Automated Generation of CPG-Based Locomotion for Robot Nao

Ernesto Torres and Leonardo Garrido

Tecnológico de Monterrey, Campus Monterrey.
Av. Eugenio Garza Sada 2501.
Monterrey, Nuevo León. México
ernesto.torres.d@gmail.com, leonardo.garrido@itesm.mx

Abstract. This paper presents a solution to the biped locomotion problem. The robot used for the experiments is robot Nao by Aldebaran Robotics and it is simulated in Webots mobile robot simulator. Our method of solution does not requires the dynamic model of the robot, thus making this approach usable to other biped robots. For faster results the number of degrees of freedom is kept low, only six are used. The walking gait is generated using Central Pattern Generators with limit-cycle oscillators. For the oscillator connection weights required for synchronization, a genetic algorithm is implemented. Our solution is generated automatically and the best results allow the robot to walk twice as fast as the Aldebaran's webots walk and four times faster than the default walk in Robotstadium.

Keywords: central pattern generator, genetic algorithm, biped robot, robot nao, robot simulation.

1 Introduction

In this paper we present a solution to create a stable and fast walking gait for the robot Nao [1] in the Webots [5] simulator. This proposed solution allows a Nao robot to walk without the requirement of the dynamic model by using a bio-inspired approach. We suggest the use of Central Pattern Generators (CPGs) based on coupled limit-cycle oscillators together with genetic algorithms (GA). The oscillators generate trajectories that are followed by the robot's servomotors. For the oscillator coupling, we use a GA to optimize the synchronization parameters. This work also minimizes the required parameters for the optimization to provide fast results with a relatively low number of evaluations.

We decided to use CPGs [10] to create a controller that is equivalent in the way humans and animals generate locomotion. The CPG is the low level locomotion controller and generates the motion type that a higher level controller requires, such as the interaction between the spinal cord and the brain. We selected genetic algorithms for being very good at finding solutions when the evaluation is simple, in this case, the distance traveled. Both CPG and GA were used to generate the controller in a bio-inspired approach.

T. Röfer et al. (Eds.): RoboCup 2011, LNCS 7416, pp. 461–471, 2012.

This paper is divided in six sections. Related Work section gives a brief summary of the state-of-the-art in biped locomotions controllers. In section 3, we describe the CPGs and the oscillator used. In section 4, the Nao robot is introduced together with Webots simulator, also includes the connections proposed for the oscillator coupling and the genome used in the GA. In Section 5, we show the results obtained with this approach, the optimized genome, the oscillation behavior and the walking gait obtained. Finally, Conclusion and Future Works section makes a brief summary on this work and our current and future research projects.

2 Related Work

Designing and implementing locomotion controllers for a robot is a complex task that often requires the knowledge of the dynamic model of the robot in order to generate robust and practical gaits. This is a task far from trivial, and can be approached in many different ways. The most common approach is using control theory. By manipulation of limb trajectories, it creates a desired motion and by using a controller, the balance is achieved.

Several works implement Zero-Movement Point(ZMP) [17] for the generation of trajectories that allows the center of mass to remain inside a stability polygon and thus avoid falling. In [16], ZMP was used to create an engine for an omnidirectional walk. For their dynamic model they use an inverted pendulum with all the mass in the CoM, then a preview controller generates dynamically balanced center of mass trajectories.

Another common approach is the use of heuristic methods to avoid using the dynamic model but still being able to generate a successful motion. Heuristic approaches include: genetic algorithms, reinforcement learning, policy gradient, among others. In [9], genetic algorithms are used together with partial Fouries series and they generate the trajectories based mainly on the CoM to monitor the stability of the gait. In [12], reinforcement learning is used for a robot to learn where to place the swing leg. They also modify the desired walking cycle frequency based on online measurements. Finally, in [3], an extension of the classic Policy Gradient algorithm that takes into account parameter relevance is used. It allows for better solutions when only a few experiments are available.

