
Yu-Te Su, Kiah-Yang Chong, and Tzuu-Hseng S. Li

Abstract

The design and implementation of Fuzzy Policy Gradient Learning (FPGL) method for humanoid robot is proposed in this paper. This paper not only introduces the phases of the humanoid robot walking, but also improves and parameterizes the gait pattern of the robot. FPGL is an integrated machine learning method based on Policy Gradient Reinforcement Learning (PGRL) and fuzzy logic concept in order to improve the efficiency and speed of gait learning computation. The result of the experiment shows that FPGL method can train the gait pattern from 9.26 mm/s walking speed to 162.27 mm/s within an hour. The training data of experiments also shows that this method could improve the efficiency of basic PGRL method up to 13%. The effect of arm movement to reduce the tilt of the trunk is also proved by the experimental results. All the results successfully demonstrate the feasibility and the flexibility of the proposed method.

Keywords: Fuzzy logic control, FPGL, Humanoid robot, PGRL.

1. Introduction

Bill Gate, the leader of the PC revolution, predicted that robotics will be the next hot field in the world in Scientific American January 2007 [1], which presented that robotics is one of the most researchable fields in present days. In fact, robots has been developed and functioned well in the industry for decades, but they became more related to our life time recently. Moreover, there are many kinds of robots that have been developed, and one of the most exciting types of robots is the humanoid robots, because they have similar structure as human beings and have the potential to be adopted in the human environment. Many small humanoid robot kits, such as KHR [2], Robonova [3], Bioloid [4], Nao [5], and others, can be purchased in the market easily, and be used for entertaining and educating.

Research about walking of humanoid robots has been developed to reach the human walking efficiency, stability and flexibility for years [6, 7]. Many concepts are developed and used in walking of humanoid robots, such as zero moment point (ZMP) [8], center of pressure (COP) [9], gait reference generation [10], dynamic walking control [11], biological inspiration [12] and etc. Furthermore, dynamic model of the biped robot is needed in most of these theories, and the model of a robot with many joints is too complex to be realized in real time. On the other hand, most of the data of a biped robot, such as ZMP and center of mass (COM), is hard to be measured and computed accurately by the sensors on small humanoid robot. Therefore, machine learning methods are the new choice for applied in learning of biped walking without any complex model.

Generating a fast and stable walking pattern is the most difficult task in the implementation of a humanoid robot, so applying intelligent learning in the system can let the robot learns walking pattern easily and efficiently. There are lots of theories of intelligent learning, such as Reinforcement Learning [13-15], Q-learning [16], Neural Network [17], and Genetic Algorithm [18], that can be used in the learning of walking pattern. In this paper policy gradient reinforcement learning (PGRL) method is chosen to be the basic learning method to generate a better walking pattern by the data that can be measured of the sensors on the robot. Furthermore, fuzzy logic is integrated with PGRL to generate a new learning method, which is called fuzzy policy gradient learning (FPGL) method to learn the fastest possible gait in real-time and less amount of experiments. Besides, the arm movement of a humanoid robot in the walking pattern can also be trained with PGRL method to reduce the tilt of trunk and improve the stability of the robot.

This paper is organized as follows. In Section II, the design of a new walking gait and the parameterization of designed gait pattern are introduced. The contents of basic PGRL and the proposed FPGL for biped robot walking motion generation are described in Section III. In Section IV, the human-machine interface and the experiments of gait learning are presented to illustrate the effectiveness of the proposed control scheme for a real biped robot. Conclusions are drawn in Section V.

Corresponding Author: Tzuu-Hseng S. Li is with the aiRobots Laboratory, Department of Electrical Engineering, National Cheng Kung University, 1 University Road, Tainan, Taiwan 701, R.O.C.
E-mail: thsli@mail.ncku.edu.tw
Manuscript received 18 March 2011; revised 18 Aug. 2011; accepted 2 Nov. 2011.
2. Design and Parameterization of Gait pattern

In this paper, we show the performance and effectiveness of the proposed learning method through the humanoid robot, “aiRobots-V,” as shown in Fig. 1. The design concept of proposed gait pattern is presented firstly. Then, the parameterization of designed gait pattern is depicted.

2.1 Concept of Gait pattern

The legs of human beings and other creatures in the world are much lighter, handier and tinier, comparing to their body, so they are able to move quickly. The muscle on their legs also causes them can walk, run, jump and do many actions in different environment. In contrast with the legs of human beings, the size, weight and strength of the actuators of a humanoid robot make it difficult to mimic the action of human beings, especially to the small-sized humanoid robots. During designing a small-sized humanoid robot, the size and position of actuators that can provide enough torque and speed are the main concern. The design and scale of the structure of the legs is not always similar to the legs of human beings, making some models or theories which are based on biology or physiology of human beings are hard to apply on the humanoid robots. Since the humanoid robot has been academically concerned for years, many methods has been applied on the gait pattern of humanoid robot, such as ZMP [8], COP [9], gait reference generation [10], dynamic walking control [11], biological inspiration [12] and etc.

