Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Programmable Central Pattern Generators: an application to biped locomotion control

Ludovic Righetti and Auke Jan Ijspeert
Biologically Inspired Robotics Group
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL) - Switzerland
Email: ludovic.righetti@a3.epfl.ch, auke.ijspeert@epfl.ch

***Abstract*— We present a system of coupled nonlinear oscillators to be used as programmable central pattern generators, and apply it to control the locomotion of a humanoid robot. Central pattern generators are biological neural networks that can produce coordinated multidimensional rhythmic signals, under the control of simple input signals. They are found both in vertebrate and invertebrate animals for the control of locomotion. In this article, we present a novel system composed of coupled adaptive nonlinear oscillators that can learn arbitrary rhythmic signals in a supervised learning framework. Using adaptive rules implemented as differential equations, parameters such as intrinsic frequencies, amplitudes, and coupling weights are automatically adjusted to replicate a teaching signal. Once the teaching signal is removed, the trajectories remain embedded as the limit cycle of the dynamical system. An interesting aspect of this approach is that the learning is completely embedded into the dynamical system, and does not require external optimization algorithms.**

We use our system to encapsulate rhythmic trajectories for biped locomotion with a simulated humanoid robot, and demonstrate how it can be used to do online trajectory generation. The system can modulate the speed of locomotion, and even allow the reversal of direction (i.e. walking backwards). The integration of sensory feedback allows the online modulation of trajectories such as to increase the basin of stability of the gaits, and therefore the range of speeds that can be produced.

I. Introduction

This contribution presents a programmable Central Pattern Generator (CPG) for the online generation of periodic trajectories, and its application to the control of biped locomotion in a simulated Hoap-2 robot. Our work is motivated by the growing interest in biologically inspired control of autonomous robots and especially the use of CPGs as a new paradigm to generate coordinated periodic movements.

As an alternative to methods using pre-recorded trajectories (e.g. ZMP-based [1]) and methods using heuristic control laws (e.g. Virtual Model control [2]), CPGs encode rhythmic trajectories as limit cycles of nonlinear dynamical systems, typically systems of coupled nonlinear oscillators. This offers multiple interesting features such as the stability properties of the limit cycle behavior (i.e. perturbations are quickly forgotten), 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. Interesting examples of CPGs applied to biped locomotion include [3], [4].

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 very few methodologies to construct a CPG for generating an arbitrary periodic signal. In [5], a method is presented which uses regression techniques to shape limit cycles of nonlinear dynamical systems, but that method requires preprocessing the teaching signal to extract its main period.

In this contribution, we present a novel system of coupled adaptive oscillators that can learn arbitrary periodic signals in a supervised learning framework. An interesting aspect of our approach is that the learning is completely embedded into the dynamical system, and does not require any external regression or optimization algorithms, nor any preprocessing of the teaching signal. The system essentially implements a kind of dynamic Fourier series representation. We apply our system to the control of locomotion of a 23-DOF simulated humanoid robot. Results are presented demonstrating how pre-recorded walking trajectories can be learned with the system and then modulated online using the CPGs limit cycle properties. In particular, we show how sensory feedback can be integrated into the CPGs to increase the basin of stability of the gaits, and how the speed of walking can be modulated and even reversed by using a single control parameter.

II. Generic Central Pattern Generators

In this section we present our model of a generic CPG that we use to encode periodic trajectories. First, we present in details the architecture of the CPG which is made of adaptive oscillators and then we discuss the intrinsic properties of the system that makes it suitable for periodic movement control.

A. Architecture of the CPG

The basic building block of our generic CPG is the adaptive frequency Hopf oscillator, which we present in the following. Then we present the complete architecture of the CPG.