A third approach exists, a bio-inspired approach. Instead of creating new methods for achieving locomotion, this approach heavily relies on the observation of the living organisms in nature. Works using bio-inspired approaches use Central Pattern Generators (CPGs) as its foundation [6]. In [8], the CPG network consists of the Matsuoka neuron model, and is introduced to realize the locomotion of a bipedal robot, they also use genetic algorithms in several steps to find out large number of parameters according to structure of CPG network. In [7], an oscillator is proposed for a salamander robot to generate the transition between swimming and walking. This same oscillator was used in [2] to generate locomotion for modular robots and in [11] to generate trajectories for the biped robot hoap2. In bio-inspired models, genetic algorithms are often the algorithm of choice for parameter optimization since it is based on evolution.

3 CPG Description

CPGs are neural networks found in vertebrate and invertebrate animals that produce rhythmic outputs without rhythmic inputs. Rhythms in nature are abundant, and are common in animal activities such as the heartbeat, breathing, chewing and digesting.

A CPG can be implemented with oscillators as the basic neural unit and create the connected network by oscillator coupling. An oscillator is a system that executes a periodic behavior. For example, a swinging pendulum executes a periodic behavior once it is released at a certain height and it returns to the same point every cycle. A common oscillator has a characteristic period.

A special form of oscillator often found in nature is a limit-cycle oscillator. A limit-cycle not only has a characteristic period but also a characteristic amplitude. If the limit cycle is affected by some perturbations it is able to return to its original trajectory after a given time. Those characteristics are important for locomotion since it makes the system resistant to small perturbations. A more in-depth information about oscillators and limit-cycles can be found in [15].

The oscillator used for the CPG was proposed by Professor A. Ijspeert from BIRG[7]. It is a non-linear oscillator and has the interesting property of having the limit cycle behave like a sinusoidal signal with an amplitude $\sqrt{E}$ and a period $2\pi\tau$.

$$\tau\dot{v} = -\alpha\frac{x^2 + v^2 - E}{E}v - x \tag{1}$$

$$\tau\dot{x} = v \tag{2}$$

The variable x represents position and v represents velocity. Variables τ,α and E are positive constants. By modifying τ, the period of the oscillator can be manipulated and changing α changes the system speed convergence. The energy of the system is represented as E. For oscillator coupling:

$$\tau\dot{v_i} = -\alpha\frac{x_i^2 + v_i^2 - E}{E}v_i - x_i + \sum a_{ij}x_j + b_{ij}v_j + \sum c_{ij}sj \tag{3}$$

$$\tau\dot{x_i} = v_i \tag{4}$$

Where a_{ij} and b_{ij} represent how oscillator j influences oscillator i. The last summatory allows for sensory input.

A previous work by Mojon [11] suggests a modification on the oscillator to allow the connection strength to be independent from the energy of the oscillators. This is achieved by normalizing the connections:

$$\tau\dot{v_i} = -\alpha\frac{x_i^2 + v_i^2 - E}{E}v_i - x_i + \sum \frac{a_{ij}x_j + b_{ij}v_j}{x_j^2 + v_j^2} \tag{5}$$

$$\tau\dot{x_i} = v_i \tag{6}$$

The last equations represent the final model of the oscillator used in this work. One oscillator per DoF was used.

4 Experimental Setup

The robot used in this paper is the robot Nao from Aldebaran Robotics [1]. This robot is now used in the official Robocup Standard Platform League(SPL) [13,14]. Since the robot requires no hardware modifications is greatly useful for software development. Nao has 21 degrees of freedom(DoF) which makes it highly customizable and has a high number of possible movements. For this paper we tried to reduce the search space all we could to be able to obtain fast results with a relatively low number of experiments. For this reason only 6 DoF were used, 3 from each leg: AnklePitch, KneePitch and HipPitch. These three were selected because the propulsion force for walking is given only in the sagittal plane. Any extra DoF could be used not for propulsion but for stabilization.

