Auton Robot (2011) 31:155–181
DOI 10.1007/s10514-011-9235-2

Toward simple control for complex, autonomous robotic applications: combining discrete and rhythmic motor primitives

Sarah Degallier · Ludovic Righetti · Sebastien Gay · Auke Ijspeert

Received: 18 July 2010 / Accepted: 4 May 2011 / Published online: 24 May 2011

Abstract Vertebrates are able to quickly adapt to new environments in a very robust, seemingly effortless way. To explain both this adaptivity and robustness, a very promising perspective in neurosciences is the modular approach to movement generation: Movements results from combinations of a finite set of stable motor primitives organized at the spinal level. In this article we apply this concept of modular generation of movements to the control of robots with a high number of degrees of freedom, an issue that is challenging notably because planning complex, multidimensional trajectories in time-varying environments is a laborious and costly process. We thus propose to decrease the complexity of the planning phase through the use of a combination of discrete and rhythmic motor primitives, leading to the decoupling of the planning phase (i.e. the choice of behavior) and the actual trajectory generation. Such implementation eases the control of, and the switch between, different behaviors by reducing the dimensionality of the high-level commands. Moreover, since the motor primitives are generated by dynamical systems, the trajectories can be smoothly modulated, either by high-level commands to change the current behavior or by sensory feedback information to adapt to environmental constraints. In order to show the generality of our approach, we apply the framework to interactive drumming and infant crawling in a humanoid robot. These experiments illustrate the simplicity of the control architecture in terms of planning, the integration of different types of feedback (vision and contact) and the capacity of autonomously switching between different behaviors (crawling and simple reaching).

Keywords CPGs · Motor primitive · Adaptive behaviors · Dynamical systems · Humanoid robots · Bio-Inspiration · Drumming · Locomotion

1 Introduction

Controlling robots with multiple degrees of freedom (DOFs) for autonomous tasks is still an open and challenging issue, notably because planning complex, multidimensional trajectories in time-varying environments is a laborious and costly process. In previous work (Gay et al. 2010; Degallier et al. 2006, 2007, 2008; Righetti and Ijspeert 2006a, 2008), we have presented a control framework where the planning and the generation of movements are decoupled, i.e., the planning consists in defining the key characteristics of the desired movement (e.g., the target position for reaching), the actual trajectory generation relying on low-level attractor dynamics. This approach drastically reduces the dimensionality of the planning problem, making it particularly appropriate for robots with multiple DOFs, and we have applied

This work was supported by the European Commission's Cognition Unit, projects RobotCub and AMARSi. S.G. is funded by a IST-EPFL grant.

Electronic supplementary material The online version of this article (doi:10.1007/s10514-011-9235-2) contains supplementary material, which is available to authorized users.

S. Degallier (✉)
CNBI Laboratory, School of Engineering, EPFL Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
e-mail: sarah.degallier@epfl.ch

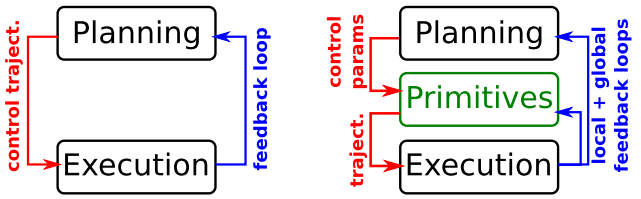
S. Gay · A. Ijspeert
Biorobotics Laboratory, School of Engineering, EPFL Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

L. Righetti
Computational Learning and Motor Control Lab, Computer Science, Neurosciences, & Biomedical Engineering, University of Southern California, Los Angeles, CA 90089, USA

it to the control of humanoid robots for diverse tasks such as drumming, crawling and reaching. In the present contribution, we review this work and fill it out with a more detailed presentation of the dynamical system on which the architecture is based, emphasizing the properties that make this system particularly well-suited for robotic applications.

In order to design our architecture, we took inspiration from the motor system of vertebrates, notably because animals are capable not only of performing highly complex tasks in a robust way but also of rapidly adapting to changes or uncertainties in the environment. Interestingly, the planning of movements and the actual generation of trajectories are most likely decoupled in vertebrates (e.g., Grillner 2006; Bizzi et al. 2008). The actual spatio-temporal sequence of activation of the muscles is produced at the spinal level through neural networks called *central pattern generators* (CPGs).[1] These CPGs are activated by simple, non-patterned control signals from the brain and that are modulated by sensory feedback. Thus, only the key parameters of the movement seem to be needed from the brain for a task to be completed.[2] In terms of control of robots, the idea behind the concept of CPGs is that movements are produced by so-called pattern generators, that have open parameters (the control signals) but whose dynamics are predefined, the output of such a generator being called a *motor primitive*. For instance, for reaching, the target of the movement is open, but features such as, e.g., the velocity and the acceleration profiles are encoded in the pattern generator and are thus fixed. As a result, the planning phase for a reaching movement consist only in specifying the final desired position, the whole trajectory being then computed by the pattern generator. Consequently, a CPG-based approach to movement generation reduces the dimensionality of the planning problem: instead of computing whole trajectories, only the key parameters of the movement need to be specified. In addition, the trajectories generated by the pattern generators (the motor primitives) can be modulated by sensory feedback in order to adapt the trajectories to partially unknown environments.

To model these motor primitives, we used dynamical systems with appropriately chosen attractor properties. We define a *pattern generator* as the system of equations that generates the trajectories and a *motor primitive* as the solution of this system. The dynamical system approach to understand movement coordination has been initiated by scientists such as Schoener, Kelso and Turvey (and others) (see, e.g. Turvey 1990; Schoener 1990; Schoener and Kelso 1988). Basically, the idea is to bring to light collective variables that



(a) **Traditional approach** (b) **Motor-primitive approach**

Fig. 1 Motor primitives for control. (**a**) In the traditional approach, a *high-level planner* computes the trajectories needed to achieve the task according to the feedback information. In redundant systems, the desired trajectory is usually found using optimization given a certain performance criterion. When the environment changes, inducing a modification of the feedback signal, the control trajectories need to be computed again. (**b**) A *low-level planner* based on CPGs. The generation of the trajectories is now divided into two steps: the definition of the control parameters of the motor primitives and the generation of the motor primitives by the CPGs. Optimization has to be done on a small set of trajectories. In addition to the main feedback loop, a local feedback loop can be added for rapid modulations of trajectories without requiring replanning. This local loop allows for fast on-line adaptation of trajectories

can describe coordination patterns, and the modeling of such patterns by dynamical systems depending on such collective variables (Schoener 1990). Schoener and Kelso (1988) have shown the importance of notions such as stability and phase transition in movement generation. The gaits observed during provides a typical example of such phenomenon: only a discrete number of stable states (the gaits) exists and a collective variable (the frequency) induces transitions between these states at critical points.

In terms of control, advantages of an approach based on motor primitives over traditional control approaches can be summarized as follows. In traditional approaches, there are usually two different processes: a *high-level planner* that computes the desired trajectories and a *low-level controller* (e.g., a PID controller) that transforms the desired trajectories into motor commands (see Fig. 1). The idea behind the concept of motor primitives is to add a *low-level planner* to the system that is composed of a set of trajectories with predefined dynamics. In terms of robotic control, the motor primitives can thus be seen as template trajectories in which a priori knowledge about the movements to be performed are embedded and that can be modulated according to feedback information. The advantage of using these primitives is threefold. First, they ease the planning problem by reducing the workspace of the robot to the control parameters of the motor primitives. Second, they provide the system with fast, local feedback loops for on line trajectory generation. It allows to rapidly correct desired trajectories according to time-varying perturbations or environmental changes without the need to replan the whole motor plan explicitly. Finally, different degrees of freedom can be coupled together to ensure inherent synchronization and coordinated behaviors. In other terms, motor primitives provide an

[1] Note here that we follow definition of CPGs of Grillner (2006) that includes both the generation of discrete and rhythmic movements.

[2] If the existence of CPGs in non-primate vertebrates is generally well accepted nowadays, the generalization to humans is still an open debate, indeed, influences from higher cortical areas and from sensory pathways are difficult to isolate (e.g., Capaday 2002).

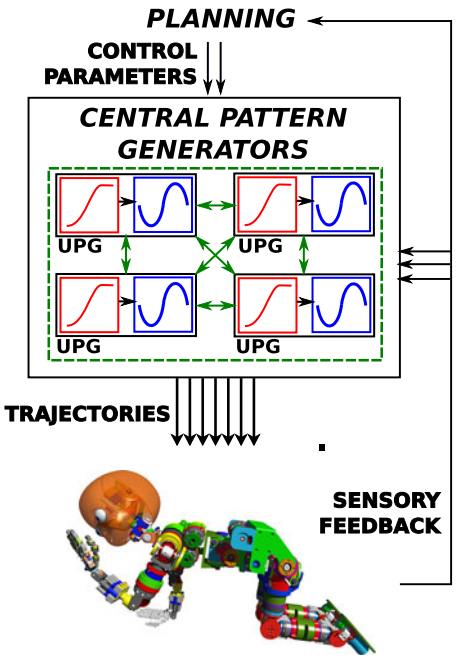


Fig. 2 Schema of the control architecture. A central pattern generator is seen here as a network of dynamical systems that allows for the generation of complex output trajectories given simple, non patterned inputs. The output of the system can be further modulated by sensory information. *In the dash line box*: The discrete and the rhythmic systems are combined together to form a unit pattern generator (UPG) that is responsible for the control of one degree of freedom (DOF). The UPGs of each DOF are then coupled together in a network, the central pattern generator (CPG, *in green*), in order to generate a coordinated behavior between the DOFs. Note that while the (open-loop) dynamics of the UPG is always the same, the CPG depends on the structure of the robot and on the task to be accomplished

effective, dynamic way to embed a priori knowledge about the task into the low-level planning system, as, for instance, arm synchronization for bi-manual tasks or trajectories with bell-shaped velocity profile for reaching movements. They thus provide a fundamental tool to develop efficient, fast architectures for the generation of movements, particularly in the case of robots with many degrees of freedom and meant to evolve in time-varying environments, such as humanoids.

Discrete and rhythmic movements are commonly considered separately in motor control theory and, mathematically speaking, different types of parameters are needed to characterize the movements. Several authors have studied the interaction of discrete and rhythmic movements, sometimes reaching different conclusions, as we reviewed in Degallier and Ijspeert (2010). In this article, we consider discrete and rhythmic movements as two basic types of movements that can be combined to generate hybrid trajectories. The general control schema that we are proposing here is depicted on Fig. 2: the CPGs generate trajectories according to the control parameters specified by the planning system, the whole architecture being influenced by feedback information coming from the robot. Note that this concept is not limited to control in the joint space and can easily be extended to operational space control for instance (see e.g. Khatib 1980). In our CPG, we model all movements through the combination of a discrete and a rhythmic motor primitives, both produced by a unique dynamical system, that we call a *unit pattern generator* (UPG). More precisely, movements are modeled as oscillatory movements around time-varying offset. Purely discrete movements can be obtained by setting the amplitude of the oscillations to zero and purely rhythmic ones by setting a constant offset.

The control of robotic devices using motor primitives modeled by dynamical systems has often been addressed in the literature, with applications to learning by demonstration—the so-called dynamical motor primitives (DMP)—(e.g., Ijspeert et al. 2003; Gribovskaya and Billard 2008; Pastor et al. 2009; Kober and Peters 2010), rehabilitation (Ronsse et al. 2010), locomotion (e.g., Kimura et al. 2007; Maufroy et al. 2008) and modular robotics (e.g., Cui et al. 2010; Sproewitz et al. 2010) for instance. Here our focus is the generation of trajectories given simple, explicit high-level commands, as for instance in Maufroy et al. (2008) for locomotion and Bullock and Grossberg (1988) and Hersch and Billard (2008) for reaching. The novelty here is that we address the generation of both discrete and rhythmic movements through the same system, a subject that as received little attention so far, as we reviewed in Degallier and Ijspeert (2010). Indeed, to the best of our knowledge, two main models for the simultaneous production of both discrete and rhythmic movements have been presented before, namely the models by De Rugy and Sternad (2003) and by Schaal et al. (2000), and they have never been applied to robotic control. The model presented by De Rugy and Sternad (2003), later extended to bi-manual tasks by Ronsse et al. (2009), is aimed at reproducing the key observations of the combination of discrete and rhythmic movements. It is based on a Matsuoka oscillator (Matsuoka 1985) modeling the output of two coupled neurons, this output being transformed into a desired trajectory through the equation of the dynamics of the joint. Schaal et al. (2000) proposed a rather complex system composed of two different motor primitives with many parameters, that allows for the reproduction of signals recorded in the brain, the drawback being that all these parameters need to be tuned precisely. Here our main focus is robotics (rather than the reproduction of observations made in humans) and our goal is thereby the design of a simple model with few, explicit control parameters corresponding to main characteristics of the movement (that is, the discrete target, the frequency and the amplitude). Note that Schoener (1990) introduced a system for the control of discrete movements, these movements being modeled as truncated rhythmic movements. More precisely, in this model, the quali-

tative solutions of a basic system are modulated by a supervising system according to the task specifications: postures are modeled as fixed points and movements by a Hopf oscillator. Discrete movements correspond to approximatively half of the limit cycle and the timing of the bifurcations (i.e., the qualitative change of the solutions to movements to posture) is controlled by the supervising layer according to the distance to the target. This system has been applied to the autonomous control of robots in navigation tasks several times (see, e.g., Steinhage and Bergener 1998; Tuma et al. 2009) and to reaching tasks in Schoener and Santos (2001). To the best of our knowledge, this system has never been applied to the generation of both discrete and rhythmic tasks for robotic applications, although the system could be easily extended to generate sequences of discrete and rhythmic movements. Here we take a different approach where discrete and rhythmic movements are considered as two separated types of movements that can be coupled to produce hybrid movements.

