Sensor-Based Programming of Central Pattern Generators in Humanoid Robots

Regular Paper

Hamed Shahbazi1,*, Kamal Jamshidi1 and Amir Hasan Monadjemi1

1 Department of Computer Engineering, University of Isfahan, Isfahan, Iran
* Corresponding author E-mail: shahbazi@eng.ui.ac.ir

Received 16 Apr 2012; Accepted 10 Dec 2012

DOI: 10.5772/55462

© 2013 Shahbazi et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract In the present article, a method for generating curvilinear bipedal walking patterns is proposed which is able to generate rhythmic and periodic trajectories for a Nao soccer player robot. To do so, a programmable central pattern generator was used which was inspired from locomotion structures in vertebrate animals. In this paper, the programmable central pattern generators were extended and new Equations were added to make a curvilinear pattern for walking Nao robots on a specified circular curve. In addition, some specific Equations were added to the model to control the arms and synchronize them with the movement of the feet. The model uses some sensory inputs to obtain some feedback from the movement and adjust it conforming to the potential perturbations. Input sensory values consist of accelerometer values and foot pressure sensor values located on the bottom of each foot. Feedback values can adopt walking to some desired specifications and compensate the effects of some types of perturbations. The proposed model has many benefits including smooth walking patterns and modulation during walking. This model can be extended and used in the Nao soccer player both for the standard platform and the 3D soccer simulation leagues of Robocup SPL competitions to train different types of motions.

Keywords Decentralized control, High-Order Neural Networks, Extended Kalman Filter, Backstepping

1. Introduction

Nature is usually a very good source of inspiration for science and technology. Taking inspiration from it, humans can usually make a very similar instance to the real case under study. For example, some complimentary algorithms in artificial intelligence are inspired from ants or bees. We can use these ideas and inspiration to build an artificial instance which is able to imitate the real specimen and use these ideas to complete our approach. “Humanoid Robots” are good examples of this kind of inspiration. Recent investigations on the walking of these robots are important parts of robotic development. The problem of walking with two feet in a humanoid robot is a very complicated problem which seems unlikely to be solved in the near future.

The problem of robot locomotion is where neuroscience and robotics converge. This common area of research centres on pattern generators in the spinal cord of vertebrate animals called “Central Pattern Generators” (CPGs)[1]. Central pattern generators are neural circuits
located in the end parts of the brain and first parts of the spinal cord of a large number of animals and they are responsible for generating rhythmic and periodic patterns of locomotion in different parts of the body. Although these pattern generators use very simple sensory inputs imported from the sensory systems, they can produce high dimensional and complex patterns for walking, swimming, jumping, turning and other types of locomotion. The origin of many movements in animals is the central pattern generators which were discovered by Brone in the early decades of the 20th century [2]. He discovered that the movement in many animals is an outcome of central neuronal activities in some parts of their neural system, and simple sensory inputs change these activations and make them capable of responding to the extraneous perturbations. The idea that CPGs are neural networks generating complex locomotion patterns with only simple inputs is a provocative one - it is this that we intend to model in this paper.

In this paper, a model for programmable central pattern generators capable of generating curvilinear biped locomotion patterns is developed. The proposed model, which is a model for programmable central pattern generators in a Hoap2 humanoid robot, is based on [3]. The concept of programmable central pattern generators is to make a model for CPGs that can learn from data and generate trajectories similar to them after a period of training. The thought that CPGs can be programmed is a fascinating idea which can be developed to obtain a systematic method of locomotion. In this way, any kind of complex behaviour can be programmed in a CPG model and it can be used in the movement of diverse types of robots.

In the next section, related works in the field of humanoid robot locomotion are reviewed and the advantages and disadvantages in each method are discussed. Section 3 introduces the method for making curvilinear patterns in the Nao robots and how to use programmable central pattern generators to generate the desired patterns. In this section, the features of Nao robots which are the main subjects in the Standard Platform league in Robocup Soccer competitions are explained in brief. Arm movements and different sensory feedbacks integrated in the model used are discussed. Section 4 includes experimental results. We show some of the implementations and results in the Webots simulator and Simulink of Matlab. In addition, we demonstrate some snapshots from the achieved curvilinear walk. In section 5, the conclusions and future prospective works are stated.

2. Related Works

There are many approaches to solving the bipedal walking issue [4]. Some of these methods use pre-recorded trajectories. These methods are completely offline and never use feedback from robot sensors. The main aim of these methods is to optimize the trajectories and find the best ones to use in locomotion. Finding the best trajectories usually deals with an optimization method in machine learning which can be solved with evolutionary algorithms. Different evolutionary algorithms for this purpose are discussed in [5]. Evolutionary algorithms, such as Particle Swarm Optimization (PSO) [6], Genetic Algorithms (GA) [7], Reinforcement Learning (RL) [8] and Policy Gradient Learning (PGL) [4] [9], have been employed commonly in these types of walk learning. Yang et al. used a truncated Fourier series method to model trajectories for walking in some types of humanoid robots and utilized a genetic algorithm to optimize Fourier coefficients [10]. Some researchers, such as Marc, Kohl and Stone, used some optimization methods for discovering the best parameters of trajectories to maximize a desired fitness function, which is usually the speed of walking or the combination of speed and energy consumption [11],[12].