The experiments were developed and tested in Webots simulator [5] by Cyberbotics. Inspired by the online robot soccer competition, Robotstadium [4], this software was selected to develop and implement a locomotion controller for Nao robot. The version of the software used is 6.2.4 PRO and the model used is the NaoV3R used for Robotstadium 2009/2010.

Connections subsection includes the DoF used, the oscillators network and the parameters required. Finally, the Genome and Genetic Algorithm section show how we optimized the CPG.

4.1 Connections

Since we use six DoF, six oscillators were used. Each DoF should require 6 or 8 parameters: A(Amplitude), $X0$ for a non-cero centered oscillation, a_{ij} and b_{ij} for a single connection or a_{ij} ,b_{ij}, a_{ik} and b_{ik} for a double connected oscillator, τ for period, and α for convergence speed between connections.

To enable every oscilator to affect and be affected, the links are double connected. With the connections proposed in Fig. 1, we have 4 DoF with 8 parameters and 2 DoF with 6, this makes 44 total parameters to optimize. Forty-four parameters represents a big search space considering each parameter could take real values from $[-1.0, 1.0]$. To reduce the number of free parameters, τ and α are fixed, reducing to 4 or 6 parameter per DoF for a total of 32. We set $\tau \approx 1hz$ and $\alpha = 1$ for maximum convergence speed. Previous work with this oscillator by Mojon [11] gave us some methods for reducing further the search space by obtaining the weight paremeters to force 0, $\frac{\pi}{2}$, π and $\frac{3\pi}{2}$ phase difference. For the first reduction, a certain phase can be forced by controlling a and b parameters. Instead of single evolving every oscillator a simetry is asumed, and only one leg is optimized while the other leg use the same parameters. The trick for this concists in forcing a π phase difference between the left hip and the right hip. With this reduction instead of 32, there are only 16 parameters.

The final model is composed by 3 oscillators. The hip oscillator is connected to the other hip and to its corresponding knee. The knee oscillator is connected to its correspondent hip and ankle. Finally, the ankle is only connected to the knee. Table 1 summarizes the possible values the oscillators are using. Extra adaptations include the maximum range for the amplitude parameter. We allow

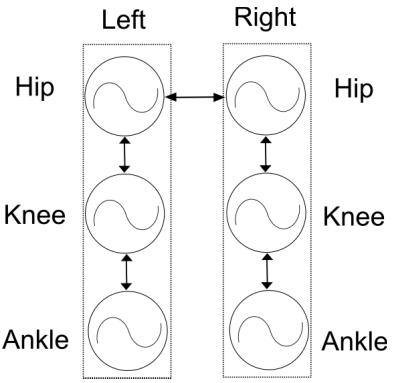

Fig. 1. The Oscillator connections. All the connections are bi-directional

Table 1. Oscillator values. The value range for a variable is displayed in each cell, missing values are values not required.

	$x0$	A	$a1$	$a2$	$b1$	$b2$
Hip	$[-1.0, 1.0]$	$[-1.0, 1.0]$	0	$[-1.0, 1.0]$	$[-1.0, 0.0]$	$[-1.0, 1.0]$
Knee	$[-1.0, 1.0]$	$[-1.0, 1.0]$	$[-1.0, 1.0]$	$[-1.0, 1.0]$	$[-1.0, 1.0]$	$[-1.0, 1.0]$
Ankle	$[-1.0, 1.0]$	$[-1.0, 1.0]$	$[-1.0, 1.0]$		$[-1.0, 1.0]$	

the max amplitude be up to $\frac{\pi}{2}$ for each oscillator. And for the connections strength the maximum allowed is 0.7 to have a smoother synchronization [2]. If the maximum connection strenght is too high the oscillators may not reach synchronization.