1) Adaptive frequency Hopf oscillator: Our CPG architecture is made of coupled adaptive frequency Hopf oscillators, which are modified Hopf oscillators that we developed in [6], [7]. These oscillators have the property that they can learn the frequency of a periodic input signal without any external optimization process fixed. Generally, the frequency of an oscillator is controlled by some parameter. In [7], we changed this parameter into a new state variable that has a

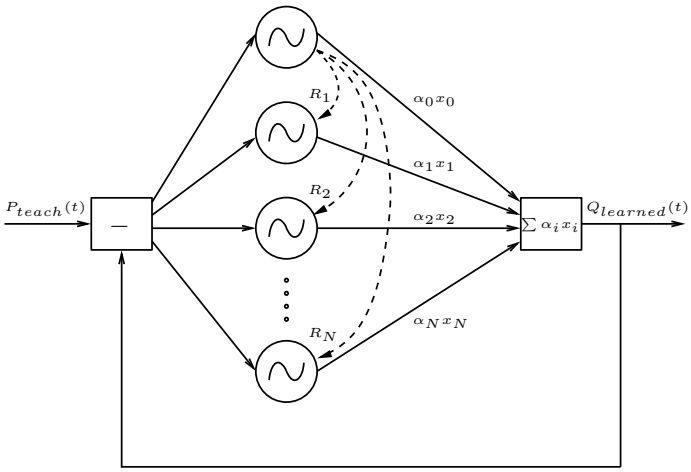


Fig. 1. Structure of the network of adaptive Hopf oscillators. Each oscillator receives the same learning signal $F(t) = P_{teach}(t) - \sum_i \alpha_i x_i$, which is the difference between the signal to be learned, $P_{teach}(t)$, and the signal already learned, $Q_{learned}(t) = \sum_i \alpha_i x_i$. Then all the oscillators (except oscillator 0) receive the scaled phase input R_i from oscillator 0. Refer to Equations (4)-(8) and to the text for more details.

general evolution rule. Then we proved that when perturbed by a periodic input, this new state variable will converge to one of the frequency components of the periodic input. Thus the oscillator will learn and, of course, synchronize to the perturbing input. The adaptation is an intrinsic property of the oscillator, no supervisor or external processing is needed. After convergence, if the input signal disappears, the learned frequency stays encoded in the system. The equation of this oscillator is as follow

$$\dot{x} = \gamma(\mu - r^2)x - \omega y + \epsilon F(t) \quad (1)$$
$$\dot{y} = \gamma(\mu - r^2)y + \omega x \quad (2)$$
$$\dot{\omega} = -\epsilon F(t)\frac{y}{r} \quad (3)$$

where $r = \sqrt{x^2 + y^2}$, μ controls the amplitude of the oscillations, γ controls the speed of recovery after perturbation, ω controls the frequency of the oscillations, $F(t)$ is a periodic input to which the oscillator will adapt its frequency and $\epsilon > 0$ is a coupling constant. Its frequency will adapt to one of the frequency component of the input $F(t)$. The frequency component adapted will depend on the initial conditions for ω. A more detailed discussion about adaptive frequency oscillators is given in [7].

2) Generic CPG: The basic idea for constructing the generic CPG is to use coupled adaptive oscillators to reproduce a periodic signal [8]. The output of a CPG is usually multidimensional but in this section we present a network of coupled oscillators to encode one dimension. However, we will show in Section III how we can use several coupled generic CPGs to encode multidimensional trajectories.

The adaptive property of the oscillators is used for learning the different frequency components of the periodic teaching signal. As the oscillations of the Hopf oscillator are harmonic, an appropriate linear combination of several Hopf oscillators could reproduce any periodic input signal. It would act as a dynamic Fourier series representation, each oscillator encoding one frequency component of the teaching signal.

We associate to each adaptive oscillator a variable representing the amplitude of the learned frequency, then the output of the network will be the weighted sum of the outputs of the oscillators with the associated amplitude variables. By using a negative feedback loop, we can substract the already learned frequencies from the teaching signals and the oscillators that have not yet converged to a stable frequency can adapt to the remaining frequency components.

We also associate to each oscillator a variable encoding for the phase difference between the oscillator and the first oscillator of the network, thus enabling us to reproduce any phase relationship between the oscillators. Figure 1 shows the structure of the network. The equations describing this CPG are as follow