In this article, we first present in details the modeling of the CPGs (Sect. 2); we then illustrate the capacity of the system to easily switch between behaviors on the fly and the possibility of integrating different types of feedback through two applications, drumming (Sect. 4) and crawling (Sect. 5). The iCub robot, as well as the preexisting software that was used, are briefly presented in Sect. 3.

2 Presentation of the architecture

We present here the precise implementation of the CPGs. As illustrated on Fig. 2, all trajectories (for each joint) are generated through a unique set of differential equations (the UPG) and which is designed to produce complex movements modeled as periodic movements around time-varying offsets. Each UPG can be divided into two subsystems: the discrete and the rhythmic one. The first subsystem is responsible for the generation of short-term, goal directed features of the movement and the second subsystem for periodic features of the movements such as the amplitude and frequency of the pattern. The dynamics of the different DOFs can then be embedded in a larger network (the CPG) by coupling them together to ensure coordinated and synchronized behaviors. We present the discrete and the rhythmic systems separately in Sects. 2.1 and 2.2 respectively and we discuss their combination in Sect. 2.3. We then present how to couple the different systems to create a CPG (Sect. 2.4).

2.1 Discrete system

To generate discrete movements, we use a set of differential equations based on the VITE (*V*ector *I*ntegration *T*o *E*ndpoint) model originally developed by Bullock and Grossberg (1988) to simulate planned and passive arm movements. The target of the trajectory is encoded through a *difference vector* that represents the difference between the desired position of the DOF (γ_i) and its actual position (y_i). The speed of the movement is controlled by the so-called activity v that is proportional the difference vector ($y_i - \gamma_i$). Such an implementation allows for a coordinated control of several DOFs, as the time of convergence to the target is independent of the length of the trajectory, that is all DOFs will attain their target position simultaneously even if the distances to be covered by the joints are different (see Fig. 3(a)). We slightly modified the original system by Bullock and Grossberg (1988) to ensure that the initial speed of a movement is zero and that the velocity profile is bell-shaped. More precisely, for each degree of freedom i, a goal directed movement towards a target position γ_i can be generated through the following set of equations:

$$\dot{h}_i = 1 - h_i, \tag{1}$$

$$\dot{y}_i = v_i, \tag{2}$$

$$\dot{v}_i = -\frac{1}{4}B^2 h_i^2 (y_i - \gamma_i) - B h_i v_i \tag{3}$$

where y_i is the output of the system, v_i and h_i are auxiliary variables and B is a constant that controls the time of convergence of the system.[3] The system is critically damped so that the output y_i of (2) and (3) converges asymptotically and monotonically to the target γ_i with a speed of convergence controlled by B. Equation (1), that we call the *go command*, is used to ensure that the velocity profile is bell-shaped and, in particular, that the initial speed is null (as illustrated on Fig. 3(b)). h_i is reset to zero at the onset of each movement (a movement is considered to be new when the target is significantly larger than the previous one (0.1 rad in our case)). This system is relatively simple in the sense that the only parameter to select is the rate of convergence B, and the trajectory is fully determined by simply specifying one control parameter: the target γ_i of the movement.

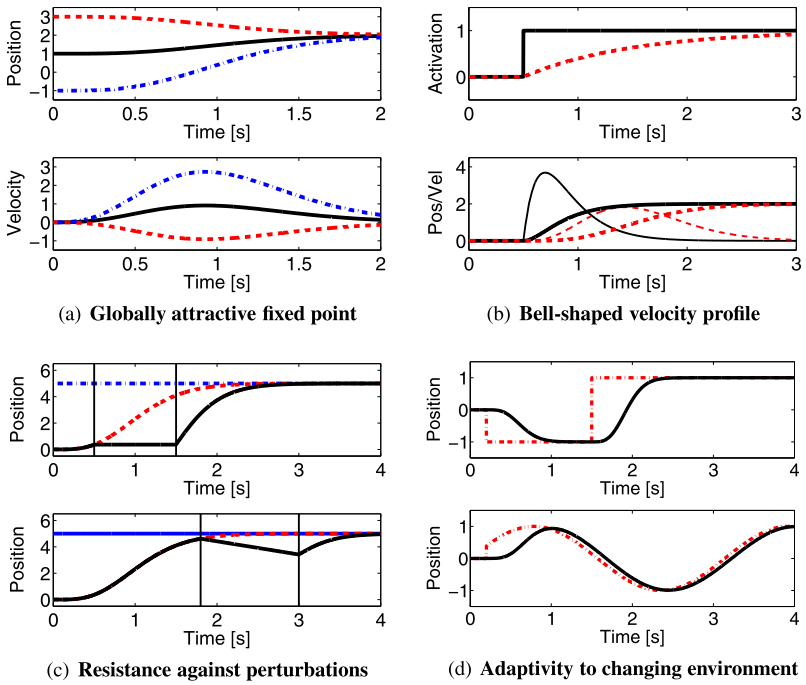
2.1.1 Stability and analytical solution

To ensure the stability of the system, we can analyze the eigenvalues of its Jacobian, that is

$$J_D = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ -0.5B^2 h_i (y_i - \gamma_i) & -0.25B^2 h_i^2 & -B h_i \end{pmatrix}$$

Thus, for any point of the state space (h_i, y_i, v_i), we have $\det(J_D - \lambda I) = (-1-\lambda)(\lambda + 0.5Bh_i)^2$, and hence $\lambda_0 = -1$

[3] Throughout this article, Greek letters will denote *control parameters*, lower-case Latin letters *variables* and capital Latin letters *constant values*.



(a) **Globally attractive fixed point** (b) **Bell-shaped velocity profile**

(c) **Resistance against perturbations** (d) **Adaptivity to changing environment**

Fig. 3 Discrete system. See text for discussion. The system was integrated using the Euler method with a time step of $t = 0.001$ s. Here the gain in (3) is set to $B = 10$. (**a**) *Top panel*: Different trajectories converging to same target position $\gamma_i = 2$ with different initial positions: $y = 1$ (*black*, *plain line*), $y = 3$ (*red*, *dash line*) and $y = -1$ (*blue*, *dash dotted line*). *Bottom panel*: Corresponding velocity profiles (same color/line type). (**b**) *Top panel*: Two types of activation command h_i: *in black*, *plain line* a step response ($h_i = 1$ at the time of activation $t = 0.5$ s, 0 before) and *in red*, *dotted line* a monotonically increasing activation (corresponding to the output of (1)). *Bottom panel*: Resulting velocity profiles with the constant activation (*black*, *plain thick line*) and with the increasing activation (*red*, *dotted thick line*) and the corresponding trajectories (same color/line type but thin lines) converging to the target $\gamma_i = 2$. (**c**) *Top panel*: The normal trajectory (*in red*, *dash line*) is modified (*in black*, *plain line*) due to a perturbation where the DOF is kept in a constant position ($y = \text{cst}$, in-between the vertical lines) from $t = 0.5$ to $t = 1.5$, but eventually converges to the target $\gamma_i = 5$ (*in blue*). *Bottom panel*: In this case, the perturbation is similar to a "force" exerted on the DOF ($\frac{dy}{dt} = -1$, in-between the vertical lines) from $t = 1.8$ to $t = 3.0$ (same color code/line types as in the top panel). (**d**) *Top panel*: The target position γ_i (*in red*, *dash dotted line*) is changed from $\gamma_i = -1$ to $\gamma_i = 1$ before convergence. In *black*, *plain line* is the resulting trajectory. *Bottom panel*: The time varying target position $\gamma_i(t)$ is here given by a sine signal (same color code as in the top panel). Note that here $B = 50$ to illustrate the fast convergence to the target

and $\lambda_1 = \lambda_2 = -0.5Bh_i$. For the go command, we have:

$$h(t) = 1 - e^{-t+t_0}$$

where t_0 is the time of initiation of the movement, as we set $h(t_0) = 0$ in our case. Hence $0 \leq h_i \leq 1$ and $B > 0$ and thus, since all the eigenvalues are negative, the general system is stable on the state space given by $[0, 1] \times \mathbb{R} \times \mathbb{R}$. The two eigenvalues of the system given by (2–3) are equal and real and hence the system is critically damped. Thus, if we consider $h_i = 1$, the solution is given by

$$y(t) = \gamma + C_y e^{-\frac{B}{2}t} + C_v t e^{-\frac{B}{2}t}$$

where C_y and C_v are constant that depends on the initial conditions $y(0)$ and $v(0)$.

2.1.2 *Properties of the discrete system*

We now present some features of the system—illustrated in Fig. 3—that will be useful for the application to robotics.

Globally attractive fixed point

The fixed point γ_i is globally attractive, which means that the trajectory will asymptotically converge to this point for any initial condition, as illustrated in Fig. 3(a). Moreover, as mentioned above, for any initial condition, all trajectories converge to the target γ_i at the same time, as the speed is proportional to the remaining distance to be covered, as can be observed on Fig. 3(a). Such a feature is interesting because all the DOFs move in a synchronized way, the drawback being that the speed of the movement is not directly controlled (unless B is changed).

Bell-shaped velocity profile

The auxiliary variable h_i modifies the velocity profile of the system to make it bell-shaped. More specifically, it is used to ensure that the velocity is null at the onset of the movement to avoid high peaks of accelerations. The effect of the chosen activation compared to a simple step response (as in the original VITE model) is illustrated on Fig. 3(b). Note that the auxiliary variable h_i must be reset to zero at each onset of a new movement.

Resistance against perturbations
Thanks to the global attractiveness of the fixed point, even if a perturbation occurs during or after the transient—as illustrated on Fig. 3(c)—the trajectory will eventually converge to the target position. Note that the duration of the perturbation does not influence the trajectory after since the system is autonomous. This feature is interesting because it can be used to modify the trajectory according to sensory information: for instance, if the DOF is stuck in a given position due to a obstacle for instance, the dynamics of the system can be temporarily modified so that the desired trajectory matches the actual environmental condition by using a perturbation similar to the one shown in Fig. 3(c) (top panel). Similarly, a repulsive force can be applied to avoid contact with a obstacle (Fig. 3(c), bottom panel).

Adaptivity to changing environment
Figure 3(d) illustrates the ability of the system to smoothly adapt to changes of the target position γ_i. In the top panel, it is shown that if the target position is suddenly changed (if for instance the object that has to be reached is suddenly moved, or if the target object changes), the trajectory is smoothly modulated to converge to the new target position. The bottom panel of Fig. 3(d) depicts the case where the target position γ_i is constantly changed. In this case the system is constantly updated so that it reproduces the trajectory of the moving target with a time delay that depends on the gain B. In order to deal with a constantly changing target position, the activity command h_i is reset only when the difference between the new target and the previous one is big enough (the threshold was set to 0.1 rad in our case). For instance, in the drumming application (see Sect. 4), a visual feedback loop constantly updates the target angles of the limb according to the actual position of the (possibly moving) drum pads.

2.1.3 Some additional remarks

Although we set B to a constant value here, it can also be used as a control parameter: as mentioned earlier, the value of B defines the duration of the discrete movement (independently of the distance to be covered), as illustrated on Fig. 3(a), upper panel. For instance, B can be tuned to reflect the so-called Fitts Law (Fitts 1954): the duration of simple reaching movement depends on the difficulty of the task, this difficulty being measured as the ratio between the distance to the target and the width of the target. B can thus be defined to be inversely proportional to the relationship defined by Fitts (smaller values of B leading to longer movements in time). In addition, since it has been shown by Kelso et al. (1979) that, in bimanual tasks, movements of different difficulties tend to have the same duration (that is, the duration of the more difficult movement), it can be postulated that the control command B is shared by the two arms. The benefit of using a common B is twofold: it reduces the number of control parameters needed and coordination between the two arms is inherent to the system.

It is important to note that, if a perturbation occurs, the trajectory will eventually converge to the desired target (thanks to the global attractiveness of the target). Now, evidence exist in motor control that animals tend to resume to the initial plan after perturbations (e.g., Bizzi et al. 1984; Won and Hogan 1995). A system where the trajectory is a sort of moving fixed point could be implemented and, in this case, if a perturbation occurs, the trajectory will converge to the position where it should be at a given time according to the initial plan. However, an attractive trajectory may cause dangerous behavior of the robot due to the (explicit or implicit) reference to time. Indeed, if a long term perturbation occurs, the system will converge back to the position initially planned with an uncontrolled speed. In our case, the trajectory after the perturbation is not affected by the duration of the perturbation, which motivates our choice since we focus on robotics application rather than on motor control modeling.