4.2 Genome and Genetic Algorithm

The genome used is displayed in Table 2. It corresponds to the values the genetic algorithm optimizes and it is the concatenation of the parameters required by the oscillators. The first five values correspond to the hip, in which $a1$ is always zero, thus ommited from the genome. The next six values corresponds to the knee oscillator and the last four are required by the ankle oscillator. All the parameters can take values from $[-1.0, 1.0]$. The only exception is $b1$ from the hip which can only take values from $[-1.0, 0.0]$.

The $X0$ value indicates the bias for the oscillator to allow for non-zero centered oscillation. A is the amplitude of the oscillation, internally it can only take

Table 2. Genome

X0	A	a2	b1	b2	X0	A	a1	a2	b1	b2	X0	A	a1	a2

positive numbers. The π phase difference between legs is forced by making hip values $a1 = 0$ and $b1 = [-1.0, 0.0]$.

For the Genetic Algorithm(GA) implementation we used GAlib [18]. All the genes are real value numbers and are represented as a float data type. The genetic algorithm is a GASimpleGA which corresponds to the genetic algorithm proposed by Holland.

Tournament selection was used. Two chromosomes from the population are randomly selected and the one with best fitness is selected.

The crossover type used was $GARealOnePointCrossover$, which randomly selects one crossover point to swap genes from the parents. The probability of crossover was 0.90. And for mutation it was used $GARealUniformMutator$, which randomly selects a new value from the alleles $[-1.0, 0.0]$. Mutation probability was set to 0.01.

The population size is 30 individuals, twice the size of the genes. The GA was configured to run until the convergence of the population. A 95% convergence ratio was used as criteria for stopping the GA and it had to be concistent for fifty generations. Elitism was used to preserve the best individual from each generation.

Only the best individual is obtained as a result of each GA execution. Several GAs were executed secuentially and all the best individuals were recorded. The fitness evaluation is the following:

$$Fitness = \begin{cases} 0 & \text{if } Distance < 20 \\ Distance * \frac{StandingSimulationSteps}{TotalSimulationSteps} & \text{otherwise} \end{cases} \quad (7)$$

The *Distance* was measured in centimeters. *StandingSimulationSteps* is the number of simulation steps until the robot falls or the 30 second timer expires. *TotalSimulationSteps* are the total simulation steps in 30 seconds.

5 Experimental Results

First of all, we have three key aspects involved which required careful tunning for optimal performance. These aspects are: the fitness function, the number of oscillators and the phase difference between oscillators.

Several fitness functions were tested, because a bad fitness generates a bad behavior or at least an unwanted behavior. At the beginning, when the fitness only measured the *Distance*, the robot learned to jump to reach for a larger distance instead of generating trajectories that reward the robot later by walking. Another approach used for the fitness function was to allow the robot to walk to anywhere it wants and still obtain a positive fitness. For example, if the robot moved backwards, the distance was multiplied by 0.50 to penalize the wrong direction, but it still receive a non-zero fitness evaluation. Later, it was observed that it was better to give a 0.0 fitness whenever the robot did not move forward. Another important aspect was to encourage the robot to avoid falling. If the reward is too big, the robot learns to reduce the amplitude of the oscillations and it avoids falling, but it does not move forward. The fitness function had to be

a balanced equation to give a reward for moving forward but not give too much reward for not falling. To address this, the minimum valid distance required is 20 centimeters.

The number of oscillators and the phase difference between oscillators was fixed to reduce to the maximum the number of free parameters and to allow a faster convergence for the genetic algorithm. Preliminary tests were made with 12 oscillators instead of the 6 presented in this paper. The servos used for each leg were: 3 from each hip, 1 for the knee and 2 from the ankles. The robot was able to walk small distances, but the walking speed was far from good and most of the times the gait of the robot was a weird non human-like movement. For that reason, the number of required oscillators was minimized to generate a walking gait with the minimum required parameters.