Fig. 2 shows the improved gait pattern. Comparing to the gait pattern in [19], there are no swinging the trunk in the lateral plane. The robot will start with standstill stepping motion, until the robot is moving in a stable oscillation, then the walking cycle is started and repeated until the robot reaches the target. In the walking cycle, there are four states. In each state, the legs are in different pose, and there are also four poses for each leg. Fig. 3 shows the four poses of right leg in the walking cycle, named lift pose, step pose, hold pose and push pose. Lift pose is the leg lifting from the ground. Step pose is the leg stepping forward. Hold pose is the leg

Among the previous researches in the gait design of humanoid robot, ZMP [8] is the most popular theory that has been applied for generating a stable gait pattern of robots. The ZMP is the point with respect to which dynamic reaction force at the contact of the foot with the ground does not produce any moment, meaning that the resultant of all the moments of gravity and inertia forces at that point equals zero. A ZMP located in the supporting polygon of the foot on the ground makes the robot in a stable pose. If the ZMP in every moment during walking is in the supporting polygon, the robot can walk very stable. By the way, a ZMP is not easy to be measured by available sensors, therefore some researchers has found out that most of the time, ZMP and COP [9] of a robot are the same point, and COP can be measured by force sensors placed on the foot. The COP is the point on a body, where the total sum of the aerodynamic pressure field acts and causes a force and no moment about that point. Even though ZMP and COP can provide a stable gait pattern, but with some limits of hardware, these theories can only generates a stable gait pattern with slow speed. Furthermore, the gait generator using ZMP or COP needs to calculate torque and position of all actuators by inverse kinematic transform and takes long process time to generate a stable gait with a micro-processor. In order to learn with an effective learning method, the gait pattern must be design and parameterized first.
holding the body of robot while another leg is lifting. Push pose is the leg pushing the body of robot forward while another leg is stepping forward. Therefore, right leg is in lift pose and left leg is in hold pose in state 1. Then, right leg is in step pose and left leg is in push pose in state 2. In state 3, right leg is in hold pose and left leg is in lift pose. Finally, right leg is in push pose and left leg is in step pose in state 4. In this paper, the time between two of the states is set to 0.12s, so total time for a walking cycle is 0.48s. The time scale of the walking cycle is shown in Fig. 4.

Furthermore, in order to reduce the danger of falling down when step length is too large and the speed of robot is too fast, a speed up motion is added before the robot starts the walking cycle. A speed down motion is also added after the end of the walking cycle to slow down the robot gently and recover it to initial pose. The whole cycle of the gait pattern is shown in Fig. 5.

Besides, the tilt of trunk is very hard to be avoided while the robot is walking, and this tilt may cause large error in the distance calculation of target by the vision system which assumes that the height and tilt of trunk are constant. In order to reduce the influence of the tilt difference of trunk in the gait, the arms have to swing synchronously with the legs. This arm movement is just like our arm when human is walking. As shown in Fig. 6, the right arm starts to swing backward just after the right leg reaches the highest position when lifting and starts stepping in state 1. The right arm stops swinging backward when the right leg steps on the ground in state 2. Then the right arm swings forward in state 3 and stops when right leg finished pushing the body of robot forward in state 4. This angular momentum induced by the arms can be used to cancel some undesired momentum generated by leg around Y axis.

### 2.2 Parameters of Gait Pattern for machine learning method

In order to learn the faster gait pattern with machine learning algorithm, the gait pattern should be parameterized firstly. In the previous subsection, the four states and four poses of the gait pattern are defined; therefore those angular positions of the actuators in the walking cycle are the parameters of gait pattern to be learned with the machine learning.

In the sagittal plane, there are three angle data for each pose for each leg, which are the angle of hip, knee and ankle, as shown in Fig. 7. Those angles of a pose of gait pattern in sagittal plane can be reduced to two by the equation (1). This reduction of parameters can speed up the leaning process and reduce the number of experiments during the gait learning.

$$\theta_{\text{Knee}} = \theta_{\text{Hip}} + \theta_{\text{Ankle}}$$  \hspace{1cm} (1)

The equation (1) not only reduces the number of parameters for learning, but also can make the trunk of robot parallel to the soles and ground in each state. Since
there are four poses in the walking cycle, the number of leg parameters for gait learning is totally eight. The set of parameters is defined as 
\[ \{ \theta_{HL}, \theta_{AL}, \theta_{HS}, \theta_{AS}, \theta_{HH}, \theta_{AH}, \theta_{HP}, \theta_{AP} \}, \]
where \( \theta_{HL} \) and \( \theta_{AL} \) are the hip and ankle angle in the lift pose, \( \theta_{HS} \) and \( \theta_{AS} \) are the hip and ankle angle in the step pose, \( \theta_{HH} \) and \( \theta_{AH} \) are the hip and ankle angle in the hold pose, and finally \( \theta_{HP} \) and \( \theta_{AP} \) are the hip and ankle angle in the push pose. Fig. 8 illustrates all the parameters of a leg in gait learning and some constraints should be defined for those parameters to fasten the process of gait learning. The learning parameters of legs in the designed gait pattern and the constraints of those parameters are defined as
\[
X_{leg} = \begin{bmatrix}
\theta_{HL} \\
\theta_{AL} \\
\theta_{HS} \\
\theta_{AS} \\
\theta_{HH} \\
\theta_{AH} \\
\theta_{HP} \\
\theta_{AP}
\end{bmatrix} \in \mathbb{R}^8, \text{ for } \begin{cases}
\theta_{HL} \in (0^\circ, 90^\circ) \\
\theta_{AL} \in (0^\circ, 90^\circ) \\
\theta_{HS} \in \theta_{HL}, (90^\circ) \\
\theta_{AS} \in (0^\circ, 180^\circ) \\
\theta_{HH} \in (0^\circ, 180^\circ) \\
\theta_{AH} \in (0^\circ, \theta_{AS}) \\
\theta_{HP} \in (90^\circ, 180^\circ) \cap [\theta_{HH}, 180^\circ) \\
\theta_{AP} \in (0^\circ, \theta_{AH})
\end{cases} \tag{2}
\]