A very putative method for bipedal walking is the (so-called) Zero Momentum Point (ZMP). ZMP-based approach uses the dynamical model of the robot to calculate a point in which all momentums of the whole robot converge to zero, and it is attempted to keep this point in a suitable and safe area. The original version of this method was discussed by Vukobratovic, who is the founder of this method [13],[14]. Beng Tay used the ZMP method to build omni-directional biped walking in the Nao robot [15], and this has already been used in some other robots [16],[17].

As an alternative to the methods using pre-recorded trajectories, ZMP-based approaches or methods using heuristic control laws (e.g., Virtual Model Control (VMC) [18]), the CPG-based methods are introduced, using some biological perspectives. They encode rhythmic trajectories as limit cycles of nonlinear dynamical systems [19]. A dynamical system is a concept in mathematics where a fixed rule describes the time dependency of a point in a geometrical space. Examples include the mathematical models that describe the swinging of a clock pendulum, the flow of water in a pipe, etc.

Coupled oscillator-based CPG implementations offer miscellaneous fascinating features, such as the stability properties of the limit-cycle behaviour (i.e., the ability to forget perturbations and compensate their effects), the smooth online modulation of trajectories through changes in the parameters of the dynamical system, and entrainment phenomena when the CPG is coupled with a mechanical system. The implementations of CPGs based on the coupled oscillators are actually designs of stable limit cycles in some interconnected oscillators generating patterns. We first design some stable walking limit cycles,
and then map external perturbation as some sensory feedbacks to them, and force them to return to the original shape in a smooth and stable manner.

Interesting examples of CPGs applied to biped locomotion are included in [2] and [20]. Matsubara et al. discussed a CPG-based method for biped walking which was combined with policy gradient learning [20]. One drawback of the CPG approach is that most of the time these CPGs have to be tailor-made for a specific application, and there are few techniques to construct a CPG for generating an arbitrary input trajectory. Righetti and Ijspeert represented a model for constructions of a generic model of CPG [3]. This method was a programmable central pattern generator which used dynamical systems and some differential Equations to build up a training algorithm. The learner model is based on the works of Righetti, Buchli and Ijspeert, which is a Hebian learning method in dynamical Hopfs oscillators. The programmable central pattern generator has been used to generate walking patterns for a Hoap2 robot. Using this type of generic CPG, they trained the generic CPGs with sample trajectories of walking patterns of the Hoap-2 robot provided by Fujitsu. Each trajectory was a teacher signal to the corresponding CPG controlling the associated joints [21].

Gams et al. discussed a system for learning and encoding a periodic signal with no knowledge on its frequency and waveform, which was able to modulate an input periodic trajectory in response to some external events [22]. Their system is not used to learn walking but some other periodic tasks undertaken by the arms of a humanoid HOAP-2 robot for the task of drumming. This model uses two layers of trajectory generation. The first layer, the Canonical Dynamical System (CDS), is actually a polar implementation of generic CPGs included in [3]. The second layer, the Output Dynamical System (ODS), is responsible for learning and regenerating the waveform of the input signal.

In this paper, a modified and developed model of the programmable central pattern is used for training a curvilinear walking pattern instead of a rectilinear (direct) one. The model for this kind of walk can be used in Nao soccer robots which can be used in the Robocup 3D Soccer league.

3. Curvilinear CPG Model

In this section, we explain what the curvilinear bipedal walk is and how generic CPGs are developed to train this walking to the controller of our Nao robot. First of all we describe the global architecture of the controller and its different components. Next, we discuss the nonlinear dynamical oscillators and the fundamental properties of these oscillators. Then the coupling scheme between joints and its role in walking is discussed. In the second subsection we explain the modifications of curvilinear behaviour and new Equations that should be inserted into the model. Subsection 3 is devoted to the arms’ movement. We investigate the effects of moving the arms during a curvilinear walking and introduce new Equations for implementing them. The last two subsections describe the offline mode of training and the online controller of the robot. We have tried to explain details of implementation here to make this paper a suitable reference for other researchers in this field.

Nao is one of the most advanced humanoid robots made at the Aldebran French Corporation which was first publicly presented in 2006. In 2007, Nao replaced the robot dog Aibo by Sony as the robot used in the Robocup competitions. This robot has 26 degrees of freedom in different parts of its body including the head, shoulders, elbows, hips, knees and ankles. The Nao academics version is now available for universities and laboratories for research and education purposes. Figure 1.a shows this robot and the different parts of its equipment and joints. The large number of servo motors in its joints has made Nao one of the most flexible humanoid robots to have ever been made and this has also made it the best choice for the Standard Platform league in Robocup soccer international competitions. Details about this robot can be found in [11],[15].

The special type of locomotion we have focused on is curvilinear walking. The curvilinear walk is a smooth form of travelling from a point in the soccer field to a new point along a curvilinear path. A special case of curvilinear path is that of part of a circle. This kind of walking is shown in Figure 1.b.