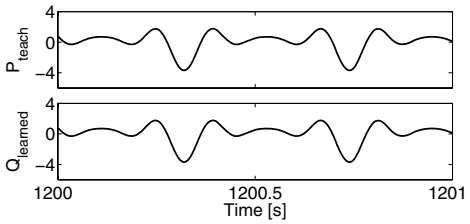
$$\dot{x}_i = \gamma(\mu - r_i^2)x_i - \omega_i y_i + \epsilon F(t) + \tau \sin(\theta_i - \phi_i) \quad (4)$$
$$\dot{y}_i = \gamma(\mu - r_i^2)y_i + \omega_i x_i \quad (5)$$
$$\dot{\omega}_i = -\epsilon F(t)\frac{y_i}{r_i} \quad (6)$$
$$\dot{\alpha}_i = \eta x_i F(t) \quad (7)$$
$$\dot{\phi}_i = \sin\left(\frac{\omega_i}{\omega_0}\theta_0 - \theta_i - \phi_i\right) \quad (8)$$

with

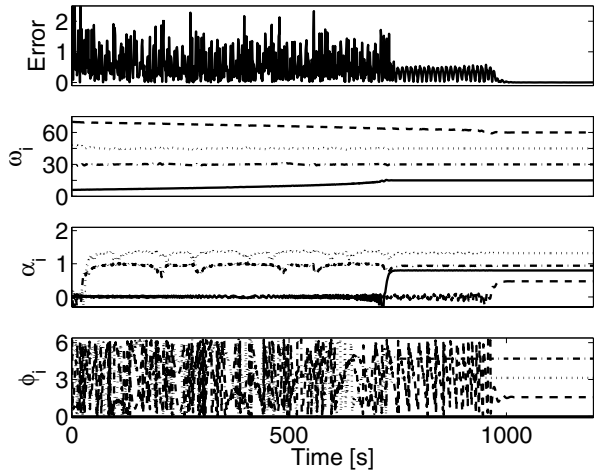
$$\theta_i = \mathrm{sgn}(x_i)\cos^{-1}\left(-\frac{y_i}{r_i}\right) \quad (9)$$
$$F(t) = P_{\mathrm{teach}}(t) - Q_{\mathrm{learned}}(t) \quad (10)$$
$$Q_{\mathrm{learned}}(t) = \sum_{i=0}^{N} \alpha_i x_i \quad (11)$$

where τ and ϵ are coupling constants and η is a learning constant. The output of the CPG, Q_{learned}, is the weighted sum of the outputs of each oscillator. $F(t)$ represents the negative feedback, which in average is the remaining of the teaching signal $P_{\mathrm{teach}}(t)$ the CPG still has to learn. α_i represents the amplitude associated to the frequency ω_i of oscillator i. Its equation of evolution maximizes the correlation between x_i and $F(t)$, which means that α_i will increase only if ω_i has converged to a frequency component of $F(t)$ (the correlation will be positive in average) and will stop increasing when the frequency component ω_i will disappear from $F(t)$ because of the negative feedback loop. ϕ_i is the phase difference between oscillator i and 0. It converges to the phase difference between the instantaneous phase of oscillator 0, θ_0, scaled at frequency ω_i and the instantaneous phase of oscillator i, θ_i. Each adaptive oscillator is coupled with oscillator 0, with strength τ to keep correct phase relationships between oscillators, using the well known Kuramoto coupling scheme [9] to achieve phase synchronization. We mention that with this coupling scheme, the system is more than just a dynamic Fourier series representation because the oscillators can have any phase relationship and not only 0, $\frac{\pi}{2}$, π or $\frac{3\pi}{2}$ phase differences.

With this generic architecture, we are able to learn any periodic input signal. We just have to provide P_{teach} the periodic trajectory we want to learn as input and integrate the system of equations. After convergence, we can set $F(t) = 0$



(a) Result of learning



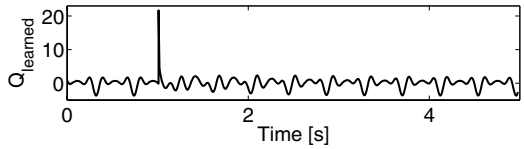
(b) Evolution of the state variables of the CPG

Fig. 2. Figure 2(a) shows the input signal to learn, P_{teach}, in the upper graph and the result of learning $Q_{learned}$ in the lower graph. It is obvious that the network correctly learned the input pattern. Figure 2(b) shows the evolution of the state variables of the generic CPG during learning of an input signal ($P_{\text{teach}} = 0.8\sin(15t) + \cos(30t) - 1.4\sin(45t) - 0.5\cos(60t)$) and the evolution of the error of learning. The upper graph is a plot of the error, defined by $error = \|P_{teach} - Q_{learned}\|$. The 3 other graphs show the evolution of the frequencies, ω_i, the amplitudes, α_i and the phases, ϕ_i. The variables for each oscillator are plotted, variables of oscillator 0 are the plain lines, variables for oscillator 1 are the dotted-dashed lines, variables for oscillator 2 are the dotted lines and the dashed lines represent oscillator 3. The initial conditions are $\alpha_i(0) = \phi_i(0) = 0$, $x_i(0) = 1$, $y_i(0) = 0\ \forall i$, $\mu = 1$, $\gamma = 8$, $\epsilon = 0.9$, $\eta = 0.5$ and $\tau = 2$. The frequencies $\omega_i(0)$ are uniformly distributed from 6 to 70.