Besides the leg, parameters of gait pattern include the angular position of arms in the swing movement. Fig. 9 shows parameters of an arm in gait pattern. There are only two parameters of an arm in the gait pattern which are \( \theta_{RB} \), the angular position of right arm when it stop swinging backward in state 2, and \( \theta_{RF} \), the angular position of right arm when it stop swinging forward in state 4. There are also some constraints defined for those parameters. The learning parameters of arms in the gait pattern and the constraints of those parameters are defined as
\[
X_{ARM} = \begin{bmatrix}
\theta_{RB} \\
\theta_{RF}
\end{bmatrix} \in \mathbb{R}^2, \text{ for } \begin{cases}
\theta_{RB} \in (0^\circ, 90^\circ) \\
\theta_{RF} \in (0^\circ, 90^\circ)
\end{cases} \tag{3}
\]

Therefore, there are totally ten parameters in the proposed gait pattern which need to be tuned by machine learning method to produce a fast and stable gait.

Fig. 8. The parameters of a leg in designed gait pattern.

Fig. 9. The parameters of an arm in designed gait pattern.

3. Fuzzy Policy Gradient Learning

In the previous section, the gait pattern for humanoid robot is designed and parameterized. In the early years, a gait pattern without complex computation needs hundred times of hand tuning by experts to make it faster and stable. These unsystematic tuning may cause lots of time and hardware consumptions. Therefore, in order to generate a fast and stable gait pattern for the humanoid robot, machine learning is applied in learning of biped walking without any complex model and hand tuning.

PGRL is a method of machine learning that is applied on gait learning of robot recently [13-15, 19], somehow there still needs numbers of experiments or simulation to learn a better gait. In order to reduce the number of experiments, FPGL method, which is integrated with the concept of fuzzy logic [20-23] and PGRL, is proposed. This advanced learning method let robot learns the gait pattern with walking distance and the tilt of trunk which are observed and computed by sensors on the robot. The amount of experiments needed in this algorithm is reduced comparing to the basic policy gradient by involving the parameter relevance and policy gradient determination with fuzzy logic.

In this section, the concept of policy gradient learning algorithm and fuzzy logic control are mentioned first, following by FPGL which integrates both methods above. The configuration of gait learning that is applied in this paper is also examined in this section.

3.1 Concept of PGRL

Most of these algorithms are very efficient to solve optimization problem, but many of them need complex calculation or large number of experiments while applying on physical robot. Large number of experiments on real robot takes such a long time for learning, so a learning algorithm which is effective with small number of experiments and should converge in short time is needed in gait learning on humanoid robot. PGRL is a type of reinforcement learning is chosen as
the base of the gait learning. PGRL method has been applied on physical robot these years and researches in [19] show that this algorithm is a proper learning method for physical robot because it needs small amount of experiments to learn a set of parameters with better reward. With the reward of each policy in PGRL method, the gradient of policies in parameter space is estimated and leads the parameters set to a local optimum.

In the basic PGRL method [24], the parameter set vector is defined as \( X^i \in \mathbb{R}^n \), where \( i \) is the iteration of PGRL and \( n \) is the dimension of parameter space. The algorithm is started with an initial parameter set \( X^0 \). In the start of each iteration of PGRL, \( p \) policies \( X^{i-1}_m \) \( (m=1,2,...,p) \), are generated randomly near \( X^{i-1} \). Each of those polices is generated as \( X^{i-1}_m = X^{i-1} + [\Delta x_1 \ldots \Delta x_j]^T \), and each \( \Delta x_j \) is chosen randomly from the set \( \{-e_j, 0, +e_j\} \) for \( j = 1 \) to \( n \).

There are three sets, \( G^e_j \), \( G^0_j \) and \( G^e_j \), for each \( j \) and each of these sets includes all \( X^{i-1}_m \) that is with the some \( \Delta x_j \). For example, \( X^{i-1}_1 \) is grouped to \( G^e_j \) if \( \Delta x_i \) for \( X^{i-1}_1 \) is \(-e \). A reward \( r^{i-1}_j \) is evaluated after each \( X^{i-1}_m \) is performed by the robot.