![Figure 1. a) Nao Robot, different joints and axis convention, joints which are numbered are used in the paper b) Curvilinear walk in a Nao robot](image-url)
During a curvilinear walk the robot should simultaneously rotate and move forward, and it must walk along the desired path [9].

3.1 Generic CPG Model Architecture in Nao

The architecture of the model used in the present research to use a generic CPG and feedback sensory inputs in the controlling structure of our Nao robot is a bi-phase structure (see Figure 2). In the first phase, called the training phase, we should train our robot to a fundamental method of walking. The Nao robot did not have any information about how to move its joints to go forward or rotate. So we should train it with some walking trajectories. The training module consists of three sub-modules. The first sub-module is a database of pre-recorded trajectories. This contains walking trajectories of the Nao robot in different forms. A gait walking trajectory which is a slow speed and stable type of walking was used in this research.

The timer is the second module which is responsible for mapping a continuous time to some discrete values of index for pre-recorded matrices of walking trajectories. The third module is the generic CPG which is the core of the learning section. The training phase of the robot is an offline process and does not need to be implemented in the robot controller. It can be performed in the MATLAB Simulink environment.

The second phase of walking generation is the online sensory-based walking generation. In this phase, we transfer state vectors which are trained in the first phase to the generic CPGs of the robot controller and initialize CPGs with their necessary values. Sensory input values used in this controller are accelerometer values and foot-pressure sensor values. An accelerometer is a hardware module in the chest of the robot measuring acceleration values along different directions X, Y and Z.

The fundamental building block of the generic CPG is the adaptive frequency the Hopf oscillator, which is proposed in [21]. These oscillators have the property of learning the frequency of a periodic input signal without any external optimization. Usually, the frequency of an oscillator can be controlled by a specific parameter. In this model all the parameters are changed into a state variable which can be trained using a general evolution rule. It can be proved that when perturbed by a periodic input signal, these state variables will converge to one of the frequency elements of that signal. The adaptation is an intrinsic characteristic of these oscillators. In addition, there is no need for a supervision or external processing.

After convergence, if the input signal disappears, the learned frequency would remain encoded in the system. The relations governing this oscillator are as follows 1, 2, 3:

\[
\frac{\partial}{\partial t} x_i = \gamma (\mu - r^2) x_i - \omega y_i + 2 e F(t) + r \sin(\theta_i - \phi) 
\]

(1)

\[
\frac{\partial}{\partial t} y_i = \gamma (\mu - r^2) y_i + \omega x_i 
\]

(2)

\[
\frac{\partial}{\partial t} \omega_i = -e F(t) \frac{y_i}{r_i} 
\]

(3)

where \( r = \sqrt{x^2 + y^2} \) is radius of the oscillation. \( \mu \) controls the amplitude of the oscillations, \( \gamma \) controls the speed of recovery after perturbation, \( \omega \) controls the frequency of the oscillations, \( F(t) \) is a periodic input to which the oscillator will adapt its frequency, and \( e > 0 \) is a coupling constant. The oscillation frequency will adapt to one of the frequency components of the input \( F(t) \), regarding the initial conditions of \( \omega \).
Construction of a PCPG requires connecting and coupling these Hopf oscillators. This connection is shown in Figure 3. The$	ext{P}_{\text{teach}}(t)$ signal is the desired trajectory which is needed to train into the CPGs. $\text{Q}_{\text{learned}}(t)$ is what the system has learned until the time $t$. The difference of these signals is used as a perturbation signal for the Hopf oscillators. The output amplitude of each oscillator is multiplied by an alpha coefficient and then all of these outputs are added together.

Equations 4-8 complete the definition of a generic CPG:

$$\frac{\partial}{\partial t} \alpha_i = \eta x_i F(t)$$  \hspace{1cm} (4)

$$\frac{\partial}{\partial t} \theta_i = \sin \left( \frac{\alpha_0}{\alpha_i} \theta_0 - \theta_i - \phi \right)$$  \hspace{1cm} (5)

$$\theta_i = \text{sign}(x_i) \cos^{-1} \left( -\frac{x_i}{\eta} \right)$$  \hspace{1cm} (6)

$$F(t) = \text{P}_{\text{teach}}(t) - \text{Q}_{\text{learned}}(t)$$  \hspace{1cm} (7)

$$\text{Q}_{\text{learned}}(t) = \sum_{i=0}^{N} \alpha_i x_i$$  \hspace{1cm} (8)

Here $\tau$ and $\epsilon$ are two constants for coupling oscillators and $\eta$ is a training constant. $\text{Q}_{\text{learned}}$ is the output of this programmable CPG which is computed as a weighted sum of each oscillator outputs. $F(t)$ is the learning feedback, which is the feedback of the system showing how much learning has been done and how much the teaching signal $\text{P}_{\text{teach}}(t)$ still should teach to the CPGs. $\alpha_i$ shows the amplitude dedicated to the frequency $\alpha_0$ of the $i$'th oscillator. During the learning phase, the aim is to maximize the correlation between $x_i$ and $F(t)$, which results to an increment of $\alpha_i$ if and only if $\alpha_0$ converges to a frequency component of $F(t)$, and will stop increasing when the frequency component $\alpha_0$ disappears from $F(t)$ because of the negative feedback loop. $\alpha_i$ is the phase difference between the $i$'th and the zeroth oscillator. It can converge to the phase difference between the instantaneous phase of the zeroth oscillator, $\theta_0$, which is scaled at frequency $\omega_0$ and the instantaneous phase of the $i$'th oscillator, $\theta_i$. All the oscillators are coupled with the zeroth oscillator, and the coefficient $\tau$ is used to keep correct phase relationships between the oscillators to build correct phase synchronization.