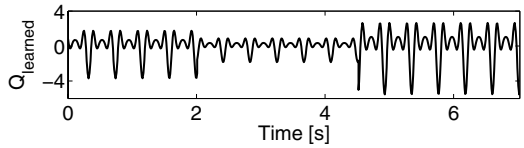
(no more input nor feedback loop) and the periodic signal stays encoded into the network of oscillators. The learning process is embedded in the equations, there is no need of any external optimization or learning algorithm. In Section III we will see how this concept of generic CPG can be extended to learn multidimensional signals.

B. Properties of the generic CPG

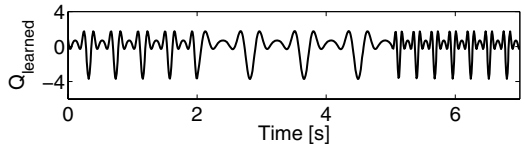
In this section, we present a numerical experiment where the generic CPG learns a simple signal $P_{\text{teach}} = 0.8\sin(15t) + \cos(30t) - 1.4\sin(45t) - 0.5\cos(60t)$. The network we use is composed of 4 oscillators. Figure 2 shows the result of the experiment. An interesting aspect of this generic CPG is that the frequencies of the oscillators are first adapted, each oscillator converges to one of the frequency component 15, 30, 45 and 60. Only when an oscillator matches the frequency of the teaching signal is the corresponding amplitude adapted and then the corresponding frequency component disappears from the signal $F(t)$, as can be seen by the sudden decrease in the error. The phase variables stabilize when the involved oscillators have their frequencies correctly tuned. After learning, the periodic signal is encoded in the network of oscillators, as can be seen in Figure 2(a).



(a) Stability against perturbation



(b) Modulation of amplitude



(c) Modulation of frequency

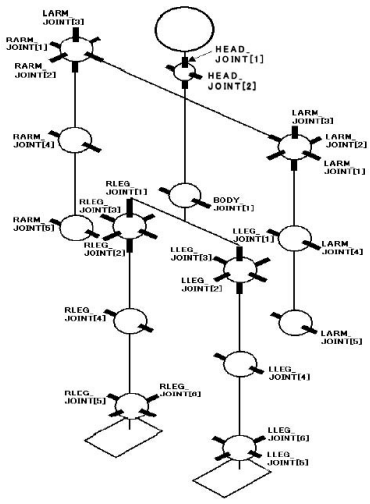
Fig. 3. Figure 3(a) presents the evolution of the output of the generic CPG when perturbed. At time $t_p = 1$ a perturbation occurs on all the oscillators of the CPG. We clearly see that the CPG quickly recovers its original behavior, thus proving the stability properties of the system. Figure 3(b) shows the behavior of the system when the amplitude $\vec{\alpha}$ is changed. At time $t = 2$, the amplitude is divided by 2 and at time $t = 4.5$ the amplitude is multiplied by 3. Figure 3(c) shows the behavior of the network when the frequency $\vec{\omega}$ is changed. At time $t = 2$ the frequency is divided by 2 and at time $t = 5$ frequency is multiplied by 3. In both graphs, we can notice the smoothness of the trajectory when the parameters are changed.

If there are not enough oscillators to code for all the frequency components of the teaching signal, the system will only learn the frequency components with the more power. Thus, the learned trajectory will only be an approximation of the teaching one. However, if there are more oscillators than frequency components to learn, either some oscillators will not converge to any frequency and their contribution to the learned signal will be null ($\alpha = 0$) or some frequency components will be coded by several oscillators and the sum of the corresponding α_i will match the amplitude of the frequency component.

This generic CPG possesses intrinsic properties of stability that are inherent to the Hopf oscillator, which has a structurally stable limit cycle. The CPG can thus produce trajectories that are stable to perturbations. This can be useful when integrating sensory feedback in the CPG to be sure that the sensory information will be forgotten as soon as it disappears from the environment.