After all \( X^{i-1}_m \) are performed, \( r_j^e \), \( r_j^0 \) and \( r_j^e \) are calculated as the average rewards respecting to \( G^e_j \), \( G^0_j \) and \( G^e_j \). The gradient \( \nabla X_j \) is constructed with \( r_j^e \), \( r_j^0 \) and \( r_j^e \) for each \( j \) by equation (4).

\[
\nabla X^i = \begin{cases}
0 & \text{if } r_j^0 > r_j^e \text{ and } r_j^0 > r_j^e \\
(r_j^e - r_j^e) & \text{otherwise}
\end{cases}
\]

Fig. 10 is an example of evaluating \( \nabla X^i \). In this example, \( n = 3 \) and \( p = 12 \). Fig. 11 is the pseudo code of the basic PGRL method. After every \( \nabla X^i \), for each \( j \) is evaluated, \( \nabla X^i = [\nabla X^i_1 \ldots \nabla X^i_T]^T \) is normalized, multiplied by a scalar step size \( \eta \), and finally added to \( X^{i-1} \), to determine \( X^i \) and begin the next iteration. In basic PGRL method, the amount of policies \( p \) is proportional to the dimension of parameter space: \( p = an \), where \( a \) is a constant factor.

Furthermore, the so-called PGRL method in this paper is in fact an extended version of traditional PGRL [24] with Parameter Relevance (PGPR) [14, 15] method.

The parameter relevance calculated by equation (5) is used to stop learning the parameters, which are trained to proper optimum or not effective to improve the reward anymore. Since the amount of policies \( p \) is proportional to the dimension of parameter space: \( p = \alpha n \), the parameter relevance can reduce the amount of policies if the relevance of some parameters is lower than the threshold, \( p^{-1} = \alpha n^{-1} \), with \( n^{-1} \leq n \), and decreases the amount of experiments to fasten the converge rate of the learning process. Fig. 12 is the pseudo code of the PGPR method, where \( T_{PR} \) is the threshold value of parameter relevance and \( I_{PR} \) is the iteration that starts removing parameters with low relevance.

![Fig. 10. An example of evaluating \( \nabla X^i \) (n=3, p=12).](image)

**Fig. 10. An example of evaluating \( \nabla X^i \) (n=3, p=12).**

**Input:** \( X^0, I_{\text{max}}, p, \eta \)

1. \begin{algorithm}
2. for iteration \( i = 1 \) to \( I_{\text{max}} \)
3. for \( m = 1 \) to \( p \)
4. \( \Delta x_j \) is chosen from \( \{-e_j, 0, +e_j\} \) randomly
5. \( X^{i-1}_m \) is grouped to \( G^e_j \), \( G^0_j \) or \( G^e_j \) depending on \( \Delta x_j \)
6. endfor
7. endfor
8. \( X^{i-1} \) is evaluated and \( X^i \) is the next iteration.
9. endfor
10. for \( j = 1 \) to \( n \)
11. \( r_j^e \), \( r_j^0 \) and \( r_j^e \) are calculated as the average reward in \( G^e_j \), \( G^0_j \) and \( G^e_j \)
12. if \( r_j^e > r_j^e \) and \( r_j^0 > r_j^e \)
13. endif
14. endfor
15. \( \nabla X^i = [\nabla X^i_1 \ldots \nabla X^i_T]^T \) is performed and \( \Delta x_j \) is evaluated
16. endfor
17. for \( j = 1 \) to \( n \)
18. endfor
19. endfor
20. endfor
21. endfor
22. endfor
23. end

**Fig. 11. Pseudo code of the basic PGRL method.**
The policy gradient determination with fuzzy logic replaces equation (4) in the extended policy gradient algorithm in order to replace the constant step-size $\eta$ and the normalization of $\nabla X^i$ that may drop down the effect of some small but important difference of parameters. This improvement increases the converge rate and decreases the amount of experiments of gait learning.

The complete process of FPGL method is expressed as the pseudo code shown in Fig. 13. Since the normalization of the policy gradient is replaced by the policy gradient determination with fuzzy logic, then the changes of those parameters are improved to be independent to each other, therefore some small changes are not ignored in the FPGL method. Thus, the policy

$$R_j = \frac{\sum_{k=1}^{f} \lambda^{i-k} |\nabla X_j^i|}{\sum_{k=1}^{f} \lambda^{i-k}}$$

where $\lambda \in [0, 1]$ is a forgetting factor to decrease the weight of the gradient in the previous iterations.

This PGPR method has been applied on physical robot and proven having higher converge rate than the basic PGRRL method. Therefore, it is used to be the base of gait learning.

### 3.2 Fuzzy Policy Gradient Learning Method

In this paper, the concept of fuzzy logic control is integrated with the PGPR method. This advanced gait learning method is named as FPGL method. The policy gradient determination with fuzzy logic in the FPGL is used to determine the gradient of parameters ($\nabla x_j^i$) with the average rewards ($r_j^x$, $r_j^y$, and $r_j^z$) by fuzzy rules.

Fig. 12. Pseudo code of the PGPR method.
gradient determination with fuzzy logic in FPGL method possesses a higher converge rate and a faster gait in shorter time.

3.3 Configuration of Gait Learning

The objective of this gait learning is to learn optimal parameters, $X \in \mathbb{R}^n$, that ensure the best performance in the walking cycle. This gait learning includes two parts, including FPGL based leg parameters learning and PGPR based arm parameters learning. Since the major purpose of this gait learning is to learn a faster walking pattern, most parameters of configuration are based on the FPGL for learning the parameters of legs. The whole structure of gait learning is shown in Fig. 14, where the structure of gait learning includes four parts, which are the sensors, reward calculation, machine learning and pattern performance in external environment. The pattern performance in the external environment means the robot performs the walking motion and let the sensors capture the needed data. The reward calculation part integrates the data from the sensors and computes the reward for machine learning which includes the walking distance of the gait pattern and the tilt of the trunk. In the machine learning part, walking distance and tilt of trunk are inputted as reward to FPGL method for legs and PGPR method for arms, then generates the next walking motion pattern from the basic stepping motion pattern. The parameters of legs and arms are updated by FPGL and PGPR, respectively.