We use one generic CPG for each joint of the Nao in each Degree Of Freedom (DOF) of each foot. Each CPG is made of exactly four oscillators, as shown in Figure 3. To coordinate these joints, for each leg and the opposite arm, a chain coupling from the hip to the ankle of the first oscillator of each CPG has been used. This coupling is illustrated in Figure 4. In this Figure, joint number 1 and 7 (hip yaw-pitch joints of the left and right feet) are synchronized together. Joints 2, 3, 4, 5 and 6 in one set and 8, 9, 10, 11 and 12 in the other set are synchronized sequentially. Joint 13 (the right shoulder pitch joint) is synchronized with joints 2 and 3, and joint 14 is synchronized with joints 8 and 9 (the left-shoulder pitch joint). In addition, some extra terms are added to keep correct phase differences between the DOFs. To train different trajectories to these generic CPGs, we need to enter the training trajectories into the system and after a long period of training, the parameters of CPGs are adapted to the desired input. Then the last state of the system will be transferred to the robot controller and in each step of the walk, calculation of an integral should be done to find the new state of the system.
The model of generic CPGs presented generates online trajectories for each joint of Nao. We use these trajectories as the desired angles (set points) for the PID controllers controlling each joint. The major benefit of this model is that we can encode arbitrary periodic signals as limit cycles in a network of coupled oscillators. When we get all the properties of such systems, we can modulate the frequency and the amplitude in a smooth way, we have stability to perturbations and we can integrate feedback pathways to adaptively regenerate trajectories and optimize different skills in a soccer player robot.

3.2 Curvilinear Generation Equations

We add a new state variable to these Equations to make a curvilinear control in the generic CPG model. In fact a curvilinear walk is a result of smooth changes in the yaw/pitch servo of the hip joint in the Nao model. So we add Equations 9 and 10 to the system to be used for hip yaw/pitch joints:

\[
\frac{d}{dt} x_i = \zeta \left( y \left( \mu - r^2 \right) x_i - \omega y_j \right) + R(t) \frac{x_i}{t} \quad (9)
\]

\[
\frac{d}{dt} y_i = \zeta \left( y \left( \mu - r^2 \right) y_i - \omega y_j \right) + R(t) \frac{y_i}{t} \quad (10)
\]

Here \( \zeta \) is a small coordinating coefficient which is used to make yaw/pitch joint behaviour similar to the other joints. \( R(t) = \frac{\text{Acc}(t)}{R} \) is the expected rotation of the robot in the time \( t \) which can be computed with this Equation:

In this Equation \( R \) is the radius of a circular curve and \( \text{Acc}(t) \) is the speed increment of the robot in the direction of the robot’s body. In fact \( R \) at the time \( t \) is the projection of rotation in CPG Equations and was computed from the radius of the circular curve and the amount of acceleration. \( R \) is inversely proportional with \( R(t) \). The larger the radius of the curve, the smaller the projection of rotation in each time, and thus the robot would deviate more slowly from its straight path.

3.3 Arms Movements

Some experiments carried out by researchers have found that the arms’ movements actually provide important indirect benefits for human walking [9]. These researches have shown that swinging the arms in opposition to the legs significantly increases the efficiency of walking. They also discovered that normal arm swinging made walking much easier and smoother. To stabilize and increase the performance of our curvilinear walk we should include the role of the arms in robot walking. The most important arm joint which mostly moves the arm is the shoulder pitch joint. This joint is responsible for moving the arm in the direction of the body (X axis) and can be used to keep the centre of the momentum in a safe and stable area. So only these pitch joints of the shoulders have been considered and the roll joints pertaining to the four elbows and the two shoulders have been locked. There are two different arm-control rules for pure rotation and pure rectilinear walks, and a linear combination of these is used for a curvilinear walk. In the rectilinear case, the angular momentum generated by the arms should repeal unwanted momenta around the Y axis which may be generated by the foot swinging. On the other hand, in a pure rotation case, the angular momentum generated by the arms must intensify the momentum around Z which is used for rotation.