Another important aspect of the CPG is that it allows easy modulation of the amplitude and the frequency of the trajectory. Since the frequency and amplitude are linearly related to the vectors $\vec{\omega}$ and $\vec{\alpha}$, simple modulation of these vectors can generate an infinite variation of stable trajectories from the learned input. Because of the properties of coupled oscillators, modulation of these parameters is always smooth and thus interesting for trajectory generation in a robot. Some

(a) The Hoap-2 Robot



(b) The DOFs of the Hoap-2

Fig. 4. Real Hoap-2 robot (Fig. (a)) and schematic of its DOFs (Fig. (b)), this pictures were taken from [10]. We can directly see which DOF the CPG of Figure 5 controls on the schematic.

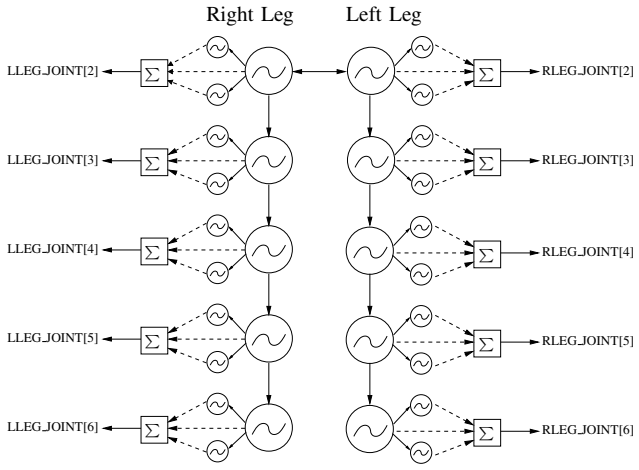


Fig. 5. Structure of the CPG for the humanoid. We use a generic CPG as presented in Section II for each DOF of the legs. We also add state variables that will learn the phase differences between the generic CPGs of the legs (the descending arrows). Antisymmetric coupling is also done between the 2 legs through the main oscillator of the first DOF of each leg (horizontal arrow). The trajectories generated for each DOF is the weighted sum of the corresponding 3 oscillators.

of these properties are shown in Figure 3.

We have now introduced our generic CPG that can encode periodic inputs as stable limit cycles. In the next section, we show an application of this generic CPG as a controller for a humanoid robot. To prove the usefulness of the architecture, we apply it to the control of bipedal locomotion.

III. Application to bipedal locomotion

In this section we show how, given a sample trajectory, we can use our generic CPG architecture to control bipedal locomotion on a simulation of the Hoap-2 (a 25-DOF humanoid robot built by Fujitsu). First, we present the controller architecture made of several coupled generic CPGs, one for each DOF. Then we show how we can easily integrate sensory feedback in the controller that generates the trajectories. The lower level control is done by a PID controller.

A. The controller architecture

In our controller architecture, we control 10 of the 25 DOFs of the robot. For the moment, the arms have fixed position. We control 2 of the 3 DOFs of each hip, the 3rd one which controls vertical rotation is not used. We also control the DOFs of the knees and the ankles. Figure 4 shows a schematic view of the Hoap-2 robot and its DOFs.

We use one generic CPG for each controlled DOF, each CPG is made of 3 oscillators as can be seen in Figure 5. For coordinating these several DOFs, for each leg we use a chain coupling from the hip to the ankle of the first oscillator of each CPG. And we add a symmetric coupling between the first oscillators of the Hip2 joints of each leg, to conserve a π phase difference between the legs.

The coupling scheme to keep correct phase differences between the DOFs of one leg is similar to the one we presented in Section II, for the oscillators of one generic CPG. The phase difference between 2 DOFs is also learned using the same evolution equation as in Equation 8. The equation of coupling between the oscillators and the learning rule for the phase difference are as follow

$$\dot{x}_{0,k}=(\mu - r^2)x_{0,k} - \omega_{0,k}y_{0,k} + \tau \sin(\theta_{0,k} - \phi_{0,k}) \quad (12)$$

$$\dot{\phi}_{0,k}=\sin(\theta_{0,k-1} - \theta_{0,k} - \phi_{0,k}) \quad (13)$$

where $(0, k)$ denotes the first oscillators of the kth CPG. The other terms are the same as defined in Equations (4)-(8). Thus, in addition to the 10 generic CPGs made of 3 oscillators, we add 8 new state variables to the system that will learn the correct phase difference between the CPGs of each DOF. Figure 5 shows the architecture of the controller.