A. FPGL Based Leg Parameters Learning

FPGL method is the major machine learning algorithm in this gait learning to learn the parameters of legs in the walking cycle. The parameters of configuration needed are already mentioned in the Pseudo code in Fig. 13. The learning parameters $X$ are the angular position of actuators in all poses of walking cycle mentioned in section 2. These parameters also must satisfy the constraints mentioned in equation (2).

Besides, the appropriate choice of the reward function for the gait learning is fundamental. As shown in Fig. 14, the reward function in FPGL method is the distance that robot walks over in some constant walking cycles. This distance can be estimated by the vision system adapted on the robot. In the learning process, the camera tracks a landmark on the floor to keep the landmark is visible in the captured image before and after the walking cycles. Since the landmark is fixed on the floor, the position of the landmark in the captured image and the angular positions of the pitch actuator of the head can be measured and computed to get the distance between the landmark and the robot before and after the walking cycle, as shown in Fig. 15. Since the repetition amount of walking cycle in every performance is fixed and the time of walking is constant, the distance as the reward for FPGL method is determined just after the end motion of gait pattern and the speed of the robot with those gait parameters can also be computed.

Furthermore, the FPGL method is started learning faster gait with a parameter set $X^0$ that is a standstill stepping motion which is hand-tuned. The direct proportion constant $\alpha$ is set as 2, which means normally the initial and maximum amount of policies is 16. For the parameter relevance, the threshold value of parameter relevance $T_{pr}$ is set as 1.5, the iteration that starts removing parameters with low relevance $I_{pr}$ is 5 and forgetting factor $\lambda$ is set as 0.8.

Somehow, some parameter sets in the parameter space may cause the robot falling down when it is walking, so in order to retain the effect of the learning, FPGL method generates another policy in the iteration once the robot falls down. Then, there are three stopping criterions for this FPGL. First, the learning stops when the number of iteration $i$ reaches $I_{max}$ which is set as 20. Second, the learning stops if all parameters have low relevance. And third, the learning stops if the robot falls down over 10 times in that iteration.

On the other hand, the flow chart of the policy gradient determination with fuzzy logic in FPGL method is shown in Fig. 16, where those inputs of policy gradient determination with fuzzy logic are the average rewards of those three groups of policies in each iteration. In the fuzzification, those three inputs which are average rewards ($r_j^a$, $r_j^0$ and $r_j^+\varepsilon$) is differed and normalized with $r_j^{-1}$ by equation (6).

$$d_j^a = \left(\frac{r_j^a - r_j^{-1}}{r_j^+ - r_j^-}\right), \text{ where } a = +\varepsilon, +0, \text{ or } -\varepsilon \quad (6)$$

![Fig. 14. The structure of gait learning.](image-url)
After the normalization, those $d_j^a$ are transferred to fuzzy sets with the triangle-shaped membership function shown in Fig. 17(a). There are three fuzzy subsets of $d_j^a$ which are negative (N), zero (Z), and positive (P).

The whole rule table in the decision making logic is depicted in Table 1. The fuzzy rules are in the form of IF-THEN rules. For example: “If $d_j^z$ is N, $d_j^0$ is Z and $d_j^+$ is P, then output is PB”. The “and” operation in the rule is realized by “min” operation as:

$$
\mu_{\text{output}}^a = \min(\mu^a_1, \mu^a_2),
$$

where $a_1$ and $a_2$ are $+\epsilon$, $-\epsilon$, $+0$, or $-\epsilon$.

Each input of fuzzy logic system normally excites one or two fuzzy values, therefore there are at most eight rules excited. If some excited rules contribute the same fuzzy subset, “max” operation is used to determine a unique membership function value.

In the defuzzification, the output membership function is defined as a single-tone type, shown in Fig. 17(b). There are five fuzzy subsets of $\mu_{\text{output}}$ which are negative big (NB), negative small (NS), zero (ZE), positive small (PS), and positive big (PB). Then all the fuzzy values of the output fuzzy subsets are summarized to determine the output value of this policy gradient determination with fuzzy logic, $\nabla x_j$, with the following equation:

$$
\nabla x_j = \mu_{\text{output}}^{PB} \times d_j^{PB} + \mu_{\text{output}}^{PS} \times d_j^{PS} + \mu_{\text{output}}^{ZE} \times d_j^{ZE} + \mu_{\text{output}}^{NS} \times d_j^{NS} + \mu_{\text{output}}^{NB} \times d_j^{NB}
$$

B. PGPR Based Arm Parameters Learning

PGPR method is used to learn the parameters of arms in the walking cycle. This is a minor part in the gait learning, so most of the parameters are the same with the FPGL part that described above, just like $p^{i-1}$, $I_{\text{max}}$ and the stopping criterions.