For the arms we do not use direct programming of generic CPGs in the shoulder pitch joints. In other words we compute a shoulder pitch joint indirectly with a linear combination of the other CPG values. Since we intend to compensate the effects of the swing phase, the left arm should be synchronized with the right foot and the right arm should be synchronized with the left foot. This means that whenever the left foot is in its swing phase and it goes forward, the right arm should simultaneously go forward and the left arm should go backward. We use Equations 11-12 to control arm movements:

\[
\text{Pitch}_{\text{RShoulder}}(t) = \rho_1 \text{Roll}_{\text{LHip}}(t) + \rho_2 \text{Pitch}_{\text{LHip}}(t) + b \quad (11)
\]

\[
\text{Pitch}_{\text{LShoulder}}(t) = \rho_3 \text{Roll}_{\text{LHip}}(t) + \rho_4 \text{Pitch}_{\text{RHip}}(t) + b \quad (12)
\]

In the above Equations the left shoulder pitch-joint position at the time \( t \) is computed from a linear combination of the right hip-roll joint position and the right hip pitch-joint position. \( R \) is the radius of curvilinear rotation discussed earlier in curvilinear Equations. \( \rho_1, \rho_2, \rho_3, \rho_4 \) are the coefficient of \( R \) which determines how much momentum the opposite arm should generate. \( b \) is a bias value which shifts the range of the movements of the arms. They can be determined by trial and error by some learning procedure [23].

3.4 Foot Sensors Feedbacks

The fact that there is a large amount of information embedded in foot pressure sensors is interesting in that this can be added to the model of bipedal walking. Foot sensor feedbacks can determine different states of walking and help the robot control its stability and respond to the changes that occur in the surface of walking. If any part of the walking surface has pits or hills, these pressure sensors tell the robot how to change its walking patterns to bypass these pits and hills. Again, foot-sensor pressure values are the main source of pattern generators located in natural CPGs in vertebrate animals.

Inspired from the natural world, we add the effects of the foot sensors to our model to tune up its efficiency. These
feedback effects control the phase differences between the joints in the two feet. There are four foot sensors on the bottom of each foot of the Nao robot. Considering the exact effects of these foot sensory inputs, according to the current research on humans, is a complicated problem. Thus, we use only one effect among several effects which is the phase deviance occurring between the two feet of the Nao Robot. When the phase differences between the robot’s feet is disturbed, it is said that a “phase deviance” has occurred. This deviance can be generated from different origins including non-smoothness of the walking surface, collision with another soccer player robot, collision with the walking ball, and rotation of the body during the walk. To solve this type of deviance, we update our trained CPGs and import a combination of foot sensor effects to compensate the phase deviance. Equation 13 is utilized to compute the foot sensory effects. We define Equations 14 to modify phase Equations in the generic CPG model and import the foot sensor effects. Using these new online phase calculation Equations, one can modify the phases of the hip joint (joint numbers 3 and 9) and knee joint (joint numbers 4 and 10). These joints are the most important ones during a curvilinear walk. Other joints of the Nao robot cause no more change in outcomes. This was found to be sufficient to import the foot sensor effects only to the hip joint and knee joints to compensate the lagging and leading of Nao’s feet.

We have designed a special value for combining foot sensor values as follows:

\[
F_{\text{diff}} = \frac{1}{m} \int_{t \rightarrow t+T} \left( \sum_{i=1}^{4} \text{Fsr}(i, \text{left}) - \text{Fsr}(i, \text{right}) \right) \text{dt} \tag{13}
\]

According to Equation 13, in each cycle the difference of corresponding sensors in both feet is computed and summed together in each time period \( T \) i.e., the period of one complete gait to form a foot sensor difference \( F_{\text{diff}} \) value. Since this value is the sum of forces, \( F_{\text{diff}} \) should be scaled by a divisor \( m \) to be used in the CPGs’ modified Equations (in Equation 14).

In this Equation, \( m \) is a normalizing constant which is used to scale the pressure values to the appropriate phase difference values. \( T \) is the time interval of a gait period. \( \text{Fsr}(i, j) \) is the foot pressure of the \( i \)th sensor (\( i \) can be 1 to 4) in the \( j \)th foot (\( j \) can be 1 and 2 for the left and right foot, respectively).

\[
\frac{\partial}{\partial t} \phi_{k,i} = \left( \frac{\alpha_k - \alpha_0}{\alpha_0} \phi_{k,i} - \phi_{k,i} \right) + \left( \phi_{k,i} - \phi_{k,i+1} \right) F_{\text{diff}} \tag{14}
\]

where \( \phi_{k,i} \) represent the phase change in the \( i \)th oscillator of the \( k \)th joint of the robot. Here we have updated Eq. 9 for all the oscillators in joints \( k=3,4,9,10 \).

3.5 Offline training phase

In the general architecture of the present model, two phases of curvilinear bipedal pattern generation have been discussed. The first phase includes the training of basic walking trajectories to our general CPGs. This should be done to find initial-state values of essential harmonic frequencies, phases and amplitudes shaping the trajectories. The first adaptive oscillator in each generic CPG sends out its frequency and phase value as synchronization criteria in other adaptive oscillators. It is noteworthy that only four adaptive oscillators have been used to reduce computational complexity of the model.

The present model has been trained and its last state values have been extracted in Simulink. Then generic CPGs have been transferred to the online controller code of the Nao which was written in Matlab.

3.6 A Discussion on the Online Controller

In this subsection we are going to describe the overall functionality of the online controller module of our model. To show how the robot can walk using our model, some pseudo codes are proposed. These pseudo codes are originated from the Matlab language syntax and use some API functions of the Webots (Robotstadium) simulator.