We trained the generic CPGs with sample trajectories of walk motion of the Hoap-2 robot provided by Fujitsu. Each trajectory was a teacher signal to the corresponding CPG controlling the associated DOF. All the control parameters of the CPGs converged correctly and, after learning, the sample trajectories are encoded in the controller as can be shown in Figure 6. We clearly see that the learning was successful since the learned trajectories match well the sample trajectories. The system is able to generate the learned trajectories and moreover we benefit of properties of the CPGs, such as limit cycle behavior, amplitude and frequency modulation and the possibility to add feedback pathways. Now, online trajectory generation rather than following fixed trajectory is possible.

B. Feedback pathways

In this section we introduce three kinds of feedback pathways. We discuss each of them in the following.

1) Lateral stability: The first feedback pathway we introduce is for maintaining lateral stability during locomotion. This feedback pathway is inspired by the vestibular system in humans that measures the tilt of the body and activates contralateral muscle to keep balance. In this sense, we use the Gyros located in the chest of the robot to calculate the lateral tilt of the body. When this tilt is increasing we want

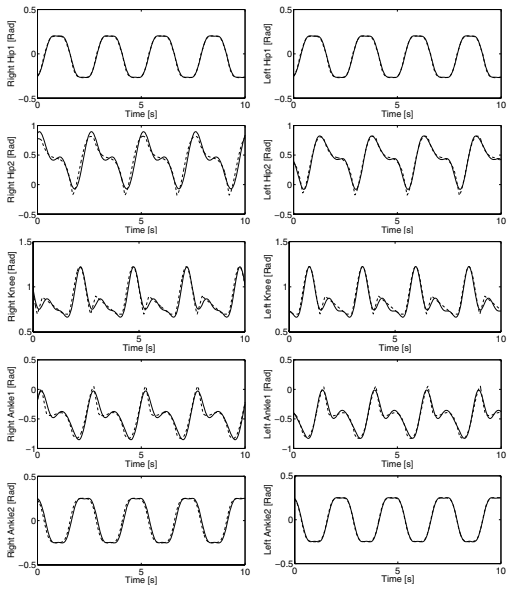


Fig. 6. Result of training of the generic CPG. We plotted the 10 controlled DOFs, the plain line corresponds to the output of the CPG for each DOF, the dashed line corresponds to the sample trajectory.

the robot to tilt in the opposite lateral direction. There are 2 DOFs controlling the lateral direction in the robot, one DOF in the hip called LEG_JOINT[2] (Hip1) and one in the ankle called LEG_JOINT[6] (Ankle2) (Figure 4). Consequently we will introduce the feedback pathways in the CPGs controlling these joints.

We notice from Figure 6 that the Hip1 joints have the same trajectories on both legs. We notice the same for the Ankle2 joints. The effect of feedback we want should be of opposite effect on the Ankles and on the Hip to keep the ankle parallel to the ground. The feedback pathways should also influence in the same way both legs.

Let ψ_{lateral} be the lateral tilt of the body, then we set the feedback for the ankles and the hips as

$$g_{\text{Ankle2}} = K_{\text{lateral}} |\psi_{\text{lateral}}| \quad (14)$$

$$g_{\text{Hip1}} = -K_{\text{lateral}} |\psi_{\text{lateral}}| \quad (15)$$

the gain K_{lateral} is the same for both feedback pathways because we want to assure that we have a symmetric change of trajectory, so when the ankle touches the ground, correct orientation is preserved.

We project these feedbacks on the radius of the limit cycle of all the oscillators associated to the Hip1 and Ankle2 joints. We make this projection to be sure that the phase is preserved, because we are only interested in amplitude of trajectories. The following Equation shows the principle

$$\dot{x}_i = (\mu - r^2)x_i - \omega_i y_i + \tau \sin(\theta_i - \phi_i) + g_k \frac{x_i}{r_i} \quad (16)$$

$$\dot{y}_i = (\mu - r^2)y_i + \omega_i x_i + g_k \frac{y_i}{r_i} \quad (17)$$

where x_i, y_i are the state variables of the ith oscillator, g_k is the feedback term (g_{Ankle2} or g_{Hip1}).

