 + bt^2 + ct + d \end{aligned} \tag{1}$$

To create or complete the third-order forward walking pattern, as shown in Eq. 3-1, four boundary conditions are needed. These boundary conditions were chosen with a number of factors taken into account. First, to avoid jerking motions and formulate a smooth walking pattern, the walking pattern must be continuous. For this reason, the position and velocity of the hip at the moment of the beginning of the walking pattern for the support leg were chosen as the boundary conditions. Additionally, when the foot of the robot is to be placed in a specific location, for example traversing uneven terrain or walking across stepping stones, the final position of the walking pattern is important. This final position is related to the desired posture or step length, and this value is defined by the user. Hence, the final position of the hip can be an additional boundary condition. Lastly, the final velocity of the walking pattern is utilized as the boundary condition. Using this final velocity, it is possible to modify the walking pattern shape without changing the final position, enabling the stabilization of the walking pattern [24]. From these four boundary conditions, a third-order polynomial walking pattern can be generated.

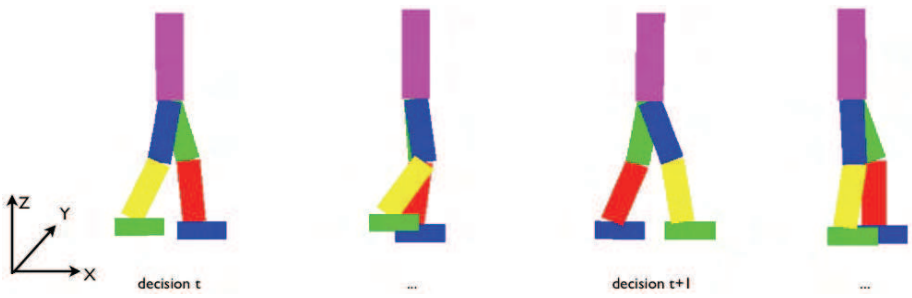

Fig. 8. Sequence of walking

However, it is difficult to choose the correct final velocity of the pattern, as exact models include the biped robot, ground and other environmental factors, are unknown. The existing HUBO robot uses a trial-and-error method to determine the proper final velocity parameter, but numerous trials and experiments are required to tune the final velocity. Thus, in order to find a proper value for this parameter, a reinforcement leaning algorithm is used.

Table 3-1 summarizes the parameters for the sagittal plane motion. And to make problem simpler z-direction movement is fixed as Z (Eq. 3-1).

Boundary condition	Reason
Initial velocity	To avoid jerk motion
Initial position	To avoid jerk motion and continuous motion
Final position	To make wanted posture
Final velocity	To make the walking pattern stable (Unkown parameter)

Table 1. Boundary conditions for the walking pattern

3.2 Coronal plane

Coronal plane movements are periodic motions; it the overall movement range of these movements is smaller than the sagittal plane motion, a simple sine curve is used. If the movement of the z direction is constant, the coronal plane motion can be described by Eq. 3-2, where Y is the sway amount and w is the step period.

$$\begin{aligned} z(t) &= Z \\ y(t) &= Y\sin(wt) \end{aligned} \tag{2}$$

From the simple inverted pendulum model, the ZMP equation can be approximated using Eq. 3-3, where l denotes the length from the ankle joint of the support leg to the mass center of the robot.

$$ZMP(t) = y(t) - \frac{l}{g}\ddot{y}(t) \tag{3}$$

From Eq. 3-2 and Eq. 3-3, the ZMP can be expressed using Eq. 3-4

$$ZMP(t) = Y(1 + \frac{l}{g} w^2) \sin(wt) \tag{4}$$

The length l and the step period w are given parameters and the acceleration of gravity g is known parameter. The only unknown parameter is the sway amount. The sway amount can be determined by considering the step period, the DSP (Double Support Phase) ratio and the support region. If the amplitude of the ZMP is located within the support region, the robot is stable. It is relatively easy to determine the unknown parameter (the sway amount) compared to the sagittal plane motion. However, it is unclear as to which parameter value is most suitable. The ZMP model is simplified and linearized, and no ZMP controller is used in this research. Thus, an incorrect parameter value may result from the analysis.

Therefore, using the reinforcement learning system, the optimal parameter value for stable walking using only low levels of energy can be determined.