The oscillation values are shown in Fig. 2. The servo motors start at 0.0 where the robot is in resting position and later begins locomotion with the synchronization of all the servos. The resulting genome is displayed in Table.3. All the values are within $[-1.0, 1.0]$. As mentioned in the Experimental Setup section, the first five values correspond to the hip, in which $a1$ is always zero, thus ommited from the genome. The next six values corresponds to the knee oscillator and the last four are required by the ankle oscillator

The best results are shown in Fig. 3 with a fitness of almost 350. The average fitness is affected by the mutation and crossover in which some individuals obtain a bad fitness. As it can be appreciated, the best results are obtained aproximately from the 50th generation where it converges. In order to obtain satisfactory results, the genetic algorithm was restarted whenever it converges to increment

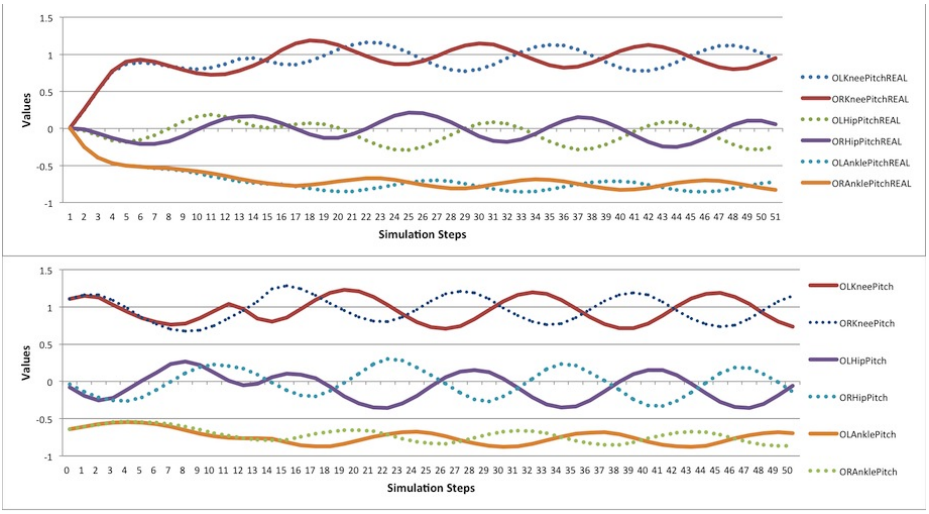

Fig. 2. Comparative between the oscillations generated by the CPG (bottom) and the real oscillations by the servomotors (top)

Table 3. Solution obtained for the CPG (Split in two for display purposes)

X0	A	a2	b1	b2	X0	A	a1
-0.118	0.132	-0.627	0.314	-0.787	0.939	0.119	-0.178

a2	b1	b2	X0	A	a1	a2
0.636	-0.164	0.040	-0.787	-0.049	0.528	0.072

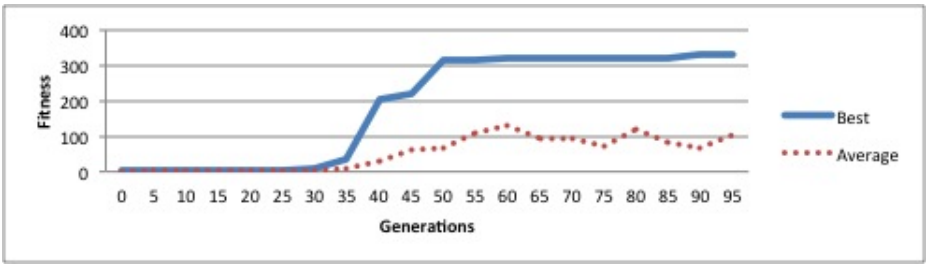

Fig. 3. Fitness per Generation displaying the best individual and the average for each generation

the number of posible solutions. Depending on the population sometimes the convergence can take from several hours to minutes. The Nao learned to walk making small steps to keep balance and avoid falling. It can be appreciated in Fig.4. Its double support phase is very brief, balancing in one feet most of the time. Since the CPG has an oscillator behavior, the cycle is repeated the required times. Once it breaks the standing pose, it can cross the whole field without problem.