The purpose of arm movement is to reduce the tilt of the trunk in the walking cycle, therefore the input parameters are the arm parameters mentioned in section 2 and the reward function is the tilt degree of trunk measured by the 3-axis accelerometer installed on robot.

In order to reduce the noise measured by the accelerometer, a simple Kalman filter is applied to get accurate data. The Kalman filter [25, 26] is a mathematical method which is to use measurements that are observed over time that contain random noise and other inaccuracies, and produce values that tend to be...
closer to the true values of the measurements and their associated calculated values. The Kalman filter used here is shown as follows:

\[
\begin{align*}
P_r &= (1 - K_{r-1}) \times P_{r-1} + Q \\
K_r &= P_r / (P_r + R) \\
\hat{d}_r &= \hat{d}_{r-1} + K_r \times (d_r - \hat{d}_{r-1})
\end{align*}
\]

where \( Q \) is the process noise covariance, \( R \) is measurement noise covariance, \( d_r \) is the data measured at time \( t \) and \( \hat{d}_r \) is the data after the Kalman filter. The tilt degree of trunk in sagittal plane measured with and without Kalman filter when aiRobots-V is standing is shown in Fig. 18. In standing pose, the tilt of trunk should be \( 0^\circ \) and the figure shows that the noise of data has decreased from between \(-0.9093^\circ\) and \(0.8117^\circ\) to between \(-0.0894^\circ\) and \(0.0882^\circ\).

After measuring the tilt degree, the maximum and minimum tilt degree of aiRobots-V in the walking cycle are computed to determine the reward values for the parameters. Somehow, on purpose to simplify the learning, the parameters of arms are separated to two different learning cycles. For the tilt degree, negative value shows that the robot is tilting backward and positive value shows tilting forward. Therefore, \( \theta_{rB} \) is learned by the minimum value of tilt degree, and \( \theta_{rF} \) is learned by the maximum value as equation (10), where \( r_{rB} \) is the reward to learn \( \theta_{rB} \), \( r_{rF} \) is the reward to learn \( \theta_{rF} \), \( \phi_{\min} \) is minimum value of tilt degree, and \( \phi_{\max} \) is maximum value of tilt degree. The purpose of the learning is to reduce the tilt degree of trunk to almost zero while walking.

\[
\begin{align*}
r_{rB} &= \frac{1}{\min(0, \phi_{\min})} \\
r_{rF} &= \frac{1}{\max(0, \phi_{\max})}
\end{align*}
\]

All parameters including the parameters of legs and arms of the gait pattern for aiRobots-V are generated randomly in every policy and decided with the FPGL method and PGPR method in each iteration.

4. Experimental Results

4.1 Human-machine Interface for aiRobots-V

A human-machine interface is necessary for a small-sized humanoid robot system because there is no proper interface on the robot that allows user can tune or test the motion of the robot easily. The human-machine interface is convenient and effective to send the motion data from external computer to the robot. The communication wires connected between the external computer and the robot may interfere the motion of the robot; hence Zigbee wireless communication module is used in the system of aiRobots-V to send the motion data in data packet. After editing the motion data of aiRobots-V with the human-machine interface, the motion data is encoded and sent out through the Zigbee transmit band. The Zigbee receiving band on the motion controller receives the data packet and decodes to motion data, letting the motion controller master all actuators to proper position with proper speed due to the motion data and the robot performs the motion. In the same time, aiRobots-V sends out the data that are measured by the sensors on the robot through Zigbee. After the human-machine interface receives those data, it shows out in the interface after some calculation to let users monitor some situation of the robot. The communication structure is shown in Fig. 19.

In the process of gait learning for aiRobots-V, the gait learning page in the human-machine interface is used to handle the whole learning experiment.
The computer with the human-machine interface is connected to the robot via Zigbee module. In every iteration and policy of the gait learning, all parameters generated by the FPGL and PGPR method are packaged and sent to the robot. The robot performs the walking pattern with those parameters and starts measuring the distance and tilt of trunk in the walking cycles. Those reward data are sent to the gait learning page via Zigbee again just after the robot stops to let the progress of gait learning can continue. Fig. 20 is showing the gait learning page in the human-machine interface. There are some information needed in the progress shown in the page, such as the recent iteration and policy, the original reward of $X^0$, the best reward in the progress, the amount of fall down in this iteration and others. The walking distance is shown in millimeter and the minimum and maximum tilts of trunk are shown in degree.

4.2 Gait Learning with PGPR and FPGL Methods

The integrated gait learning includes the learning for parameters of the legs and the learning for parameters of the arms. In both experiments, the parameters of arms are all trained with PGRL method. By the way, the parameters of the legs in gait pattern are trained with PGPR method in first experiment and with FPGL method in second experiment to show the improvement of proposed method.

The rewards of the gait learning method are the walking distance which can be estimated by the vision system and the tilt of trunk which can be measured by the accelerometer. Fig. 21 is the flow chart of the gait learning for aiRobots-V. The initial gait pattern is designed and performed first. Then, policies are generated randomly and performed to evaluate the rewards. After that, the average rewards are calculated and policy gradient is determined. After the parameter relevance is computed, the stopping criterions are checked and then the learning is continued if no stopping criterion is satisfied. The difference of both experiments in the flow chart is the method to determine the policy gradient. With PGPR method, the policy gradient is determined as the pseudo code in Fig. 12, but the policy gradient is determined by fuzzy logic computation with FPGL method.