The main goal of this section is to help the reader better understand the details of implementation. The main robot online controller pseudo code is shown below. One part of this code is the initialization of servos and sensors. This is usually required when the robot controller begins to enforce its control.

1. Initialize the Robot Servo motors in robot feet and arms;
2. Initialize the Robot Sensors;
3. Set State0 to the initial values Achieved from Training;
4. Step = 1;
5. While Robot_Step(TIME_STEP) -= -1 do
   6. Acc = Accelerometer_Get_Values ;
   7. Calculate Fdiff from Equation 13;
   8. \( [T,X] = \text{ode45} (@\text{CalculateNextState}, \text{State0}, \text{Acc}, \text{Fdiff}); \)
   9. Calculate Qlearn from X;
10. Calculate Arm Positions;
11. For All k in Qlearn do
   12. Joints = Qlearn/100;
   13. Servo_set_position(Servo_Name, Joints (k));
14. End;
15. Step = Step + 1;
16. state0 = X;
17. End;

In this pseudo code, CalculateNextState is a function which can calculate the CPG next state from the inputs which are the time interval of integration, the current state of the robot named State0, Acc values, and the value
of \( F_{\text{diff}} \) computed from the foot pressure sensors. To compute the integrations, the ode45 function has been used in the implementations since it is a very flexible method of integration. The next pseudo code shows CalculateNextState computations in the box below. In this pseudo code, three inputs are entered into the function. The first one is the time value. The second is current state of the robot which consists of all values of \( X, Y, \Omega, \alpha, \gamma, \phi \). The third input is the values of the accelerometer sensor from the main pseudo code. The function computes dstate vector which is the input of ode45 or integrating function.

An important problem when using this generic CPG model is the quantization of the time. The time is continuous and should be quantized to be used in the training vector indices. To solve this problem, the time has been multiplied by 25 and then the integer part of the number has been used in the calculations. We have selected 25 because the time step was 40 milliseconds. Another problem in our experiments is the small values of input trajectories - with small values of input signals CPGs never converging to an acceptable point. It was found that the training trajectories should be large enough to train CPGs. So all trajectories were multiplied by 100. After the training phase, CPGs generate trajectories 100 times bigger than they should be. So in the main part of the code we have multiplied the Qlearn vector by 1/100.

1. Function [ dstate ] = CalculateNextState (state, Acc, Fdiff)
2. gama = 8; epsilon = 0.9; taw = 2; eta = 0.5; miu = 1; tao = 0.5;
3. Set number of adaptive oscillators \( N = 4; \)
4. TIME_PERIOD = 66;
5. TIME_STEP = 40;
6. % calculate the Qlearn
7. For \( k = 1 \) to \( 12 \) do
8. For \( j = 1 \) to \( N \) do
9. temp = state((k-1)\( \times \)N+5)+j*state((k-1)\( \times \)N+3);\( \times \)N+j));
10. Qlearn(k) = Qlearn(k)+temp;
11. End;
12. End;
13. t3 = floor (t / 25);
14. t3 = mod (t3,TIME_PERIOD) + 1;
15. For \( k = 1 \) to \( 12 \) do
16. Pteach(k) = Md(t3,k)*100;
17. End
18. Ro = Acc(X) / R;
19. For \( k = 1 \) to \( 12 \) do
20. For \( i = 1 \) to \( N \) do
21. Compute dx(i,k), dy(i,k), domega(i,k),
22. domega(i,k), dphi(i,k);
23. End;
24. End;

4. Experimental Results

In this section we present implementation methods and experimental results of the CPG-based curvilinear bipedal walking model used in this research. The first stage experiments (training phase of CPGs) were performed using the Simulink toolbox in Matlab. In the second stage (the online controlling of the robot), an integrated simulation of the Nao robot in Webots Robotstadium was used [7]. Robotstadium is an online simulated version of Robocup Standard Platform competitions. This contest is proposed by the Cyberbotics corporation, which has pioneered and developed the WebotTM simulator. This simulator is based on ODE, an open source physics engine for simulating 3D rigid-body dynamics. The model of the robot is as close to the real robot as the simulation enables us to do. This means we simulate the exact number of DOFs, the same mass distribution and inertia matrix for each limb, the same sensors (gyroscope and accelerometer in the chest and the load sensors on the bottom of the feet).

4.1 Offline programming of CPGs

In Figure. 5 an example of a diagram in training mode is illustrated. The red trajectory is the left-hip pitch-joint values (Pteach, multiplied by 100 to be able to be trained in the CPGs) and the blue one (Qlearn) is what the corresponding CPGs is generated during the first 10 seconds. It is observed that the training is very fast and almost efficient. In fact the learning period is approximately 3 seconds; i.e. after 3 seconds, the Qlearn signal trajectory will converge to the Pteach signal.