2) Pendulum effects compensation: When walking the body of the robot has the dynamics of an inverted pendulum. When modulating the speed of walking, we will change these effects and the controller has to compensate for these effects. Therefore we introduce feedback to compensate tilt of the body in the sagittal plane in the same way we did above. Let ψ_{Pendulum} be the angle of tilt of the body in the direction of walking, then we set the following feedback term

$$g_{\text{Pendulum}} = K_{\text{Pendulum}} \psi_{\text{Pendulum}} \quad (18)$$

we project this feedback term on the radius of all the oscillators of the CPGs associated to the Ankle1 joints (LEG_JOINT[5]) and the Knee joints (LEG_JOINT[4]).

3) Phase resetting: The effect of pendulum will also influence the frequency at which the legs touch the ground, which will be slightly different than the frequency of the trajectory generation in the controller.

To compensate this effect, we introduce phase resetting of the oscillators each time the right leg touches the ground. Importance of phase resetting for biped locomotion was already discussed in [11] where they showed that it creates entrainment of the controller with the body dynamics of the robot. This induce tight coupling between the body and the controller.

IV. Experimental results

In this section we present experiments we did with the CPG we presented. We did these experiment with a simulation of the Hoap-2 robot in Webots [12]. This simulator is based on ODE [13], 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. It 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, load sensors on the bottom of the feet).

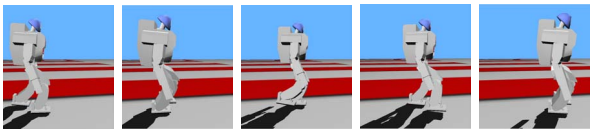
The architecture of CPG we presented generates online trajectories for each joint. We use these trajectories as desired angles for the PID controllers controlling each joint.

When increasing the stepping frequency and therefore the speed of locomotion, the CPG has to react faster to sensory feedback. By changing the gains of the feedback pathways, we can change this speed of response, thus we define gains for the feedback that depends on the speed of locomotion

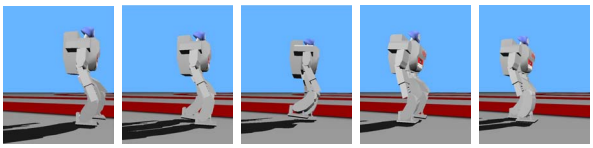
$$K_{\text{Lateral}} = 2000.0 + 200.0 * (\zeta - 1.0) \quad (19)$$

$$K_{\text{Pendulum}} = 1000.0(\zeta - 1.0) \quad (20)$$

where ζ is the ratio between the frequency to which we modulate the controller and the original frequency of the learned trajectory. In these equations we see that the gains increase as the speed of walking increases. We tested the CPG model with the simulated Hoap-2 robot. First of all we modulated the speed of walking by changing ζ. We managed to increase the speed of the robot up to 50% of the original speed by simply setting $\zeta = 1.5$. This correspond to a speed of approximately $0.12\ \text{m.s}^{-1}$. We also managed to generate backward locomotion by simply inverting the sign of $\vec{\omega}$. Pictures of the robot walking at $0.12\ \text{m.s}^{-1}$ and walking backward can be seen on Figure 7.



(a) The robot is walking at 0.12 $\mathrm{m.s}^{-1}$



(b) The robot is walking backward

Fig. 7. Snapshots of the robot while walking at higher speed (Figure (a)) and while walking backward (Figure (b)). The pictures have to be seen from left to right.

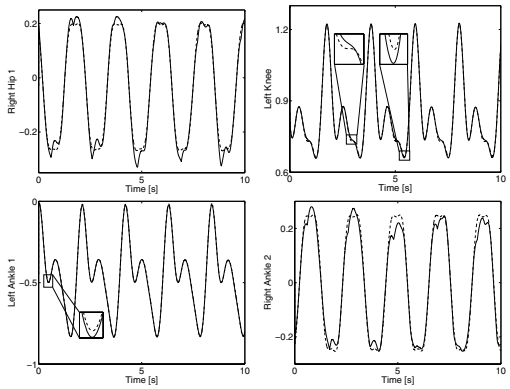


Fig. 8. Effect of feedback on the generation of trajectory. The dotted line shows the trajectory initially encoded in the CPG, the plain line shows the trajectory generated by the CPG with the feedback pathways. The graphs are taken from trajectories at speed $\zeta = 1.20$

Moreover, by linearly changing $\vec{\alpha}$ we managed to control the step length. It was possible to control the robot so that it made smaller steps and eventually stops if $\vec{\alpha} = 0$, it was possible to walk again by increasing $\vec{\alpha}$.