2.2 Rhythmic system

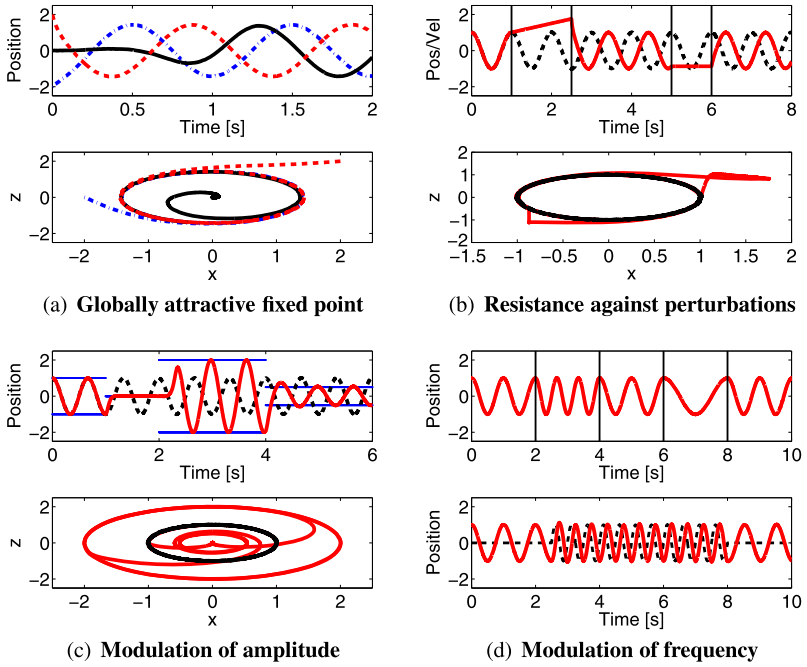
For the rhythmic system, we use a modified Hopf oscillator. Indeed, such an oscillator has many interesting properties, among which: (i) it has a unique periodic solution that is globally stable, (ii) this solution can be found analytically and is a perfect sine, and (iii) the frequency and the amplitude are explicit parameters. The system can be written as:

$$\dot{m}_i = C\,(\mu_i - m_i)\,, \tag{4}$$

$$\dot{x}_i = \frac{A}{|\mu_i|}(m_i - r_i^2)x_i - \omega_i z_i + n, \tag{5}$$

$$\dot{z}_i = \frac{A}{|\mu_i|}(m_i - r_i^2)z_i + \omega_i x_i + n \tag{6}$$

where x_i is the output of the system, z_i and m_i auxiliary variables, $r_i = \sqrt{x_i^2 + z_i^2}$, A and C are constant controlling the rate of convergence and n is a noise signal distributed normally ($n \sim \mathcal{N}(0, 1)$) added to avoid unstable solutions. The first term of the right-hand side of (5) and (6) ensures a constant amplitude while the second term induces the oscillatory behavior. When $\mu_i > 0$, (5) and (6) describe an Hopf oscillator whose solution x_i is a sine of amplitude $\sqrt{\mu_i}$ and frequency ω_i. A Hopf bifurcation occurs when $\mu_i < 0$ leading to a system with a globally attractive fixed point at (0, 0). Note that (4) was added to the canonical system to ensure that the output trajectory is smooth even when bifurcations occur.



(a) **Globally attractive fixed point** (b) **Resistance against perturbations**

(c) **Modulation of amplitude** (d) **Modulation of frequency**

Fig. 4 Rhythmic system. See text for discussion. The system was integrated using Euler method with a time step of $t = 0.001$ s. The gain in (5) and (6) is set to $A = 5$, and the gain in (4)is set to $C = 20$. The noise ε is distributed normally with mean 0 and standard deviation 1. (**a**) *Top panel*: Different trajectories converging to same limit cycle of amplitude $\sqrt{2}$. with different initial positions: $x = 0$, $z = 0$ (*black*), $x = 2$, $z = 2$ (*red*) and $x = -2$, $z = 0$ (*blue*). *Bottom panel*: The same trajectories in the phase plane-xz (same color). (**b**) *Top panel*: The normal trajectory (*black*) is modified (*red*) due to a perturbation where the DOF is kept in a constant position ($x = \text{cst}$) from $t = 0.5$ to $t = 1.5$ and from $t = 1.8$ to $t = 3$. *Bottom panel*: The same trajectories in the phase plane-xz (same color). (**c**) *Top panel*: The initial trajectory (*black*) is modulated through the parameter m (*in blue* $\pm\sqrt{m}$) resulting in changes in amplitude (*red*). At $t = 1$, m is set to a negative value (-5), leading to a Hopf bifurcation: the limit cycle becomes a fixed point system. At $t = 2$, m is set to 4 and the reverse bifurcation occurs. *Bottom panel*: The same trajectories in the phase plane-xz (same color). (**d**) *Top panel*: Modulation of the parameter ω_i: at $t = 0$ s, $\omega_i = 2\pi$, at $t = 2$ s, $\omega_i = 3\pi$, at $t = 4$ s, $\omega_i = 2\pi$, at $t = 6$ s, $\omega_i = \pi$ and at $t = 8$ s, $\omega_i = 2\pi$ (the *black vertical lines* denote times where ω_i is changed). *Bottom panel*: The original signal (*red*, *plain line*) is entrained by a signal $F(t) = \sin(4\pi)$ (*black*, *dotted line*) with a gain equals to 10, i.e. $\dot{x} = \cdots + 10F$, from $t = 2.5$ s to $t = 8$ s

2.2.1 *Stability and analytical solution*

To analyze the system, we rewrite the oscillator in polar coordinates (r, θ)[4] for x and z:

$$\dot{m} = C(\mu - m), \tag{7}$$

$$\dot{r} = \frac{A}{|\mu|}(m - r^2)r, \tag{8}$$

$$\dot{\theta} = \omega \tag{9}$$

with $r \in \mathbb{R}^+$ and $\theta \in \mathbb{R}$. In this way the radius and the phase dynamics are decoupled. The solutions of (7) and (9) are straightforward:

$$m(t) = \mu - (\mu - M_0)e^{-Ct}, \tag{10}$$

$$\theta(t) = \omega t + \Theta_0 \tag{11}$$

where $M_0 = m(0)$ and $\Theta_0 = \theta(0)$. It can easily be seen that μ is a stable fixed point. The phase θ is increasing at a constant rate. It is said to be neutrally stable, i.e., perturbations will not be forgotten, but will also not increase. Equation (8) bifurcates depending on the value of μ (as m will eventually converge to μ), indeed for $\mu \leq 0$ the system has a unique solution $r = 0$, while for $m > 0$, it has two solutions, $r = 0$ and $r = \mu$, as illustrated on Fig. 5.

If we consider that $m(t) = \mu > 0$, we can solve the system for the non-zero solution by using the fact that (8) is a Bernoulli equation. We obtain:

$$r(t) = \sqrt{\frac{\mu}{1 + \mu C_r e^{-\frac{2A}{|\mu|}(\mu t)}}} \tag{12}$$

where C_r is a constant depending on the initial conditions.

2.2.2 *Properties of the rhythmic system*

We now present some features of the system—illustrated in Fig. 4—that will be useful for the application to robotics.

[4]We do not follow here the convention stated before (see Footnote 3) according to which Greek letters denotes control parameters, since the Greek letter θ is commonly used to denote the variable corresponding to the phase.

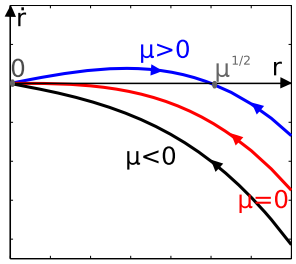


Fig. 5 Hopf bifurcation. Depending on the value of μ, the solutions of system change qualitatively. If $\mu > 0$ (in *blue* in the figure), the system has two solutions $r = 0$ and $r = \sqrt{\mu}$. In this case, and as indicated by the *arrows* that shows the direction of trajectories, $r = 0$ is a "repeller" and $r = \sqrt{\mu}$ an attractor. However, for $\mu = 0$ (*in red*) and $\mu < 0$ there is only one solution left ($r = 0$) and it is attractive

Attractive limit cycle

As illustrated on Fig. 4(a), all trajectories will eventually converge to the limit cycle for any initial conditions. Indeed, the system has two solutions, a stable limit cycle (a circle centered at the origin and of radius $\sqrt{\mu_i}$) and an unstable fixed point at $(0, 0)$. Thus, thanks to the noise (n) added in the equation, the system will eventually converge to the oscillatory solution even if initially at the unstable fixed point. However, as illustrated in Fig. 4(a), the convergence might be slower in that case (black, plain line) than for any other initial condition (blue, dashed and red, dotted-dashed lines). Note that the noise averages out over the duration of the movement, since it randomly affects velocity at each time step of integration.

Resistance against perturbations

Thanks to the attraction of the limit cycle, even if a short-time perturbation occurs, the system will resume to the limit cycle afterwards, as depicted on Fig. 4(b). Similarly to the discrete case, this feature can be used to modulate the dynamics of the system according to feedback information.

Modulation of amplitude and Hopf bifurcation

The amplitude of the oscillation is directly controlled by the parameter μ_i, more precisely, the amplitude is equal to $\sqrt{\mu_i}$ (when $\mu_i > 0$). Such feature allows us to very easily, and smoothly, modulate the system behavior according to the desired trajectory output, as illustrated in Fig. 4(c). As mentioned before, different types of solutions exist depending on the value of μ_i. In Fig. 4(c), Hopf bifurcations occur at $t = 1$ and $t = 2$. Thanks to the addition of (4), both transition are smooth. Note that without the addition of noise, the transition from the fixed point solution to the limit cycle can be very slow, as the fixed point remains a solution (even if unstable) after the bifurcation.

Modulation of frequency

Similarly to the amplitude, the frequency can be modulated directly through parameter ω_i, as shown in Fig. 4(d), top panel. Note that a periodic perturbation, if strong enough, can induce entrainment, i.e. the overall frequency of the oscillator will synchronize to the one of the external signal, as can be seen on Fig. 4(d), bottom panel. We will see in Sect. 2.4 that entrainment between oscillators can be used to couple them together.

In certain application, as for instance locomotion, it is desirable to have a independent control of the duration of the ascending phase (stance) and the descending phase (swing). Indeed, it is well known that in animal locomotion change of the overall speed are achieved by changing the duration of the stance phase, the duration of the swing phase being almost constant. In Righetti and Ijspeert (2006a), the term for frequency ω_i was modified to reflect this behavior, more precisely,

$$\omega_i = \frac{\omega_{\text{swing}}}{e^{-Dz_i} + 1} + \frac{\omega_{\text{stance}}}{e^{Dz_i} + 1} \tag{13}$$

where D is a constant parameter controlling the duration of the switch between the two phases. The frequency is now a function of two variables, ω_{swing} and ω_{stance}, that explicitly and independently control the swing and stance durations. Note that when $\omega_{\text{swing}} = \omega_{\text{stance}}$, we obtain the same output as before. Figure 6 illustrates the modulation of the original sine (in red) with a four times longer or four times shorter stance (in blue and in black respectively). The overall frequency of the system (and thus the speed of the robot) can be modulated by changing the duration of the stance only.

2.2.3 *Some additional remarks*

Note that in this system the phase is neutrally stable (perturbations neither decay nor grow), which means that perturbations may cause permanent phase shifts of the signal. In particular, the phase of a signal before and after a Hopf bifurcation can be different. The neutral stability of the phase avoids backwards movements on the limit cycle after perturbations, while ensuring that it will not diverge. In addition, as will be seen in Sect. 2.4, the phase difference between two signals can be controlled by coupling them if needed (as, for instance, to control the gaits in locomotion).

2.3 Unit pattern generator

In order to develop a low-level planner that can generate both discrete and rhythmic movements, we superimpose the dynamics of the two systems presented before in order to obtain a limit cycle that can be moved in the x-direction (as depicted of Fig. 7(b), bottom panel), i.e., the discrete movement is applied as a translation of the rhythmic one. This is obtained by embedding the discrete movement output y_i as

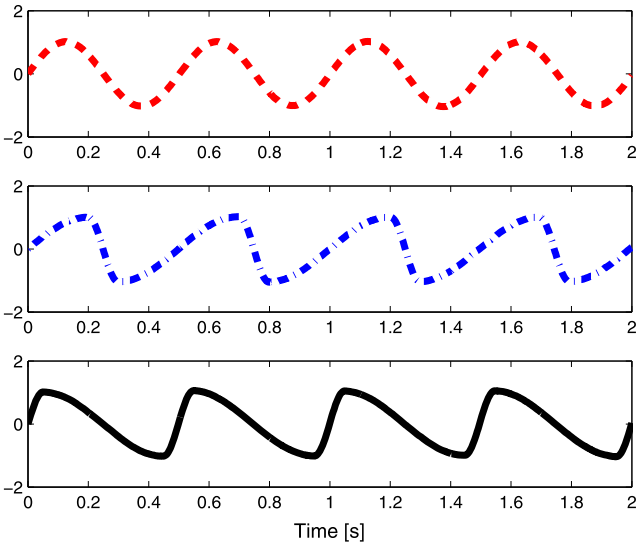


Fig. 6 Modulation of stance duration. In all the trajectories, the general frequency is fixed to $\omega = 4\pi$ and we modulate the duty factor of the walking cycle, defined as $d = T_{\text{stance}}/(T_{\text{stance}} + T_{\text{swing}})$ if $T_i = 1/\omega_i$ for $i =$ swing, stance. The swing period is then computed according to the stance period and the overall period (i.e. $T_{\text{swing}} = 2T - T_{\text{stance}}$). For the *red trajectory*, $\omega_{\text{swing}} = \omega_{\text{stance}}$ and the resulting trajectory is a symmetric sine (duty factor $d = 0.5$), for the *blue curve*, $\omega_{\text{stance}} = 2.5\pi$ and $d = 0.8$, i.e. the stance lasts four times longer than the swing, and for the *black curve* $\omega_{\text{stance}} = 10\pi$ and $d = 0.2$, i.e. the stance last four times shorter than the swing. Here $f = 100$, $b = 10$, $a = 5$, $\omega_i = 4\pi$ and the matlab function `randn` to generate the noise, the time step of integration being set to 0.001. The behavioral results of the change of the stance duration is illustrated by the Online Resource Movie 7

an offset of the rhythmic output x_i, that is

$$\dot{h}_i = 1 - h_i, \tag{14}$$

$$\dot{y}_i = v_i, \tag{15}$$

$$\dot{v}_i = -\frac{1}{4}B^2 h_i^2 (y_i - \gamma_i) - B h_i v_i, \tag{16}$$

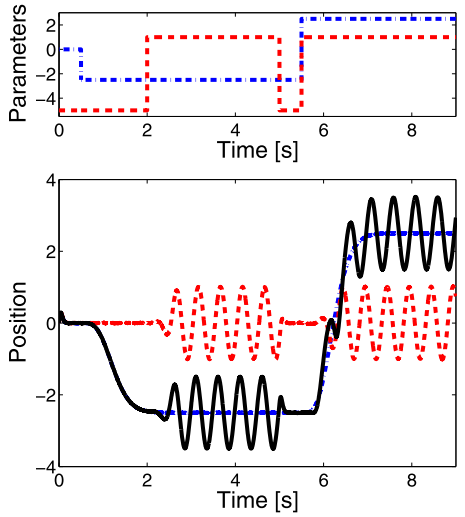
$$\dot{m}_i = C(\mu_i - m_i), \tag{17}$$

$$\dot{x}_i = \frac{A}{|\mu_i|}(m_i - r_i^2)(x_i - y_i) - \omega_i z_i + n, \tag{18}$$