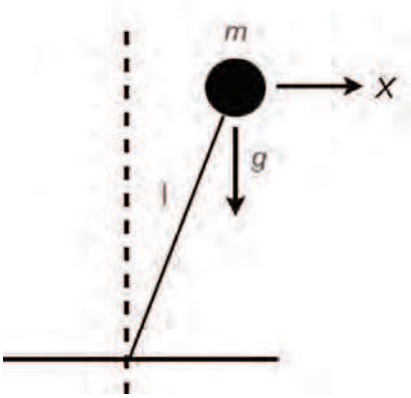

Fig. 9. Inverted pendulum model

4. Simulation

4.1 Simulator

4.1.1 Introduction

Reinforcement learning is based on trial-and-error methodology. It can be hazardous to apply a reinforcement learning system to an actual biped walking system before the learning system is trained sufficiently through many trials, as walking system likely has not been fully analyzed by the learning system. In particular, when such a system is inherently unstable, such as in the case of a biped walking robot, attention to detail is essential. Therefore, it is necessary to train a learning system sufficiently before applying it to a real system. For this reason, simulators are typically used.

A simulator can be used purposes other than for the training of a learning system. For example, simulators can be used for testing new control algorithms or new walking patterns. Various research groups investigating biped walking systems have developed simulators for their own purposes [32][33][34][35][36][37].

The HUBO simulator, which was developed for this study, is composed of a learning system that is in charge of all leaning processes, a physics engine that models a biped robot and its environment, and utility functions to validate the simulation results. Fig. 4-1 shows these modules and the relationships between them. As shown in the figure, learning contents or data obtained from the reinforcement learning module are stored through generalization process. In this study, the CMAC algorithm is used as the generalization method; however, other generalization methods can be easily adapted. The dynamics module, which contains a physics engine, informs the reinforcement learning module of the current states of HUBO. It also receives the action (final velocity of the walking pattern and the sway amount) from the reinforcement learning module, generates a walking pattern, and returns a reward. For the visualization of the movement of a biped walking robot, the OpenGL library is used. Because all components of the HUBO simulator are modularized, it is easy to use with new algorithms or components without modification.

The HUBO simulator contains all of the components necessary for simulating and testing biped walking systems and control algorithms. In addition, all modules are open and can be modified and distributed without limitation. The HUBO simulator follows the GPL (GNU General Public License) scheme.

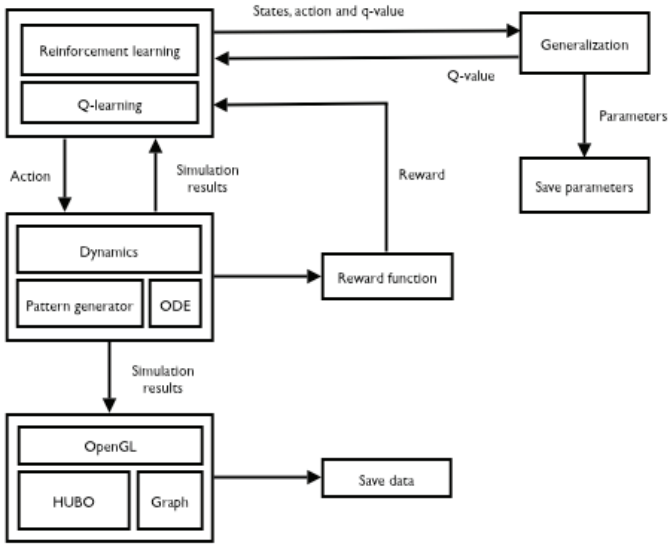

Fig. 10. Structure of the HUBO simulator

4.1.2 Physics Engine

To obtain viable simulation results, the dynamics model of a simulator is very important. If the dynamics model differs greatly from a real model, the result of the simulator is useless. Therefore, it is important to ensure that the simulation model resembles the actual model to the greatest extent possible. Essentially, the model of a biped walking system should contain a robot model as well as a model of the environment of the robot. Many researchers only consider the robot model itself and neglect a model of the environment, which is in actuality more important in a realistic simulation of a biped walking.

For this reason, a physics engine was used to build realistic dynamic model in this study. A physics engine is a tool or APIs (Application Program Interface) that is used for computer simulation programs. In this research, ODE (Open Dynamics Engine) [38] was used to develop the robot and environmental model in an effort to represent the actual condition of the robot accurately. ODE is a rigid body physics engine initially developed by Russell Smith. Its source code is open and is governed by the open source community. ODE provides libraries for dynamics analyses, including collision analyses. The performance of ODE has been validated by various research groups [37][39][40], and many commercial and engineering programs use ODE as a physics engine.