An arbitrary input signal can have a large number of harmonics representing it, but some of them are more important. The generic CPGs, as shown in Figure. 3, consist of four adaptive oscillators which are responsible for searching and finding the most important harmonics constructing the input signal. In Figure. 6, we have shown different state values of adaptive oscillators in the left-hip pitch CPG. Figure. 6.a represents Alpha values, i.e., the amplitudes of each harmonic of the signal. All these Alpha values converge to specific final amplitudes of 31, 13, 5 and 3. Figure. 6.b shows the evolution of Omega values, i.e., frequencies of each harmonic of the signal. Note that the first two adaptive oscillators go down to zero. This means that they act as constant (offset) values in the weighted sum of the Fourier representation. The other two oscillators converge to positive values of 38 and 57. Finally, Figure. 6.c presents the Phi values, i.e., the phases of each harmonic of the signal of the left hip pitch trajectory.
Figure 5. Left Hip Pitch Trajectory generated by our model

Figure 6. a) Alpha values of Left Hip Pitch Trajectory generated by our model b) Omega values of Left Hip Pitch Trajectory generated by our model c) Phi values of Left Hip Pitch Trajectory generated by our model

Figure 7. All the Trajectories of the foot generated by our model

Fig. 7 shows Pteach and Qlearn trajectories. One can see that only four adaptive oscillators can completely represent the input trajectory and we need not have more oscillators to improve the accuracy.
4.2 Results of Online Control of the Robots

In the second phase we import trained state values to the Webots Nao model and design the controller of curvilinear walking in the Robotstadium environment. When the robot decides to walk on the perimeter of the circular curve with radius R, the online trained CPGs start to generate patterns based on the specific R value. In Figure 8, 12 snapshots from the robot’s curvilinear walking in simulation environment are shown. 9 shows such a circular curve on which the robot has traversed. In the first part of the figure, the robot is in the middle of the soccer field. Then it slowly walks and rotates up to the left of the soccer field. The desired path of the robot is a circle with radius R which is compared with the actual path traversed by the robot. The notion that the robot never stops rotating is a key point in this walk. One can see that it is a curvilinear walk from the starting point in the middle to the corner point.

Figure 8. Snapshots of our curvilinear walking in simulation environment

Figure 9. Comparison between actual and desired circular path during a curvilinear walk

Figure 10. Snapshots of curvilinear walking of real Nao in the real world

In order to achieve the absolute parameters of the programming of CPGs, we will transfer them to a real Nao. An online controller described in tables 1 and 2 can be implemented in a real Nao controller module to generate a curvilinear walking. In Figure 10, a curvilinear walk in a real Nao is shown. Sub-figures are numbered 1 to 10 from the starting point to the middle point. This walk goes on to a final point at the end of a circular curve. In this way we obtain a systematic method for building and adjusting the CPG controller on a Nao humanoid robot.

4.3 Analysis of the Sensory Feedbacks

Foot sensor values are the main source of feedback which is used in our model to control curvilinear walking. Figure 11 presents four diagrams of sensory values in the left foot of the robot during a gait. These values are use in Equation 13 to compute $F_{diff}$ value. The diagram of changes in $F_{diff}$ during a gait of curvilinear walking is illustrated in Figure 12. It can be seen that this diagram is partially symmetric, but the two sides are not exactly the same. This means that in a curvilinear walk, the distribution of force values are not exactly similar in the two feet. We extract this difference between both feet and import it to the CPGs’ Equations 13,14. Different phases of walking during a gait are shown in this Figure. These are initial stance, initial contact, initial loading, swing, terminal stance, terminal contact and terminal loading.

Figure 11. Values of the Foot sensors (for the left foot) during a gait
Figure 12. Values of the $F_{\text{diff}}$ during a gait

Figure 13. Modulation of the basic frequency of walking in two cases

4.4 Modulation of Basic Frequencies

One of the most beneficial aspects of CPG-based approaches in bipedal walking is modulation of generating trajectories. Modulation of these trajectories helps the robot to change its speed and style of walking. It can increase or decrease its speed by modulating the basic frequencies of the trained trajectories. Extracting the Omega values of walking trajectories, we can obtain basic frequencies of all the joints. A basic frequency is the greatest common divisor of almost all the frequencies (Omega values) of the trained trajectories. It is possible to find a number which can generate the entire Omega values. Increasing or decreasing this basic frequency would cause modulation of trajectories and lead to increment or decrement of the walking speed. Figure 13 presents this modulation of trained trajectories. We have calculated a basic frequency of the whole system named bf. This bf value has made up of original trajectories in blue colour. Then we have multiplied bf to number 2. The red trajectories are generated in this case which make the robot walk twice as fast as in the original case. Green trajectories are achieved by dividing the bf by 2. We have tested and measured these speeds, and obtained a very smooth and stable curvilinear walking.

5. Conclusion

A new model for generating a curvilinear bipedal walking pattern using a programmable central pattern generator has been introduced which is trained by the Nao basic walking trajectories. This type of locomotion is a beneficial movement in soccer playing Nao robots in RoboCup SPL competitions because it helps them play faster and localizes them in the points they should be at in their team arrangements. They must avoid collision with other soccer playing robots because crashing may be harmful for their hardware and also may make them fall on the ground, which may cause a period of time during which they must stand up and resume playing again. Curvilinear walking assists humanoid robots in walking naturally and efficiently.