The feedback pathways enabled us to increase the speed of locomotion. Indeed if we do not activate the pathways, the robot falls when we increase speed of locomotion more than 5%. The contribution of the feedback, when increasing speed of locomotion can be seen in Figure 8. We plotted the trajectories generated without and with feedback when the robot walks 20% faster ($\zeta = 1.2$). It is obvious on these graphs that the lateral feedback modifies quite a lot the trajectories of the Hip1 and Ankle2 joints. The importance of feedback on the other joints is less obvious but the experiments showed that without this pathway, the robot falls.

V. Conclusion

In this contribution we presented a new architecture for building programmable Central Pattern Generators used for online trajectory generation in autonomous robots. The interest of the method we presented is that we can encode arbitrary periodic trajectories as limit cycles in a network of coupled oscillators. Then 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. Moreover we showed that it was easy to couple several of such networks to generate coordinated multidimensional periodic trajectories. Furthermore, this new architecture is general enough to be applied to various fields where the control of periodic signals is important as, for example, in signal processing.

Afterwards, we showed an application of this programmable Central Pattern Generator to control a simulated Hoap-2 humanoid robot for bipedal locomotion. We introduced simple feedback pathways and showed how from a sample trajectory we could build a controller able to modulate the speed of locomotion and the step length of the robot. These modulation are simple to do since we only have to vary the value of two parameters (ω and α).

We are currently implementing this architecture in a real humanoid Hoap-2 in order to completely validate the approach.

Acknowledgment

This work was made possible thanks to a Young Professorship Award to Auke Ijspeert from the Swiss National Science Foundation (A.I.) and to the support of the European Commission's Cognition Unit, project no. IST-2004-004370: RobotCub (L.R.).

References

[1] M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic, *Biped locomation. Dynamics, stability, control and application*, 1990.

[2] D. P. Pratt, J. and G. Pratt, "Virtual model control of bipedal walking robot," in *Proceedings of the IEEE Int. Conference on Robotics and Automation (ICRA1997)*, 1997, pp. 193–198.

[3] G. Taga, "Emergence of bipedal locomotion through entrainment among the neuro-musculo-skeletal system and the environment," *Physica D: Nonlinear Phenomena*, vol. 75, no. 1–3, pp. 190–208, 1994.

[4] G. Endo, J. Nakanishi, J. Morimoto, and G. Cheng, "Experimental studies of a neural oscillator for biped locomotion with qrio," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005, pp. 598–604.

[5] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2002)*, 2002, pp. 958–963.

[6] J. Buchli and A. Ijspeert, "A simple, adaptive locomotion toy-system," in *From Animals to Animats 8. Proceedings of the Eighth International Conference on the Simulation of Adaptive Behavior (SAB'04)*, S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J. Meyer, Eds. MIT Press, 2004, pp. 153–162.

[7] L. Righetti, J. Buchli, and A. Ijspeert, "Dynamic hebbian learning in adaptive frequency oscillators," *Physica D*, 2005, accepted.

[8] ——, "From dynamic hebbian learning for oscillators to adaptive central pattern generators," in *Proceedings of the Third International Symposium on Adaptive Motion in Animals and Machines AMAM2005*, 2005, in press.

[9] J. Acebrón, L. Bonilla, C. Pérez Vicente, F. Ritort, and R. Spigler, "The kuramoto model: A simple paradigm for synchronization phenomena," *Reviews of Modern Physics*, vol. 77, no. 1, pp. 137–185, 2005.

[10] Hoap-2 instruction manual. [Online]. Available: http://www.automation.fujitsu.com/en/products/products09.html

[11] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of locomotion with dynamical movement primitives," *Special issue of Robotics and Autonomous Systems*, no. 47, pp. 79–91, 2003.

[12] O. Michel, "Webots: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[13] Open dynamic engine documentation. [Online]. Available: http://www.ode.org