4.1.3 Learning System

The learning system of the HUBO simulator consists of a learning module and a generalization module. The reinforcement learning module uses the Q-learning algorithm, which uses the Q-value. To store the various Q-values that represent actual experience or trained data, generalization methods are needed. Various generalization methods can be used for this. In the present work, the CMAC (Cerebella Model Articulation Controller) algorithm is employed. This algorithm converges quickly and is readily applicable to real systems.

Setting up states and a reward function is the most important process in the efficient use of reinforcement learning. When setting up states, using physical meanings is optional; however, it is important that the most suitable states for achieving the goal are selected. Additionally, the reward function should describe the goal in order to ensure success. The reward function can represent the goal directly or indirectly. For example, if the goal for a biped walking robot is to walk stably, the learning agent receives the reward directly if the robot walks stably without falling down. Otherwise, it is penalized. In addition, the reward function describes the goal of stable walking indirectly, including such factors as the pitch or roll angle of the torso while walking and the walking speed. However, it is important that the reward should suitably describe the goal.

4.1.4 Layout

Fig. 4-2 shows the main window of the HUBO simulator. The motion of HUBO calculated using ODE is displayed in the center region of the HUBO simulator using OpenGL. Each step size or foot placement can be modified from the main window. Fig. 4-3 shows the

learning information window. This window shows information such as the current states and the reward associated with the learning module. In addition, the learning rate and the update rate can be modified from this window. Fig. 4-4 shows the body data window. This window shows the current position and orientation of each body. As lower body data is important for the system, only the data of the lower body is represented. Fig. 4-5 shows the joint angle of the lower body. The data of the force and torque for each ankle joint is shown in the force-torque data window in Fig. 4-6.

The HUBO simulator was developed using the COCOA[○,R] library under a Mac OS X [○,R] environment. As COCOA is based on the Object-C language and all structures are modulated, it is easy to translate to other platforms such as Linux[○,R] and Windows[○,R].