A method has been introduced for designing and learning central pattern generators in Nao robots that can learn trajectories in an offline mode and transfer trained parameters to an online controller. The present model is a development of the model of PCPGs which can generate curvilinear bipedal walking trajectories in the online mode. The sum of the Equations has been updated to make programmable CPFGs and enable them to control the hip yaw-pitch joints of Nao to generate curvilinear patterns. In addition, the arm movement trajectories have been generated from a weighted linear sum of the outputs of other CPFGs.
It has been shown in this paper that sensory feedback can be integrated in the trained CPGs to control bipedal rotation during curvilinear walking. Two different sensory data are used: accelerometers and foot-pressure sensors, which are combined and used in the trained generic CPGs. New Equations have been introduced to compute the distinctions of pressures in the two feet and these have been inserted into the CPGs real Equations.

Future works will consist of using this model in real Nao soccer players in Standard Platform leagues in Robocup 2014 competitions. It has been planned to implement the method used in a real Nao and test its efficiency in the real world. It is also intended to compare the method with ZMP-based methods currently in use in real Nao robots.

6. References

[1] Bachar, Y.: Development Controllers for Biped Humanoid Locomotion. School of Informatics, University of Edinburgh (2004).
[2] Beng, T.: Walking Nao Omnidirectional Bipedal Locomotion University of New South Wales (2009).
[3] Buchli, J., L. Righetti, et al.: Engineering Entrainment and Adaptation in Limit Cycle Systems, From biological inspiration to applications in robotics. Biological Cybernetics 95(6) (2006).
[4] Cherubini, A., F. Giannone, et al.: "Policy gradient learning for a humanoid soccer robot." Journal of Robotics and Autonomous Systems 57: 808-818 (2009).
[5] Gams, A., A. J. Ijspeert, et al.: On-line learning and modulation of periodic movements with nonlinear dynamical systems. Auton Robot 27: 3-23 (2009).
[6] Heebel, M., R. Kosse, et al.: Modeling and learning walking gaits of biped robots. IEEE-RAS International Conference on Humanoid Robots, (2006).
[7] Hong, S., Y. Ohy, et al.: An Omni-directional Walking Pattern Generation Method for Humanoid Robots with Quartic Polynomials. IEEE/RSJ International Conference on Intelligent Robots and Systems San Diego, CA, USA (2007).
[8] Ijspeert, A. J.: Central pattern generators for locomotion control in animals and robots: a review. Journal of Neural Networks 21(4): 642-653 (2008).
[9] Kohl, N. and P. Stone: Policy gradient reinforcement learning for fast quadrupedal locomotion. IEEE International Conference on Robotics and Automation (2004).
[10] Kuffner, J. J. and S. Kagami.: Dynamically-stable Motion Planning for Humanoid Robots. Autonomous Robots 12(1): 105-118 (2002).
[11] Marc, P.: Nao programming for the Robotstadium on-line contest. Ecole Polytechnique Federale de Lausanne EPFL Biologically Inspired Robot Group (2010).
[12] Matsubaraa, T., J. Morimob, et al.: Learning CPG-based biped locomotion with a policy gradient method, Robotics and Autonomous Systems, 54: 911-920 (2006).
[13] Niehaua, C., T. Rofer, et al.: Gait optimization on a humanoid robot using particle swarm optimization. IEEE-RAS International Conference on Humanoid Robots (2007).
[14] Pratt, J., P. Dilworth, et al.: Virtual model control of a biped walking robot. IEEE Int'l Conf. on Robotics and Automation 193-198 (1997).
[15] Righetti, L., J. Buchli, et al.: Hebbian learning in adaptive frequency scillators. Physica D: 105-116 (2005).
[16] Righetti, L. and A. J. Ijspeert.: Programmable Central Pattern Generators: an application to biped locomotion control IEEE International Conference on Robotics and Automation Orlando, Florida (2006).
[17] Sato, M., Y. Nakamura, et al.: Reinforcement learning for biped locomotion. International Conference on Artificial Neural Networks (2002).
[18] Shahbazi, H., K. Jamshidi, et al.: Curvilinear Bipedal Walk Learning in Nao Humanoid Robot using a CPG-based Policy Gradient Method international Conference on Mechanical and Aerospace Engineering (ICMAE 2011), Bangkok, Thailand (2011).
[19] Strom, J., G. Slavov, et al.: Omnidirectional Walking Using ZMP and Preview Control for the NAO Humanoid Robot. LNAI 5949, Springer-Verlag Berlin Heidelberg: 378-389 (2009).
[20] Vukobratovic, M. and D. Juricic.: Contribution to the Synthesis of Biped Gait. IEEE Trans. On Bio-Medical Engineering BME-16(1): 1-6 (1969).
[21] Yamasaki, F., K. Endo, et al.: Acquisition of humanoid walking motion using genetic algorithm considering characteristics of servo modules. IEEE International Conference on Robotics and Automation (2002).
[22] Yang, L., C. M. Chew, et al.: Adjustable bipedal gait generation using genetic algorithm optimized Fourier series formulation. IEEE/RSJ International Conference on Intelligent Robots and Systems (2006).
[23] Zielnić ska, T.: Biological inspiration used for robots motion synthesis. Journal of Physiology - Paris 103: Journal of Physiology - Paris (2009).