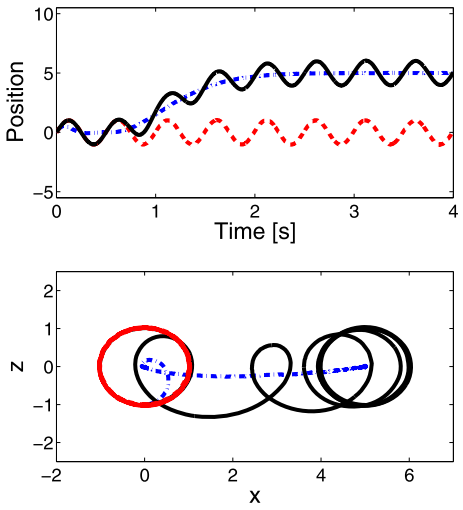
$$\dot{z}_i = \frac{A}{|\mu_i|}(m_i - r_i^2) z_i + \omega_i (x_i - y_i) + n \tag{19}$$

where x_i is the output of the system and now $r_i = \sqrt{(x_i - y_i)^2 + z_i^2}$. When $\mu_i > 0$, (5) and (6) describe a Hopf oscillator whose solution x_i is a periodic signal of amplitude $\sqrt{\mu_i}$ and frequency ω_i with an offset given by γ_i. A Hopf bifurcation occurs when $\mu_i < 0$ leading to a system with a globally attractive fixed point at (γ_i,0). The set of equations (15–19) is a *unit pattern generator*, that is the minimal set of equations controlling one degree of freedom, while (14) can be shared by several DOFs to ensure synchronized discrete movements.

Figure 7(a) (black line) depicts the qualitative behavior of the system depending on parameters μ_i and γ_i: the system can switch between purely discrete movements (from $t \approx 1$ s to $t \approx 2$ s), purely rhythmic movements (from $t \approx 2$ s to $t \approx 5$ s), and combinations of both (from $t \approx 6$ s to $t \approx 7$ s), the



(a) Switching between types of movements



(b) Three types of movements

Fig. 7 Unit pattern generator. See text for discussion. We used $B = 10$, $A = 5$, $C = 20$, $\omega_i = 4\pi$ and Euler integration (time step $t = 0.001$). (**a**) *Top panel:* The control parameters: *in red, dash line*, the amplitude, *in blue, dash-dotted line*, the target of the movement. *Bottom panel*: *In black*, *plain line* is the trajectory corresponding to the control commands of the top panel, in *red*, *dash line*, the movements resulting when no discrete movement is elicited ($\gamma_i = 0$) and *in blue, dash-dotted line*, when the rhythmic movement is switched off ($\mu_i = -5$). (**b**) *Top panel* A purely discrete movement *in blue*, *dash-dotted line*, a purely rhythmic one *in red*, *dash line*, and the combination of both *in black*, *plain line*. *Bottom panel*: The corresponding trajectories in the phase plan (same color/line code)

Table 1 Types of movements. This table summarizes the influence of the control parameters on the type of the movement. Here D = purely discrete, R = purely rhythmic, D+R = a combination of rhythmic and discrete movements

	γ_i	μ_i	ω_i
D	non constant	negative	any
R	constant	positive	non zero
D+R	non constant	positive	non zero

control parameters being extremely simple as it can be seen from the top panel. Discrete movements are simply elicited by specifying the target position γ_i (blue, dash-dotted line), while rhythmic movements are controlled through the specification of the parameter μ_i (red, dash line), which is the square of the amplitude of the output movements. Table 1 summarizes the control parameters and the induced types of behaviors.

The control of each degree of freedom is thus defined by a set of 6 equations (one of which—(14)—can be make common to all the DOFs to ensure a synchronized onset for the discrete movements), 3 internal constant parameters (A, B and C) and 3 control parameters (γ_i, μ_i and ω_i). This implementation is thus extremely economic, as the target γ_i (the amplitude μ_i and the frequency ω_i) are the minimal information needed to characterize a discrete (rhythmic) movement. Note that, as mentioned above, the parameter B can be used as a control parameter to modulate the speed of the discrete component of the movement, but is kept constant in all the applications presented here.

2.3.1 *Some examples of feedback loops*

We present here two possible ways of integrating feedback into the system that will be later used in the applications to drumming and crawling. Feedback loops can be designed as control policies in case of specific perturbations of the system. For instance, if a collision occurs, the system should react in a compliant way to absorb the shock, but also a strategy should be implemented to define a new trajectory that is consistent with the task to be performed. An implementation of such behavior is shown in the first example. The second example illustrates how a parameter can be controlled according to the feedback information. More precisely, we use the load information to control the phase of locomotion according to a simple rule: when a limb supports weight, it means that it should be in the stance phase, and in the swing phase otherwise.

Contact feedback

First, we present a feedback loop designed so that the robot stops its movement in the current position when the discrepancy between the desired and the actual position increases [see Ijspeert et al. (2002) for a first implementation]. The system of equations for the UPGs is simply modified in the following way: an attractor with a high gain ($E_x = 1000$ in our case) is added to the system to stop the movement in its current position $\hat{x}_i$ (in (20)) if the difference between the actual position $\hat{x}_i$ and the desired position x_i is large, i.e. we have

$$\dot{x} = \frac{A}{|\mu_i|}(m_i - r_i^2)(x_i - y_i) - \omega z_i + E_x(\hat{x}_i - x_i); \quad (20)$$

where $\hat{x}_i$ is the current position of joint i when the feedback is received and E_x is a constant controlling the gain of the feedback. Note that when $x_i = \hat{x}_i$ the feedback term vanishes and the system is equivalent to the open loop one. In application, as small differences may occur, e.g., due to time delay, a error threshold ϵ can be defined to avoid the activation of the signal in undesired cases (e.g., by using an expression such as $\max(0, |\hat{x}_i - x_i| - \epsilon)$ instead of the raw error).

Such a feedback term can be useful in different situations, as illustrated on Fig. 8. The top panel simulates a situation where an obstacle (denoted by the blue line) prevents the DOF to follow the initially planned trajectory (red, dotted line). In this case, the trajectory is modulated by the feedback to adapt to this constraint (black, plain line). In other words, the trajectory defined by the high-level planner (red, dotted line) is modified at the CPGs level (black, plain line) to adapt to the environmental con-

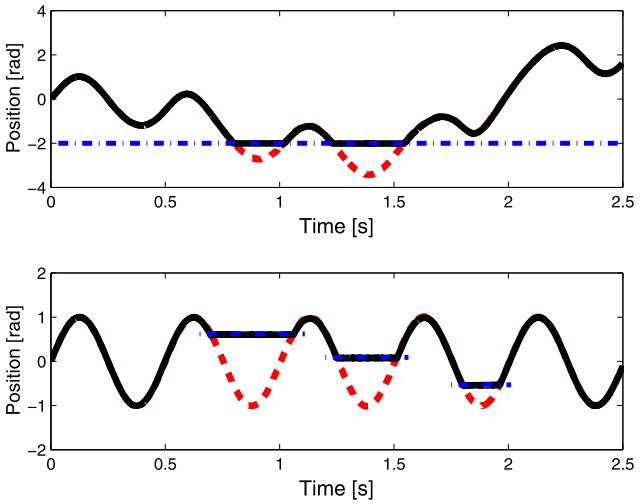


Fig. 8 Contact feedback. In *blue*, *dash-dotted line* is the obstacle (or joint limit), in *red*, *dotted line* the trajectory defined by control parameters (high-level feedback planner) and in *black*, *plain line* CPGs trajectory modulated by the feedback. *Upper panel:* A constant obstacle located at −2 rad prevents the DOF to follow the desired trajectory, creating a difference between the actual and the desired position. The fixed

straints (blue, dash-dotted line). Note that reaching a joint limit will induce a similar behavior. The bottom panel illustrates a similar situation with a moving obstacle, or several obstacles located in different positions. This latter situation is the one that we will encounter in the drumming application.

Phase-dependent feedback

The second feedback loop that we present here has been developed by Righetti and Ijspeert (2008): it consists in a phase-dependent sensory feedback that is used to increase locomotion stability on uneven terrains. It is inspired from mammalian locomotion where local sensory information such as load sensing on the extremities of a limb has an important role in the modulation of the onset of swing and stance phases (see Frigon and Rossignol 2006 for a review). The dynamics of the oscillator and thus the policy generation is modified on line according to load sensing on the end effectors (hands and knees of the robot), more precisely, a feedback term is added to (6) to modulate the transitions as follows

$$\dot{z}_i = \cdots + \begin{cases} -\text{sign}(z_i)F & \textit{fast transitions} \\ -\omega(x_i - y_i)(+\text{couplings}) & \textit{stop transition} \\ 0 & \textit{normal} \end{cases} \tag{21}$$

where F ($= 10$ in our case) controls the speed of the transition. Figure 9 shows the activation of the feedback depending on the phase of the limb and the resulting modification of the phase space of the oscillator. As long as a limb supports the body weight the transition from stance to swing phase for this limb is delayed. A faster transition occurs in the case of early limb unloading. In the case of swing to stance transition, a analogous behavior is implemented: transition is delayed as long as the limb does not touch the ground and is triggered in case of an early contact with the ground.

These feedback loops use only local information to change the control policies locally and as such is a first layer of adaptation in unpredicted environments. In other words, it provides a way to act locally on the trajectories for fast adaptation under environmental constraints without requiring a modulation of the motor plan. Such feedback pathways serve as an example to show the flexibility and versatility of the proposed architecture for on line trajectory generation.

2.3.2 *Some additional remarks*

Concerning the interaction between the two subsystems, the key parameters of the trajectory—the target, the frequency and the amplitude—are not affected in a permanent way (although very small transient perturbations may occur) but the phase of the rhythmic movement can be perturbed (as it is neutrally stable). Systematic tests show that for most choices of parameters B and ω, the phase of the rhythmic signal will change after a discrete movement. This difference depends on the phase of the signal at the onset of the discrete movement as shown on Fig. 10. Similarly, the time of onset of a rhythmic movement during a discrete one will have an influence on the phase (in the same way as different initial conditions influence the phase), as illustrated on Fig. 11. As mentioned earlier, this phenomenon can be overcome by coupling the system to a reference signal whenever it is needed in practical applications, as will be discussed in the next section.

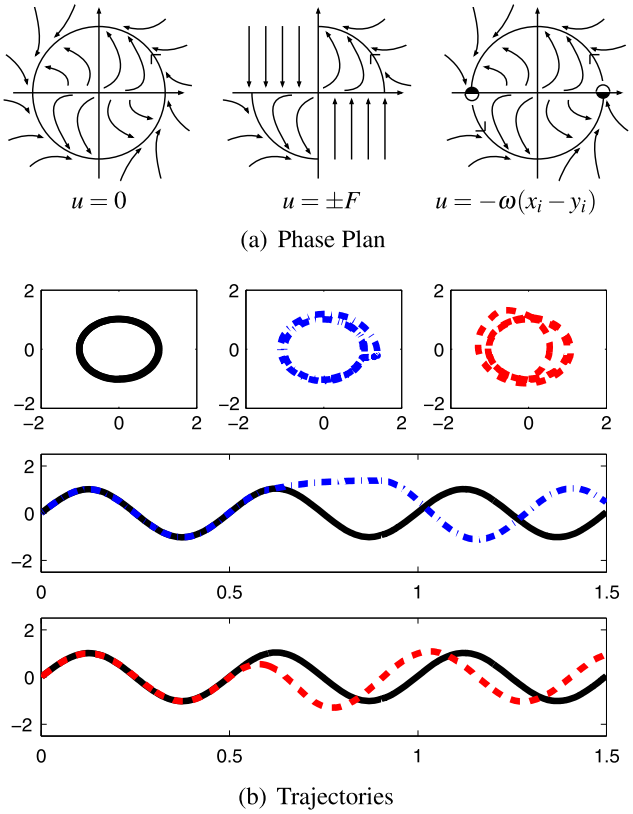


Fig. 9 Feedback strategy for the independent control of the swing and stance duration. (**a**) Modulation of the phase plan by the feedback *Left panel*: normal phase plan of the Hopf oscillator. *Middle panel*: Strategy for accelerating the transitions. The speed is increased by F ($= 10$ in our case). *Right panel*: Strategy for slowing down transitions. The system is stopped by canceling the oscillatory terms. (**b**) Corresponding trajectories. *Top panel*: Trajectories in the phase plan corresponding the strategies in (**a**): *normal* (*black*), *fast transition* (*red*), *stop transition* (*blue*). *Middle panel*: *Stop transition*. Decelerated trajectory (*blue*, *dash dotted line*) compared to unperturbed trajectory (*black*, *plain line*) in time domain. *Bottom panel*: *Fast transition*. Accelerated trajectory (*red*, *dash line*) compared to unperturbed trajectory (*black*, *plain line*) in time domain