In order to compare the results obtained from the GA in a more graphical way, a special world in Webots was programmed to allow three different robot controllers to compete on a race to cross half of the soccer field. The controllers were: the one generated in this paper, the default walking motion from Robotstadium and the default walking controller from the Nao provided as default by Aldebaran Robotics.

The results of the race can be appreciated in Fig.5. From top to bottom: Aldebaran's controller, our controller and Robotstadium's controller. Our controller can walk four times faster than the default controller in Robotstadium and aproximately twice as fast than the Nao default walk by Aldebaran Robotics in naoqi 1.6. The results obtained are satisfactory since our controller is an open loop that may be improved further with reflexes and/or a stability controller.

An additional aspect to have in consideration is the first step of the robot, since the resting position is outside the walking trajectories. Several ways to help with the first step were implemented. The first one was to allow some time for the CPG to synchronize and then connect the servos. It wasn't as useful as we thought and the results were negative. Another variation was to let them synchronize and then when the values were closer to the $X0$ the servos

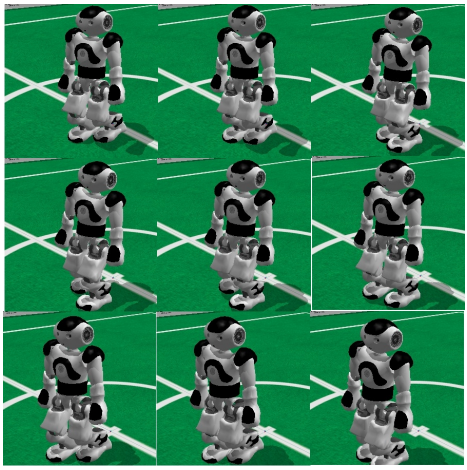

Fig. 4. Nao walking. Can be appreciated from top-left to bottom-right.

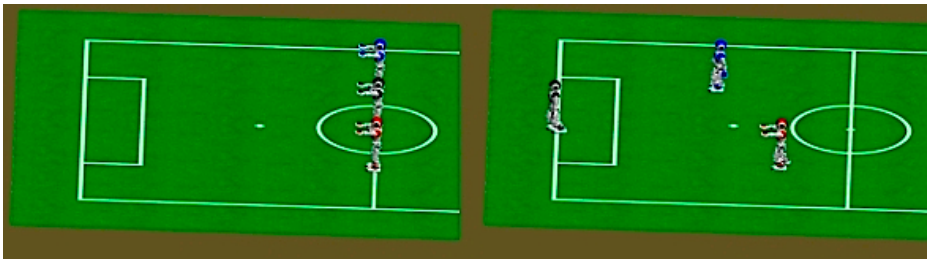

Fig. 5. Race between the controllers. Top blue: Aldebaran's controller. Center black: our controller. Bottom red: Robotstadium's default walk

were connected, but the same negative results were obtained. Finally the last approach was to initiate the servos at the same time of the CPG and allow it to synchronize.

About the time requiered for good individuals, in one 16-hour session good individuals were generated and tested. Our Webots PRO license allows to run in fast mode which gave us 12x the speed of a simulation.

6 Conclusions and Future Work

As the previous section showed, a good locomotion controller can be created by using CPGs. Limit-cycle oscillators provide some resistance to perturbations and allow the robot to generate by itself the first step to break the rest position and walk forward. It was shown that a walking gait can be generated by using only the pitch servomotors: hipPitch, kneePitch and anklePitch. Another advantage

of using a low number of oscillators is that the size of the search space is reduced, obtaining faster results.