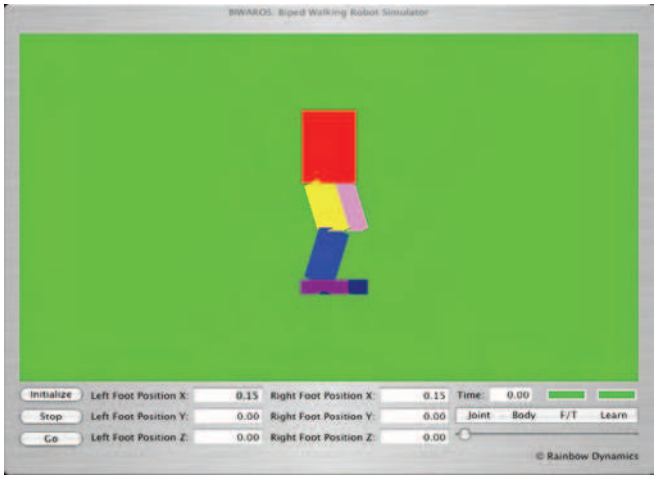

Fig. 11. Main window of the HUBO simulator

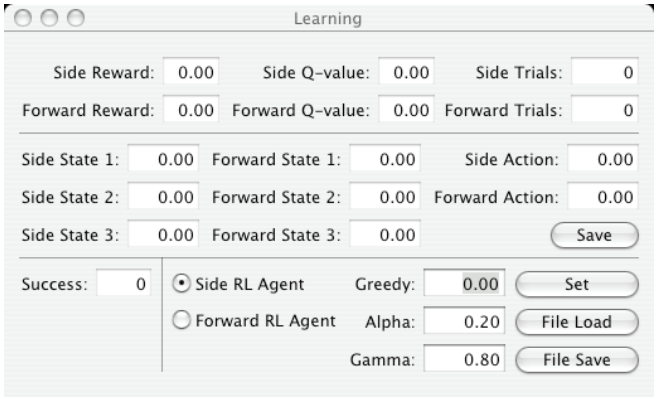

Fig. 12. Learning information window of the HUBO simulation

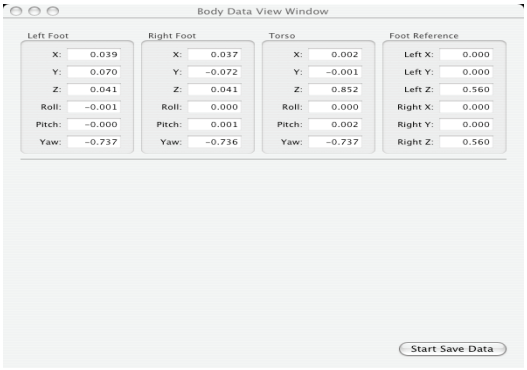

Fig. 13. Body data window of the HUBO simulator

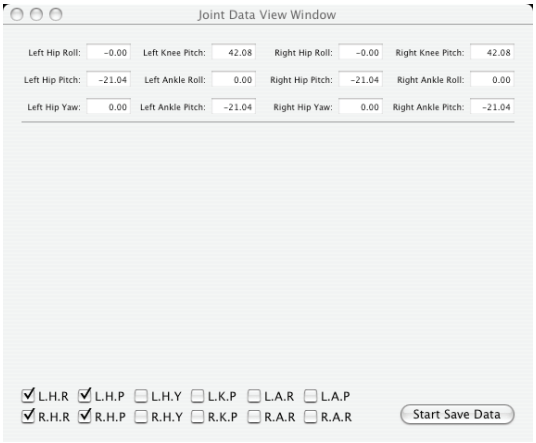

Fig. 14. Joint data window of the HUBO simulator

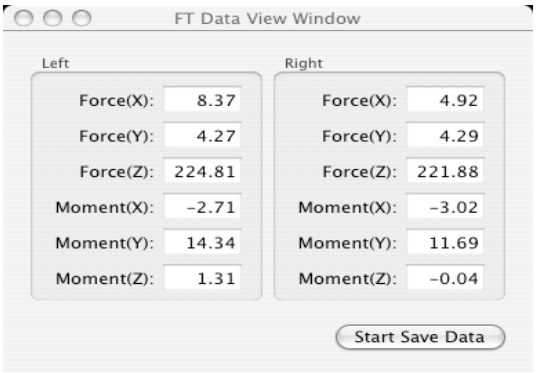

Fig. 15. Force-Torque data window of the HUBO simulator

4.2 States, action and reward

The biped walking pattern generation system can be viewed as a discrete system. Before a new walking step begins, the learning module receives the information of the current states and generates the walking pattern. The robot then follows the generated walking pattern. Following this, the walking pattern is finished and the process starts again. Therefore, this system can be viewed as a discrete system in which the time step is the walking pattern period or the step period. In this study, the walking pattern starts at the moment of the SSP (Single Support Phase) and ends at the moment of the next SSP. At the beginning of the SSP, the learning module receives the current states and calculates the action. Simultaneously, an evaluation of the former action is carried out by the learning module.

4.2.1 Sagittal plane

To set up proper states for the sagittal plane motion, a simple inverted model is used. From the linearized inverted pendulum model, the ZMP equation can be formulated, as shown in Eq. 4-1.

$$ZMP(t) = x(t) - \frac{l}{g}\ddot{x}(t) \tag{5}$$

From Eq. 4-1, the position and acceleration of the mass center is directly related to the ZMP. As the ZMP is related to the stability of the biped walking system, it is feasible to select the position and acceleration of the mass center as states. In addition, to walk stably with minimal energy consumption, the robot should preserve energy, implying that the robot should utilize its momentum (angular or linear). The momentum reflects current and future states; it is related to the velocity of the mass center. Therefore, the velocity of the mass center was chosen as the state in this study. Selected states and the reasons for their selection are summarized in Table 4-1.

All states are normalized to -1.0 ~ 1.0. However, the reinforcement learning agent has no data regarding the maximum values of the states. It receives this data during the training and updates it automatically. First, these maximum values are set to be sufficiently small; in this research, the value is 0.1. The reinforcement learning agent then updates the maximum value at every step if the current values are larger than the maximum values.

State	Reason
The position of the mass center with respect to the support foot	Relation between the position of the mass center and ZMP and the body posture
The velocity of the mass center	Angular or linear momentum
The acceleration of the mass center	Relation between the position of the mass center and ZMP

Table 2. States for the sagittal plane motion

The learning parameter learnt through reinforcement learning is the final velocity. It is an