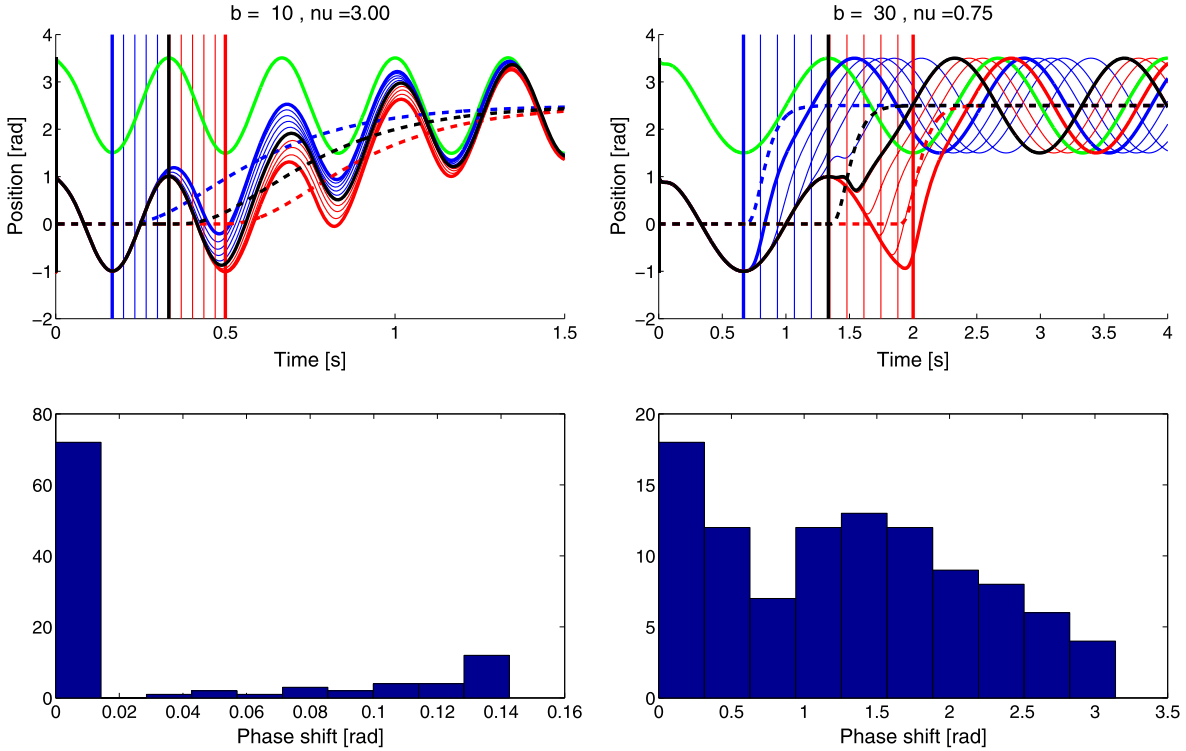


Fig. 10 Effects on the onset of a discrete movement on the phase of an ongoing rhythmic movement. This figure shows the impact of the discrete movement on the rhythmic one depending on the time of onset of the former relatively to the phase of the latter. The effect is shown for two extreme value of the set of parameters (B, ω). It can be seen from the figure that the discrete movement affects the rhythmic one in any cases, even if for the case (10, 3 Hz) the influence is restricted to a small interval of values. *Top panels*: *In green* is the unperturbed rhythmic signal that is used as a reference to compute the phase shift. We show 11 trajectories corresponding to 11 different discrete onsets equally spaced over one period of the rhythmic movement. Trajectories *in blue* indicates when the two dynamics are cooperating (i.e., same directions of velocity), the reverse being indicated by red lines (i.e. opposite directions of velocity). The *black line* indicate the border case (null velocity for the rhythmic movement). *Vertical lines* show the time of onset of the movements. *Bottom panels:* Histograms showing the distribution of the phase shifts for 100 trajectories with their onset being equally distributed during the phase of the movement. A phase shift of zero meaning no perturbation. Again, it can be seen that, while in the left case, there is no influence for more than 70% of the trajectories, in the right case there is an influence for more than 80% of them

2.4 Central pattern generator

In order to obtain a coordinated behavior between several DOFs, their UPGs can be coupled in a network to obtain coordinated behaviors. Such networks, that we call central pattern generators (CPGs), ensure fixed time relationships between the different rhythmic outputs (i.e. phase-locking), a feature which is particularly convenient for generating different gaits for locomotion for instance, as illustrated on Fig. 12.

The coupling of a DOF i with other DOFs (j's) is done by extending (18) and (19) in the following way

$$\dot{x}_i = \cdots + \Sigma_{j \neq i} K_{ij}^x (\cos(\theta_{ij})(x_j - y_j) - \sin(\theta_{ij}) z_j), \quad (22)$$

$$\dot{z}_i = \cdots + \Sigma_{j \neq i} K_{ij}^z (\sin(\theta_{ij})(x_j - y_j) + \cos(\theta_{ij}) z_j) \quad (23)$$

where the θ_{ij}s control the phase difference between DOF i and j and the $K_{ij}^{x/z}$'s are the constant gains of the coupling, i.e. the rate of convergence to a stable solution.

CPGs design

Thanks to the couplings, a network with fixed relationships between the different elements can be designed. Figure 12 depicts CPGs corresponding to the following gaits: trot, pace, (asymmetric) bound and walk, and the corresponding trajectories. As shown by Table 2, the matrix of θ_{ij}'s is skew-symmetric with a null diagonal. Note that, to ensure convergence to the desired phase shifts, the CPG network should be designed in a coherent way, in the sense that the sum of every phase differences along a closed path must be a multiple of 2π (for the obvious reason that a cell should be in phase with itself).

Smooth on line modulation

In addition, the phase relationships between the different elements of the CPG can be modified on line as illustrated on Fig. 13. Note that the time required to converge to the new solution depends on the parameters K_{ij}^x and K_{ij}^z. Similarly, if a short-term perturbation occurs, the system will resume to the desired phase-shift relationship afterwards.

The system that we have developed is general enough to be applied to various tasks, as will be presented in Sects. 4 and 5. First, we briefly present the hardware and software setup that we used as well as the physics simulator.

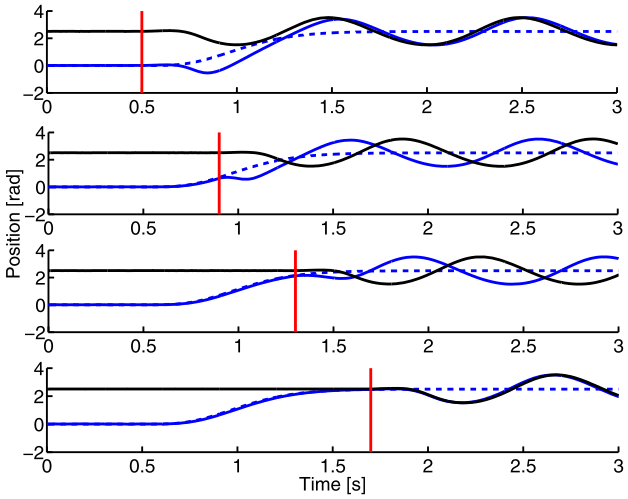


Fig. 11 Effects on the onset of a rhythmic movement during a discrete one on the phase. Depending on the time of onset of the rhythmic movement during a discrete movement (denoted by the *vertical red lines*), the phase shift between the purely rhythmic movement (in *black*) and the hybrid one (in *blue*) are different. This can be explained by the fact that the initial conditions at the time of onset of the rhythmic movement (that define the phase) will be different

3 Hardware and software

Before presenting the actual application of the architecture, we briefly present here the RobotCub project and the iCub robot, as well as the main software tools that were used in our implementation of drumming and crawling.

3.1 RobotCub

RobotCub is a 5 year-long EU-funded project that ended in January 2010. Its goals were twofold: first, to develop a humanoid robot—the iCub—of the size of a 3.5 years old infant, and second, to use this platform to study cognition and its development (see Tsagarakis et al. 2007 for instance). All the software developed during this project for the iCub robot, and notably the code for crawling and drumming that will be presented below, is open source. The software is based on the open source library YARP developed by Fitzpatrick et al. (2008) to support software development and integration in robotics.

3.1.1 Hardware

The robot has 53 degrees of freedom (DOFs): 6 for each leg, 16 for each arm (among which 9 for each hand), 3 for the torso and 6 for the head. Most of the DOFs axes and their names are depicted on Fig. 14. From now on, we will refer to the joints by their names as they appear on this figure.

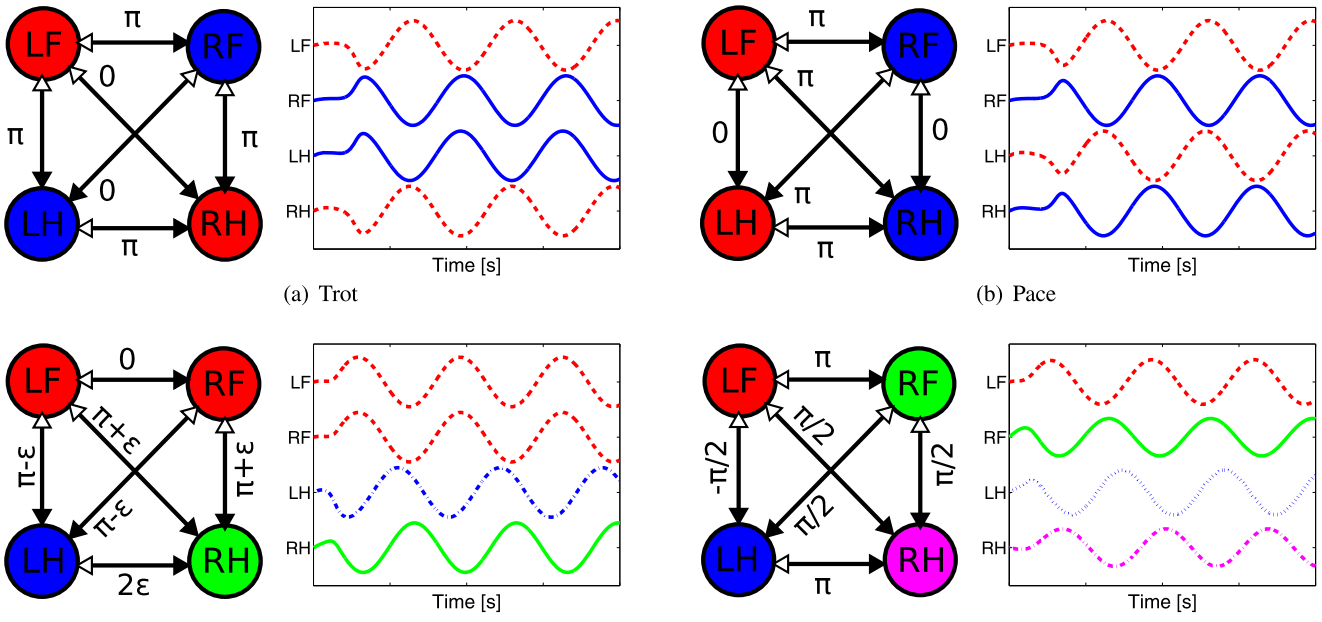
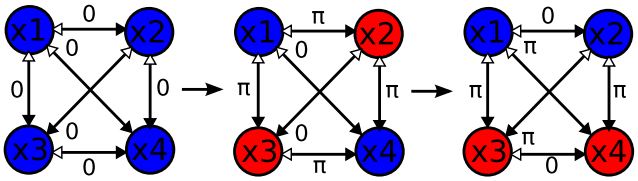


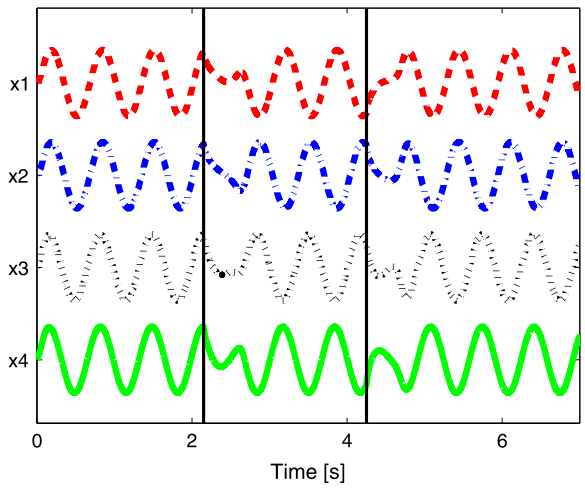
Fig. 12 CPGs applied to Gait Generation. See text for discussion. LF = left forelimb, RF = right forelimb, LH = left hind-limb and RH = right hind-limb. Here $K_{ij}^{x/z} = 8$, $\forall i, j$ and we used $B = 10$, $A = 5$, $\omega_i = 4\pi$. The system was integrated using Euler method with a time step of $t = 0.001$ s. (**a–d**) Schemes of the phase shifts for the different gaits (*left*) and the corresponding trajectories (with same color) (*right*). Cells/trajectories of the same color/line type means that they are in phase. Note that for each arrow, the angle θ attached is the angle corresponding to the *full*, *black arrow*, whereas the angle corresponding to the white arrow should be taken as the opposite $(-\theta)$ to ensure coherence. For a more explicit specification of the angles, please refer to Table 2

Table 2 Angles needed for different gaits. These tables summarize the angles required to generated the different gaits presented on Fig. 12. See text for discussion. LF = left forelimb, RF = right forelimb, LH = left hind-limb and RH = right hind-limb. ε is an open parameter that controls the phase shift between the two hind legs in the bound gait

Trot	LF	RF	LH	RH	Pace	LF	RF	LH	RH	Bound	LF	RF	LH	RH	Walk	LF	RF	LH	RH
LF	0	π	π	0	LF	0	π	0	π	LF	0	0	$\pi-\varepsilon$	$\pi+\varepsilon$	LF	0	π	$-\pi/2$	$\pi/2$
RF	$-\pi$	0	0	π	RF	$-\pi$	0	$-\pi$	0	RF	0	0	$\pi-\varepsilon$	$\pi+\varepsilon$	RF	$-\pi$	0	$\pi/2$	$\pi/2$
LH	$-\pi$	0	0	0	LH	0	π	0	π	LH	$-\pi+\varepsilon$	$-\pi+\varepsilon$	0	-2ε	LH	$\pi/2$	$-\pi/2$	0	π
RH	0	$-\pi$	0	0	RH	$-\pi$	0	$-\pi$	0	RH	$-\pi-\varepsilon$	$-\pi-\varepsilon$	2ε	0	RH	$-\pi/2$	$-\pi/2$	$-\pi$	0



(a) Scheme of the different CPGs



(b) Transitions between the different CPGs

Fig. 13 Transition between different couplings. See text for discussion. Here $K_{ij}^{x/y} = 5, \forall i, j$ and we used $B = 10$, $A = 5$, $\omega_i = 4\pi$. The system was integrated using Euler method with a time step of $t = 0.001$. (**a**) Scheme of the different CPGs configurations and (**b**) the corresponding trajectories. The modification happen at $t = 2.15$ s and $t = 4.25$ s and are denoted by *vertical black lines*

3.1.2 iCub Software

The iCub software architecture is based on YARP, an interprocess communication layer, which enables complete abstraction of the communication protocol between different software modules. Each module streams its output data through YARP ports, and these building blocks can be interconnected regardless of their physical location on the network (same computer, Ethernet network, etc.). iCub capabilities are thus implemented as a set of modules that can be easily connected together through YARP ports.