Since the results of these experiments are going to be compared, both experiments have to be run in similar conditions. Therefore, the configuration of these two gait learning are almost the same. The configuration of the gait learning with FPGL method is introduced in section 3. The only difference in the gait learning with PGPR method is the scalar step size, $\eta$, which is defined to 10 as the maximum value of possible output of fuzzy logic analysis as in Fig. 17(b). Furthermore, the order of randomizations of all policies in each iteration in the progress are saved in the first experiment, and applied in the second experiment to make this two experiments run in similar conditions.

In order to let the vision system compute the walking distance of robot, the training environment must include a landmark fixed on the ground. The training environment of this experiment is shown in Fig. 22(a). The Zigbee module in the figure is connected with a computer where the human-machine interface is running. Both experiments are run in the same environment.

The walking cycle is repeated 8 times for every performance in these experiments. Therefore, the total time including the speed up and speed down motions for every performance is about 6.24 seconds. Removing the time period when the robot starts stepping, the walking time is about 4.32 seconds. As the initial gait pattern, aiRobots-V only lifts up the legs while in the lift states in the walking cycle. These initial parameters make robot move about 40 mm forward which means the speed of aiRobots-V is about 9.26 mm/s before training. The position and the walking distance of the robot with the initial parameters after the first performance are shown in Fig. 22(b).

At first, the gait pattern of aiRobots-V is trained with PGPR method. In the progress of gait learning for aiRobots-V, the gait is faster and faster after every iteration. After 8 iterations, aiRobots-V can walk up to 693 mm in the best performance. This gait learning process takes about 80 minutes and totally 159 performances are done by aiRobots-V. The learning stops after the 8th iteration because it reaches the limit amount of fall down meaning that the robot falls down more then 10 times in that iteration. Table 2 is the training data about the gait learning and Fig. 23 shows the learning curve for aiRobots-V in first experiment.

Same with the first experiment, the training progress with FPGL method is started with an initial gait which can walk up to 40 mm with 9.26 mm/s. Then, training
runs in the same environment as in Fig. 22(a). After about 7 iterations, aiRobots-V can walk up to 701 mm in the best performance which is quite similar to the best performance in the experiment with PGPR method. The position and the walking distance of the robot with the initial parameters after the best performance are shown in Fig. 22(c).

Table 2. The training data of gait learning with PGPR method.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original distance (mm)</td>
<td>40</td>
<td>126</td>
<td>203</td>
<td>265</td>
<td>372</td>
<td>443</td>
<td>546</td>
<td>674</td>
</tr>
<tr>
<td>Minimum distance of policies (mm)</td>
<td>0</td>
<td>98</td>
<td>166</td>
<td>256</td>
<td>288</td>
<td>302</td>
<td>398</td>
<td>502</td>
</tr>
<tr>
<td>Maximum distance of policies (mm)</td>
<td>221</td>
<td>266</td>
<td>305</td>
<td>382</td>
<td>491</td>
<td>597</td>
<td>683</td>
<td>693</td>
</tr>
<tr>
<td>Amount of policies</td>
<td>16</td>
<td>19</td>
<td>21</td>
<td>19</td>
<td>23</td>
<td>21</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Amount of fall down</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Parameters with high relevance</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>-</td>
</tr>
</tbody>
</table>

Fig. 23. The learning curve of gait learning with PGPR method.

Table 3. The training data of gait learning with FPGL method.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original distance (mm)</td>
<td>40</td>
<td>158</td>
<td>251</td>
<td>332</td>
<td>423</td>
<td>558</td>
<td>701</td>
</tr>
<tr>
<td>Minimum distance of policies (mm)</td>
<td>-5</td>
<td>102</td>
<td>175</td>
<td>247</td>
<td>302</td>
<td>398</td>
<td>556</td>
</tr>
<tr>
<td>Maximum distance of policies (mm)</td>
<td>221</td>
<td>385</td>
<td>402</td>
<td>508</td>
<td>597</td>
<td>695</td>
<td>692</td>
</tr>
<tr>
<td>Amount of policies</td>
<td>16</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>19</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Amount of fall down</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Parameters with high relevance</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>-</td>
</tr>
</tbody>
</table>

Fig. 22. (a) The training environment of gait learning (b) The position and walking distance of the robot after the first performance (c) The position and walking distance of the robot after the best performance.
This gait learning process takes about an hour and totally 138 performances are done by aiRobots-V. The learning stops after the 7th iteration caused by the same stopping reason as the previous experiment which means the robot falls down more than 10 times in that iteration. Table 3 is the training data about the gait learning and Fig. 24 shows the learning curve for aiRobots-V in this experiment. The speed of aiRobots-V is about 162.27 mm/s after the training. This speed is about 17.5 times of the original speed before the gait learning.

From the results, FPGL method can train the gait of aiRobots-V in shorter time and fewer amounts of performances comparing to PGPR method. Since the best gait pattern is learned after 159 performances with PGPR method and learned after 138 performances with FPGL method, FPGL method can reduce up to 13% performances to learn a fast and stable gait. This improvement can save the time and consumes of hardware in the gait learning for humanoid robot.