Research in progress includes the automated generation of side steps. In preliminary experiments we have obtained very good results, but are still not conclusive. Additional experiments will include rotational walk and hopefully an omnidirectional walk with the integration of all the results. At the same time, we are working in optimizing the algorithm in order implement it in the real Nao.Future work will include an implementation in the physical Nao to replace the actual default walk for a hopefully better gait. Additional future work includes the evolution of an adaptive controller simultaneously with the CPG to allow a closed-loop control and achieve faster and more stable gaits.

References

1. Aldebaran Robotics: Nao, official player of the RoboCup since 2008 (2011), http://www.aldebaran-robotics.com/en/node/1168
2. Bourquin, Y.: Self-Organization of Locomotion in Modular Robots. Dissertation, Ecole Polytechnique Federale de Lausanne - EPFL (2004), http://birg.epfl.ch/page53073.html
3. Cherubini, A., Giannone, F., Iocchi, L., Lombardo, M., Oriolo, G.: Policy gradient learning for a humanoid soccer robot. Robotics and Autonomous Systems 57(8), 808–818 (2009), http://www.sciencedirect.com/science/article/B6V16-4W0R0MN-3/2/1de409f7de564e83189ac81bf3a6ca5f , Humanoid Soccer Robots
4. Cyberbotics: Robotstadium: online robot soccer competition (2011), http://robotstadium.org/
5. Cyberbotics: Webots: mobile robot simulation software (2011), http://www.cyberbotics.com
6. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: A review. Neural Networks 21(4), 642–653 (2008), http://www.sciencedirect.com/science/article/B6T08-4SH6B9F-2/2/2e0a2fdad02d315becc218a6602f054d , robotics and Neuroscience
7. Ijspeert, A.J., Cabelguen, J.M.: Gait transition from swimming to walking: investigation of salamander locomotion control using nonlinear oscillators. In: Proceeding of Adaptative Motion in Animals and Machines (2003)
8. Inada, H., Ishii, K.: Bipedal walk using a central pattern generator. International Congress Series 1269, 185–188 (2004), http://www.sciencedirect.com/science/article/B7581-4D338VC-1K/2/ac44f599e008ec39e662ff3e41763cb1 brain-Inspired IT I. Invited papers of the 1st Meeting entitled Brain IT 2004
9. Kong, J.S., Lee, B.H., Kim, J.G.: A study on the gait generation of a humanoid robot using genetic algorithm. In: SICE 2004 Annual Conference, vol. 1, pp. 187–191 (August 2004)
10. MacKay-Lyons, M.: Central Pattern Generation of Locomotion: A Review of the Evidence. Physical Therapy 82(1), 69–83 (2002), http://ptjournal.apta.org/content/82/1/69.abstract

11. Mojon, S.: Using nonlinear oscillators to control the locomotion of a simulated biped robot. Diploma thesis. Ecole Polytechnique Federale de Lausanne - EPFL (2004), http://birg.epfl.ch/page44565.html
12. Morimoto, J., Cheng, G., Atkeson, C., Zeglin, G.: A simple reinforcement learning algorithm for biped walking. In: Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2004, April 1-May, vol. 3, pp. 3030–3035 (2004)
13. RoboCup Organization: Robot world cup initiative (2011), http://www.robocup.org
14. RoboCup Organization: Standard Platform League (2011), http://www.tzi.de/spl/bin/view/Website/WebHome
15. Strogatz, S., Stewart, I.: Coupled oscillators and biological synchronization. Scientific American 269(6), 68 (1993)
16. Strom, J., Slavov, G., Chown, E.: Omnidirectional Walking Using ZMP and Preview Control for the NAO Humanoid Robot. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS (LNAI), vol. 5949, pp. 378–389. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-11876-0_33
17. Vukobratovic, M., Borovac, B., Surdilovic, D.: Zero-Movement Point - Proper Interpretation. Submitted to International Journal of Robotics Research (2004)
18. Wall, M.: Galib: A c++ library of genetic algorithm component (2011), http://lancet.mit.edu/ga/