The software for the iCub comes with a set of kinematics libraries called iKin, developed by U. Pattacini. It allows forward and inverse kinematics computations on any subchain of the iCub degrees of freedom. The forward kinematics library uses standard Denavit-Hartenberg convention to enable the projection of a position to the reference frame of any part of the robot. It can be used for instance to project the position of an object in the camera reference frame to the root reference frame of the robot, or to check for internal collisions. The inverse kinematics library uses the IPOPT library (Wächter and Biegler 2006) to solve the non-linear inverse kinematics problem with N DOFs under the set of constraints defined by the limits of each joint. A maximum error as well as a maximum number of iterations of the optimization algorithm can be set for a compromise between precision and computational complexity.

3.2 Webots

Webots™ (Michel 2004) is a simulator based on the Open Dynamics Engine (ODE) library for simulating rigid body dynamics. A model of the iCub was developed according to the Denavit-Hartenberg parameters of the real iCub as well as the joints limits and maximum torque of the motors. A YARP interface similar to the iCub robot interface was also developed, so that the same YARP modules can be used on the simulator and on the real robot without any modification. This interface is freely available on the RobotCub website.

3.3 ARToolKitPlus

ARToolKitPlus (Wagner and Schmalstieg 2007) is a marker-based 3D vision tracker. It allows for detection and tracking of specific black and white markers and computes the full transformation matrix of the marker in the camera frame. This tracker uses only one camera and is widely used for its robustness to changes of lightning. We use this tracker together with the iKin library to compute the 3D position of markers and project it to the root reference frame of the robot. The capability of the tracker to compute the orientation of the markers can be used to set an offset in 3D between the position of the marker and the actual position of

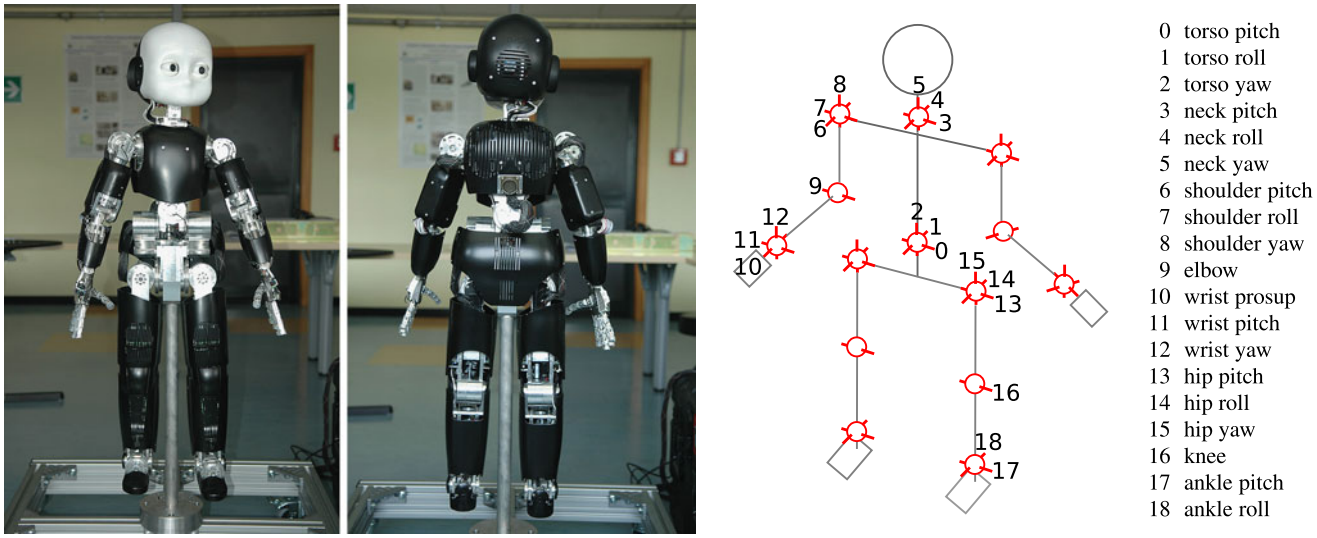


Fig. 14 Structure of the iCub. Schematic of the dofs of the iCub (excluding the dofs of the hands and eyes)

the object. In the drumming application for instance, the actual position of the center of each drum is then computed according to a predefined 3D offset between the marker and the drum. The position of each drum is then projected to the root reference frame of the robot, the default frame of reference for all kinematics computations. For each drum, the corresponding target positions for each DOFs (γ_i) of the closest arm of the robot are computed using and extended version of the default inverse kinematics solver provided by iKin. We modified the solver to include the drumming stick as an additional link in the iCub arm chain so that the tip of the stick reaches the center of the drum. Note that since we are using an optimization algorithm we are not sure to converge to a solution and thus the maximal error was set to 5 cm to ensure that suitable angles could be find rapidly enough.

4 Drumming

Drumming is a challenging application as it requires coordination between the limbs, precise timing and the robust on line modulation of the parameters—without raising the question of balance, as the robot is fixed to metallic structure in our case. Drumming has been implemented on robots several times before, to study agent-object interaction (Williamson 1999), learning from demonstration (Ijspeert et al. 2002) or human-robot interaction (Kose-Bagci et al. 2010) for instance. Here we focus mainly on the adaptability and robustness of the implementation: trajectories are modulated on line both by high level commands and by feedback information.

In this application, a user can define in real time the score that the robot is playing through a graphical user interface (GUI). The robot is playing on an electronic drum set, with two drum pads for each arms and two pedals for the legs. More precisely, the user can decide if the limb is idle or not, on which drum the limb is hitting (for the arms), the general frequency and the phase differences between the four limbs. The architecture is robust enough so that any user can play with the interface in a way that is secure for the robot. This demonstration was shown in CogSys 2008 in Karlsruhe and Automatica 2008 in Munich and ran during several hours. Note that the implementation is not platform dependent and that a similar application, although with predefined scores, has been implemented before on the Hoap2 robot (see Degallier et al. 2006 for details).

The whole system is modulated by two feedback signals, that are contact detection between the limbs and the drums (on the real robot) and visual tracking of the drums (in simulation for now). More precisely, contact detection is used to stop the robot in its current position when encountering an obstacle to safely handle the collision, and the drums are visually tracked so that the trajectories of the limbs can be adapted autonomously to their actual position. A synthetic version of this section, excluding the visual feedback, was presented at the BIOROB conference in 2008 (Degallier et al. 2008).

4.1 Software implementation

The implementation, depicted on Fig. 15, consists of four main blocks:

(i) **Task specification**: a graphical interface (GUI) that allows a user to define the behavior (i.e. the score) of the robot on line.
(ii) **Whole-body CPG**: the network is composed of a CPG for each of the four limbs and the head; each of them are

coupled to a clock that is used as an absolute referential of time (similarly to a metronome in music).

(iii) **Constraints**: the control commands need to be adapted to the actual environment and state of the system; this is done through two subsystems:
   (a) *Timing*: The phase of the different CPGs is monitored to know when a new note of the score should be played and thus to gate the update of the control commands of the CPGs.
   (b) *IK*: Thanks to the visual feedback, the Cartesian position of the drums, indicated by markers, are transformed into target joint angles through the inverse kinematics algorithm iKin (see Sect. 3).

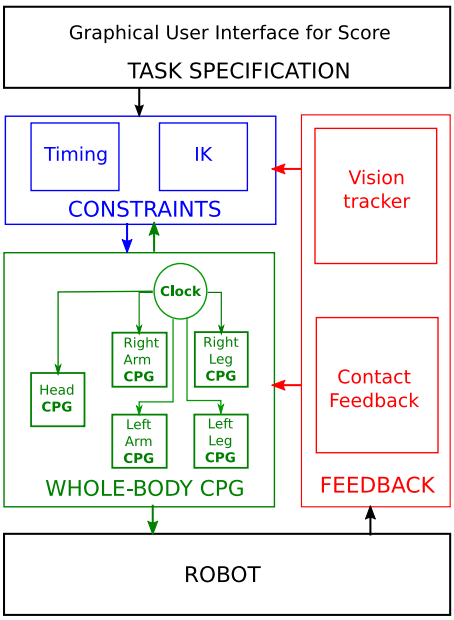


Fig. 15 Implementation of the drumming behavior. This implementation is designed so that any user can interact with the robot to make it play a score of his/her choice through a simple graphical user interface. Visual feedback is added to the system so that the robot can detect the drums through ARToolKItPlus markers and autonomously adapt its movement to their position. Finally, a feedback to deal with collisions between the arms of the robot and the drums is added for security reasons. Five parts are controlled, namely the head, the left arm, the right arm, the left leg and the right leg. *Green arrows* denote couplings

(iv) **Feedback**: the information from the robot is used to modulate the trajectories; it consists in two subsystems
   (a) *Contact feedback*: The movement is temporarily stopped when a collision with an obstacle is detected.
   (b) *Visual tracker*: Vision is used to track and update the Cartesian position of the drums.

These blocks will be presented more in details in the following. The set up for drumming can be seen on Fig. 16: the robot is fixed to a metallic structure by the hips and plays on an electronic drum set. The four limbs together with the head are controlled. We control actively four joints for each limb (the shoulder pitch, roll and yaw and the elbow for the arms, and the hips pitch, roll and yaw and the knee for the legs) and the six DOFs of the head (neck and eyes). The sticks are grasped by the hands which remain closed afterwards. The pedals are placed so that the robot can easily reach them when its legs are stretched.

4.2 Design of the whole-body CPG

All the DOFs are controlled by the unit pattern generator defined in Sect. 2.3. Only two joints per limb are oscillating: the shoulder pitch and the elbow for the arms and the hip pitch and the knee for the legs, the other DOFs output being always purely discrete. The neck yaw is also oscillating. The network is illustrated on Fig. 15: all the limbs CPGs are unilaterally coupled to the clock, as well as the head. Note that the coupling is unilateral so that the limbs do not affect the clock. The coupling parameters were set to $K_{ij}^x = K_{ij}^y = 2$. This value was chosen as it allows for a rapid convergence to the desired state (less than half a cycle) with a limited acceleration.