4.3 Tilt of Trunk with and without Arm Movement

In the gait pattern and gait learning for aiRobots-V, arm movement is quite important to reduce the tilt degrees of trunk of robot. In this experiment, the tilt degrees of trunk in the walking cycles with and without arm movement which learned in the gait learning process with FPGL method. Fig. 25(a) shows the tilt of trunk without arm movement and Fig. 25(b) shows the tilt of trunk with arm movement in the best performance in the first experiment. In these figures, there are 32 sampling data about the tilt of trunk in the walking cycles. In these sampling data, the even sampling time shows the tilt of trunk in state 1 or 3 and the odd sampling time shows those tilt in state 2 or 4. Before addition of arm movement, the maximum tilt of trunk is around 2° and minimum is around 4°. In contrast, the maximum tilt of trunk is around 1° and minimum is around 2° after adding the arm movement in the gait pattern which means the arm movement reduces about 50% of the tilt of trunk while the robot is walking.

5. Conclusions

In this paper, a new gait pattern generation for small-sized humanoid robot is designed and implemented. The design concept of gait pattern is introduced and the gait pattern for aiRobots-V is parameterized in section 2. The improvement in the gait pattern for aiRobots-V makes it walk faster. Moreover, movements of arms are added to the gait pattern to reduce the tilt degree of trunk in the walking cycle. Those parameters of legs and arms in the gait pattern can be trained with machine learning method.

In section 3, the machine learning used in the gait learning is named fuzzy policy gradient learning (FPGL) method. This method is based on policy gradient reinforcement learning (PGRL) method and policy gradient with parameter relevance (PGPR) method, and then integrated with fuzzy logic control system to determine the policy gradient in the learning progress. This improvement increases the converge rate and decreases the amount of experiments of gait learning.
The reward of FPGL method for learning the parameters of legs in the gait pattern is the walking distance in constant walking cycles which can be measured by the vision system. On the other hand, the maximum and minimum tilt degrees of trunk in the walking cycled are chosen as the rewards to train the parameters of arms in the gait pattern.

Several experiments are executed to evaluate the effect of FPGL method in section 4. In the gait learning experiment with FPGL method, the gait pattern of aiRobots-V is trained from 9.26 mm/s walking speed to 162.27 mm/s in about an hour and 138 times of experiment. These shows the feasibility of proposed method and it is much faster and more systematical than hand-tuning. In contrast, the robot is trained to the similar speed in about 80 minutes and 159 times of experiment with PGPR method. This experimental result shows the FPGL method improves the efficiency of PGPR method up to 13%. The data of experiments also shows the arm movement could reduce about 50% of the tilt of trunk while the robot is walking.

From the experimental results, the effectiveness and applicability of the proposed learning scheme for humanoid robot walking are verified. Additionally, aiRobots-V is also applied in RoboCup 2010 and won the second place of the technical challenge.

Acknowledgment

This work is supported by the National Science Council of the Republic of China under NSC97-2221-E-006-172-MY3 is gratefully acknowledged.

References


Yu-Te Su was born in Kaohsiung, Taiwan, in 1982. He received the B.S. degree in the Department of Engineering Science and the M.S. degree in the Department of Electrical Engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C., in 2004 and 2006, respectively. He is currently working toward the Ph.D. in Electrical Engineering at National Cheng Kung University. His research interests include fuzzy control, reinforcement learning, humanoid robot, image processing, and FIRA/RoboCup games.

Kiah-Yang Chong received the B.S. and M.S. degree from the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C., in 2008 and 2010. His research interests include fuzzy control, humanoid robot, and FIRA/RoboCup games.

Tzuu-Hseng S. Li (S’85–M’90) received the B.S. degree in electrical engineering from the Tatung Institute of Technology, Taipei, Taiwan, R.O.C., in 1981, and the M.S. and Ph.D. degrees in electrical engineering from the National Cheng Kung University (NCKU), Tainan, Taiwan, in 1985 and 1989, respectively. Since 1985, he has been with the Department of Electrical Engineering, NCKU, where he is currently a Distinguished Professor. From 1996 to 2009, he was a Researcher and Consultant with the Engineering and Technology Promotion Center, National Science Council, Tainan. From 1999 to 2002, he was the Director of the Electrical Laboratories, NCKU. In 2006, he received the Outstanding Automatic Control Award from the Chinese Automatic Control Society (CACS) in Taiwan. He also received the CACS Fellow in 2008. He was elected as the President of the CACS from 2008 to 2011. He has served as a dean of the college of electrical engineering and computer science of National United University and the vice President of the Federation of International Robot-soccer Association (FIRA), since August 1, 2009. His current research interests include automobile active suspension systems, computational intelligent algorithms, fuzzy system and control, humanoid robots, mechatronics, mobile robots, reinforcement learning, and singular perturbation.

Dr. Li was a Technical Editor for the *IEEE/ASME Transactions on Mechatronics* and an Associate Editor for the *International Journal of Nonlinear Science and Numerical Simulations*, and now is Associate Editors for the *Asia Journal of Control and International Journal of Fuzzy Systems*. 

---

Yu-Te Su

Kiah-Yang Chong

Tzuu-Hseng S. Li