4.3 Task specification and constraints

A graphical interface (based on the open source library Qt3) was developed to ease the control of the robot. The open parameters are the following:

- for each arm: ID of the target drum or idle, phase shift relatively to the clock

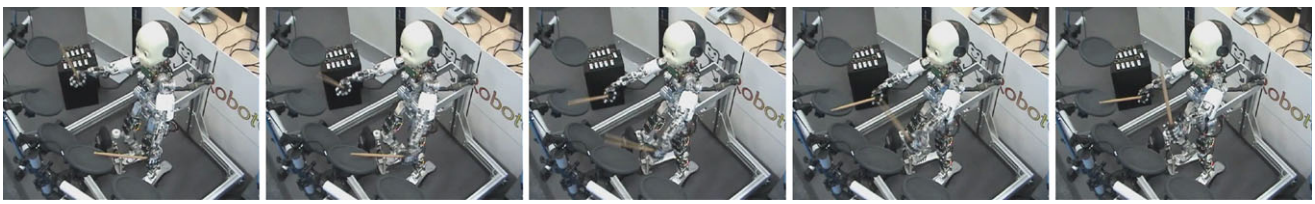
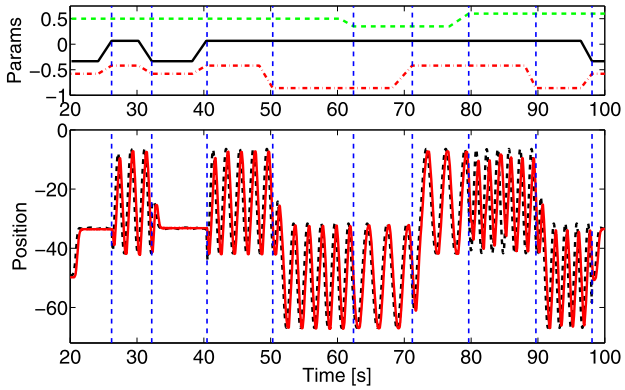
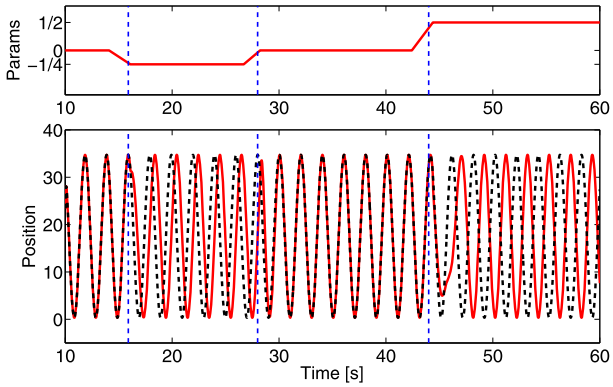


Fig. 16 Snapshots of the iCub drumming (Automatica fair, Munich, 2008). The robot is playing with both his arms and legs, the score being defined on line by a user through a graphical interface. Note that the robot is kept in a upright posture through a metallic structure. Movie available as an Online Resource, Movie 1



(a) Left Shoulder Pitch



(b) Hip Pitches

Fig. 17 Drumming trajectories. The *blue vertical lines* indicate when the new control commands are received by the limb CPG. (**a**) Simple commands sent to the UPG of the left shoulder pitch result in drumming trajectories. *Top panel:* Control commands: the target (in *red*), the amplitude (in *black*) and the frequency (in *green*). *Bottom panel:* Resulting desired and actual trajectories (in *black* and *red* respectively). (**b**) A simple parameter allows for the modulation of the phase shift of the right hip pitch relatively to the left hip pitch1. *Top panel:* Control command for the phase shifts. *Bottom panel:* Resulting trajectories: In *black* the left hip pitch and in *red* the right one

- for each leg: drumming or idle, phase shift relatively to the clock
- for the head: idle or scanning (to locate drums) or looking at one of the drums
- for the whole system: the frequency

The user can modify the score at any time, however the parameters of the CPG are changed only when it is safe for the robot, that is during the period where the limb moves away from the drums. An intermediate module (the "constraints" block on the figure) is responsible to monitor the phase of each limb and to send the new commands during the secure phase. The frequency, the amplitude and the phase shifts can be sent to the CPGs without transformation, while the drums ID are mapped to target joint angles (for each controlled DOF of the limb) defined relatively to the position of the drums. These target joint angles can be either predefined or determined through a visual tracker system combined with an inverse kinematics algorithm, as will be discussed below.

Figure 17 illustrates the trajectories obtained for drumming in open loop on the iCub robot. It can be seen that they are modulated through simple parameters, that are the target, the amplitude and the frequency for one UPG (top figure) and the phase shift between the DOFs for the CPG (bottom figure). The vertical lines on the figure indicate the time at which the CPG of the limb receives the new commands; it can be seen that the trajectories converge rapidly—in less than a cycle—to the new desired solutions. Figure 17(a) shows the possible modulations of the UPG for one DOF (the left shoulder pitch): the target, the oscillations (on and off) and the frequency, and Fig. 17(b) the control of the differences between the two hip pitches.

4.4 Visual feedback

A visual tracker based on ARToolKitPlus (see Sect. 3.3) combined with the kinematic library iKin (see Sect. 3.1) is used to obtain the (possibly time-varying) positions of the different drums (for the arms). More precisely, the different drum pads are tagged with ARToolKit markers. Once a mark is detected, the inverse kinematics algorithm iKin is used to compute suitable target joints angle to reach the pad. The target angles are stored by the controller and updated whenever a significant displacement of the drum pad is detected. Thanks to this feedback, the robot can autonomously adapt to any new configuration of the drums and to modifications of this configuration. Figure 18 shows snapshots of the robot drumming (in simulation) while the drum pads are moved.

4.5 Contact feedback

A feedback policy term was introduced to safely, and smoothly, handle collisions with the drums. We used the feedback term introduced in Sect. 2, (20). We postulate than when a collision occurs, it means that the drum pad has been hit earlier than expected (due to imprecision of the visual feedback for instance). The desired behavior for the robot is thus the following: we want the robot to stay in the current position for a while and to resume to the desired trajectory when the new beat starts. Such feedback simulates compliance on the robot and controls the way that the system resumes to the desired trajectory.

Figure 19 illustrates the effect of the feedback in a real application where the robot is drumming with a relatively high frequency (1 Hz). It can be seen that the efficiency of the feedback is related to the speed at the time of impact:

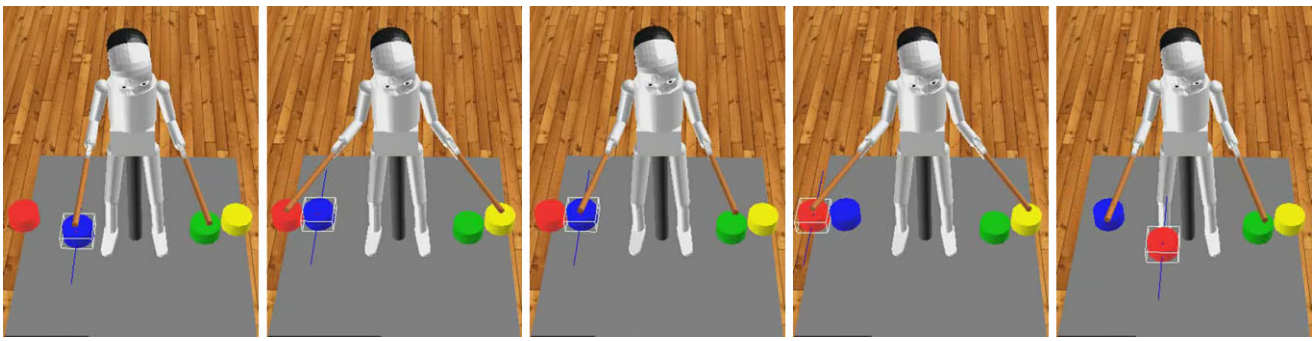


Fig. 18 Simulation of drumming with moving drums. The target angles corresponding to the different drums are updated continuously so that the robot can adapt its trajectories to their actual position. Here the robot beats alternatively the two drum pads corresponding to each arms. Every new beat the position of one of the drums is changed. Movie available as an Online Resource, Movie 3. The frames were taken at regular intervals (0.45 s)

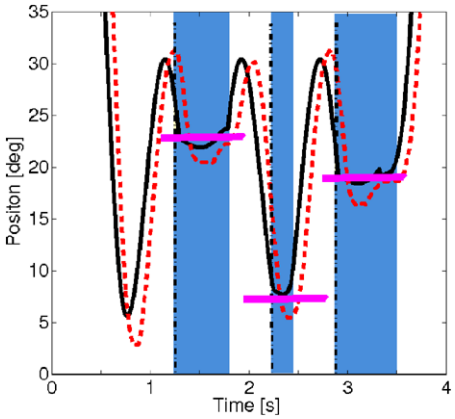


Fig. 19 Contact feedback. Right shoulder pitch trajectories while drumming. The colored areas correspond to the interval of time where the feedback is active, the *plain line* to the desired trajectory and the *dashed one* to the actual trajectory. The *horizontal thick lines* denote the approximated position of the drums. First the robot is in a rest position, then at $t \approx 5$ s, it starts drumming. During the first cycle, no contact with the drums occur, then, at $t \approx 1.2$ s, a collision is detected by the drum set and the arm is stopped in its current position. At $t \approx 1.7$ s, the normal trajectory being safe again, the feedback is removed and the arm start moving again. For the next two cycles, the drum pad is moved in different positions. It can be seen that the feedback is more efficient near the peak of amplitude (middle case), where the velocity is smaller. Snapshots of the behavior of the system with contact feedback is depicted in Fig. 20

the higher the speed, the longer it takes for the system to stabilize to the fixed point. Thus the feedback will be more efficient if the impact occurs near the peaks of amplitude (middle case on the figure). However, even when a collision occurs close to the peak of velocity (right case) the feedback successfully stabilizes the arm in its current position. Figure 20 shows snapshots of the robot adapting its trajectory to different positions of the drum pad. Note that the detection of a collision is made through the electronic drum set (as no feedback from the robot was available at the time), i.e. every time a collision is detected by the set, a message was sent to the controller indicating which drum had been hit.

5 Infant crawling

Crawling is the first stage of locomotion in infants; it allows them to explore their environments and to move towards persons or objects of interest. This behavior was implemented on the iCub in simulation and partly on the real robot; we developed a controller that allows for modulations of the locomotion, such as changes of speed and steering, and integrates both contact and visual feedback. Contact feedback is used to trigger transitions between swing and stance according to load information in order to increase locomotion stability; visual feedback is used to detect obstacles and objects of interest, and more precisely to create a map of the environment surrounding the robot. Simple reaching for marks on the ground based on vision has also been implemented, providing a demonstration of superimposition and switch between discrete and rhythmic tasks. Finally, a high-level planner algorithm based on potential fields was combined with the CPG to obtain an autonomous, infant-like behavior where the robot, attracted by an object of interest, moves towards it while avoiding obstacles, and finally reaches for it (in simulation only, for now). Steady-state crawling and reaching have been implemented on the real robot iCub.

Note that this section is partly based on work published before in conference articles, notably a previous implementation of crawling based on infant data analysis (Righetti and Ijspeert 2006a) and a study of the combination of discrete and rhythmic movements for switching between crawling and reaching (Degallier et al. 2007). The contact feedback policy was presented in Righetti and Ijspeert (2008) and the high-level planner algorithm in Gay et al. (2010). A synthetic version of this section, excluding visual feedback, reaching and the high-level planning, was presented in Degallier et al. (2008).

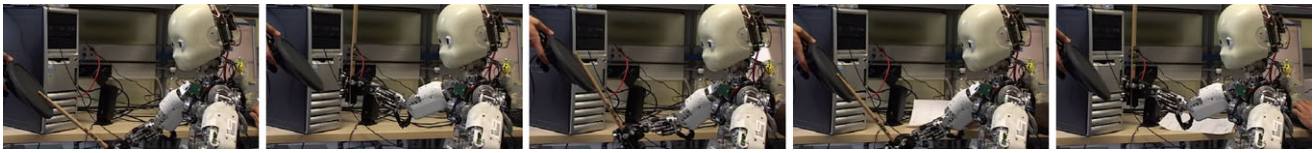


Fig. 20 Snapshots of the robot drumming with contact feedback. The drum pad is moved while the robot is drumming (between first and second snaphsots). The movement of the arm is adapted to the new situation: it is stopped as soon as it touches the drum pad. Note that the drum pad could not e held in this position without the feedback, as the robot is controlled in position, with a high gear motors. Movie available as an Online Resource, Movie 2

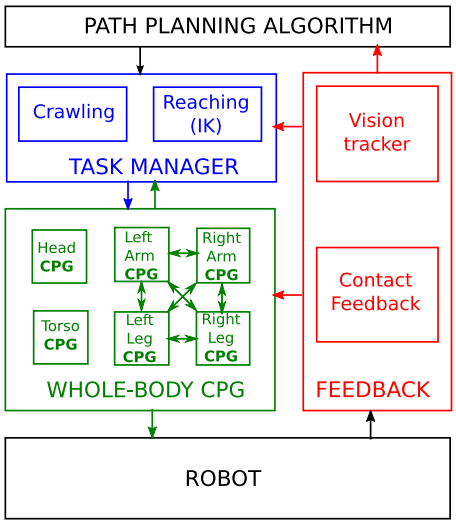


Fig. 21 Implementation of the crawling behavior. This implementation is designed so that the robot can autonomously evolve in a (time-varying) environment containing obstacles and target objects. It can be seen from the figure that the CPG output is modulated both by high-level and feedback commands. The high-level commands can be either triggered manually by the user (not depicted on the figure) or through visual feedback for reaching (if the robot is close enough to a target object) or for steering the robot to avoid obstacles and move towards target object according to a path-planning algorithm based on potential fields. Obstacles and targets are indicated by ARToolKitPlus markers. There is also a local feedback based on force sensors that modulates the behavior of the robot according to the ground contact information. Six parts are controlled, namely the head, the torso, the left arm, the right arm, the left leg and the right leg. *Green arrows* denote the couplings between the different parts

5.1 Software Implementation

The implementation, depicted on Fig. 21, consists of four main blocks:

(i) **High-level planner**: A path avoiding the obstacles and reaching points of interest is determined through an algorithm based on potential fields.
(ii) **Whole-body CPG**: the network consist of a CPG for each of the four limbs and the head and torso; the four limbs are coupled together to obtain a trot gait.
(iii) **Task manager**: a module that sends the parameters according to the task to be performed, that are here:
  (a) *Crawling*: The manager sends the parameters corresponding to crawling, with the turning angle and the speed of locomotion as open parameters.
  (b) *Reaching (IK)*: When the robot is close enough to a target, it stops and reaches for it; the joint angles for the reaching arm being provided by the iKin library.
(iv) **Feedback**: the information from the robot is used to modulate the trajectories; it consists in two subsystems
  (a) *Contact feedback*: The load information is used to control the transition of each limb between swing and stance.
  (b) *Visual tracker*: Vision is used to detect obstacles and objects of interest, indicated by predefined markers.

In this application, both arms and legs are controlled as well as the head and the torso. For each arm and leg, we actively control 4 DOFs, that are the shoulders pitch, roll and yaw and the elbows for the arms and the hips pitch, roll and yaw and the knee for the legs; the six degrees of freedom of the head and the three of the torso are also controlled. We thus actively control 22 DOFs. The remaining DOFs are set in particular position at the beginning of the task and remain fixed at that position afterwards.

5.2 CPG design and choice of parameters

The design of the low-level planner is based on observations on the crawling of human infants (Righetti 2008; Righetti and Ijspeert 2006a). While infants can have various locomotion strategies prior to walking, most of them crawl on hands and knees using a gait that is close to a walking trot. The crawling gait has a duty factor higher than 50% (i.e. the duration of the stance phase is longer than half of a step cycle). The duration of stance is highly correlated with speed of locomotion while the duration of swing remains constant, as is generally observed in quadruped mammals.

The expression for the frequency (13) is used to enable independent control of the swing and stance duration. To ensure the trot gait, the oscillators of shoulders and hips pitch joints are coupled according to (22) and (23) with the K_{ij}^{x}s set to 0 and the K_{ij}^{z}s to 1. The θ_{ij} are chosen as described for the trot in Sect. 2, Table 2.

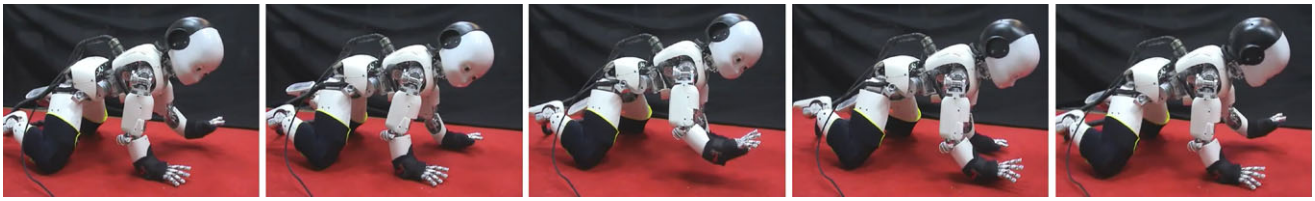


Fig. 22 Snapshots of the robot crawling. Steady state crawling, with $\omega_{\text{stance}} = \omega_{\text{swing}} = 0.3$. Note that the hands are protected by wrist sport pads because of their fragility and the knees are also covered by sport pads as they are extremely slippery due to their shape and material. The robot turns its head while walking to enhance the detection of the visual markers. Movie available as an Online Resource, Movie 4

Table 3 Parameters for crawling. Only the shoulder and the hip pitches are oscillating, and the amplitude of the shoulder is determined by the amplitude of the legs due to physical constraints (i.e. the horizontal distance covered by the hand during one step has to be equal to the one covered by the knee when the robot goes forward). The position of the shoulder roll and the elbow, and of the hip roll, is fixed during the stance but is modulated during swing to avoid contact with the ground. The resulting trajectories can be seen on Fig. 24

Left arm	γ	μ	Right arm	γ	μ	Left leg	γ	μ	Right leg	γ	μ
shoulder pitch	−1.20	0.09	shoulder pitch	−1.20	0.09	hip pitch	1.40	0.10	hip pitch	1.40	0.10
shoulder roll	0.35	−5.00	shoulder roll	0.35	−5.00	hip roll	0.40	−5.00	hip roll	0.40	−5.00
shoulder yaw	0.26	−5.00	shoulder yaw	0.26	−5.00	hip yaw	0.00	−5.00	hip yaw	0.00	−5.00
elbow	0.50	−5.00	elbow	0.50	−5.00	knee	−2.00	−5.00	knee	−2.00	−5.00

Concerning the joints other that the hips/shoulders pitches, they are controlled in the following way. The shoulder roll, the elbow and the hip roll are kept in a fixed position during the stance and move proportionally to the speed of the shoulder pitch joint during swing to ensure that the knees and the hands are lifted enough to avoid collision with the ground. Here we set

$$\gamma_i = F_i z_j$$

For the arms, j denotes the shoulder pitch joint and i the shoulder roll or the elbow joint. For the legs j denotes the hip pitch joints and i the hip roll. The F_i's are constant values chosen by hand. Figure 22 shows snapshots of the robot crawling with the parameters of Table 3 with a duty factor of 50% ($\omega_{\text{stance}} = \omega_{\text{swing}} = 0.3$).

5.3 Steering

To make the robot turn, the strategy that we used is to set the torso roll angle to a non-zero value and to modulate the amplitude of the different limbs according to the new posture of the body. More precisely, the amplitudes of the inner pitch angles (i.e. the pitch angles of the arm and the leg on the side to which the robot is turning) will be smaller and the outer angles larger to compensate the fact that the outer side has to cover a larger distance than the inner one. Once the amplitude of the legs is deduced, we deduce the ones of the arms according to the closed kinematic chain constraints. Figure 23 shows snapshots of the robot turning in simulation.

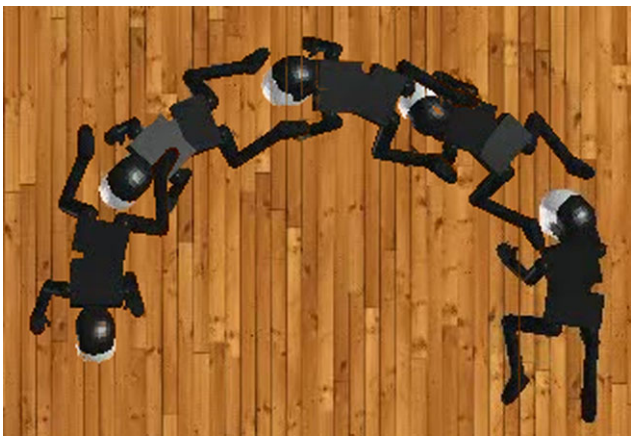
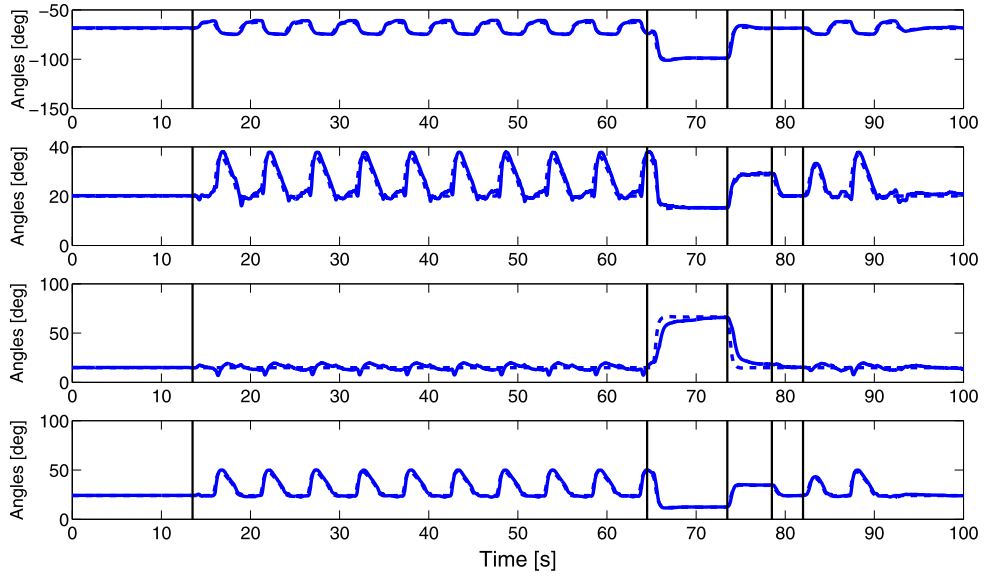


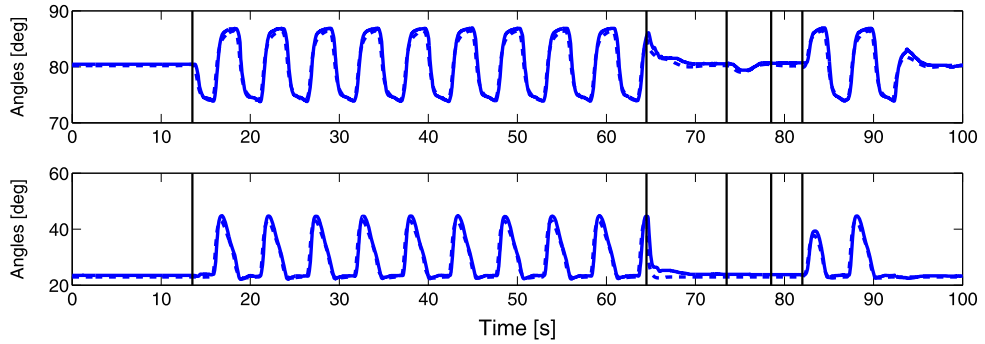
Fig. 23 The iCub turning. Superimposed snapshots of the robot turning with a torso angle of 0.5 radians. Movie available as Online Resource, Movie 6

5.4 Switching between crawling and reaching

When the robot is close enough to a target object to reach it, the task manager sends commands to stop crawling and go to a rest position that is defined by the parameters γ_i in Table 3, with all the oscillations "switched off" thanks to the Hopf bifurcation ($\mu_i < 0$). Once in this position, the robot is controlled so that it first lifts the arm that is going to reach for the object and then reaches it, as illustrated by the snapshots on Fig. 25. The intermediate position (with the reaching arm lifted) has been added to avoid contact with



(a) Right Arm



(b) Left Leg

Fig. 24 Trajectories of the (real) iCub crawling and reaching for the right arm and the left leg. At $t \approx 13$ s, the robot starts crawling. At $t \approx 64$ s the robot autonomously switch from the crawling behavior to the rest position because it is close enough to a mark to reach it. It goes to an intermediate position (in which the reaching arm—the right one here—is lifted above the ground) for 10 s and then it reaches the mark on the ground (at $t \approx 74$ s) until it is asked to crawl again ($t \approx 78$ s): it first resumes to the rest position and start crawling again after a while (at $t \approx 82$ s). *Plain lines* indicate the actual trajectories and *dotted lines* the desired ones, the tracking of the robot being quite good in general. *Top panel*: Trajectories of the four actively controlled DOFs of the right arm: shoulder pitch (*blue*), shoulder roll (*green*), shoulder yaw (*red*) and elbow (*black*). (**a**) Trajectories of the four controlled DOFs of the right arm, from top panel to bottom one: the shoulder pitch, the shoulder roll, the shoulder yaw and the elbow. (**b**) Two DOFs of the left leg: the hip pitch (*top panel*) and the hip roll (*bottom panel*). Note that the hip yaw and the knee are kept in a constant position and are not shown here. Note that Figs. 22 and 25 show snapshots of the robot crawling and reaching respectively. Movie available as an Online Resource, Movie 5

the ground during the reaching movement. The object to be reached is indicated with a ARToolKitPlus marker and the target angle positions of the reaching arm are given by the inverse kinematics algorithm iKin mentioned above, while the other limbs stay in the same position. Note that as the mark is on the ground, the range of reachable positions for the arm is quite limited and thus the tolerated error for the inverse kinematics was set to a relatively high value (5 cm). Trajectories of the iCub crawling, going to the rest position, reaching for a mark and crawling again are depicted on Fig. 24.

5.5 Contact feedback

We use the phase-dependent sensory feedback terms presented in Sect. 2 to increase locomotion stability on uneven terrains (Righetti and Ijspeert 2008). The dynamics of the oscillator and thus the policy generation is modified on line according to load sensing on the end effectors (hands and knees of the robot). This feedback loop was tested in simulation in Righetti and Ijspeert (2008). Locomotion stability was improved on various terrains such as slopes. Figure 26 gives an illustration of the effect of feedback on the locomo-

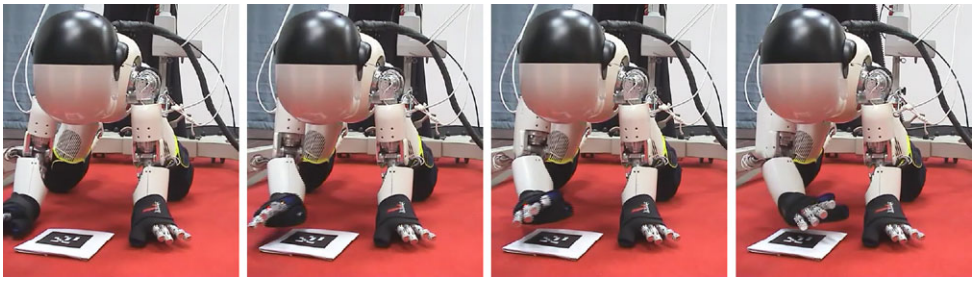


Fig. 25 Snapshots of the robot reaching a mark. When the robot is close enough to a target object (here the marker on the ground), it stops crawling and goes back to the position defined by the discrete target (g in Table 3). Then it lifts the hand that is going to do the reaching movement (second snaphsot). Finally it reaches for the target (third and fourth snapshots). Note that here the robot does not touch the ground as the maximal error tolerated was set to a high value (5 cm). Typical reaching trajectories are depicted on Fig. 24. Movie available as an Online Resource, Movie 5

(a) With Feedback

(b) Without Feedback

Fig. 26 Clearing of a 10-degree slope with and without feedback. In both experiments, the robot was initially at the same position and snapshots were taken at the same time intervals. With the feedback, the length and duration of the steps are adapted to the terrain and it can be seen that robot successfully goes through the obstacle (*top panel*), while it fails to do it in open loop (*bottom panel*). Movie available as an Online Resource, Movie 8

tion behavior. The detailed analysis of the feedback behavior, which is out of the scope of this paper, can be found in Righetti and Ijspeert (2008), Righetti (2008).

5.6 High-level path planning

A high level planner based on potential fields has been developed in (Gay et al. 2010) to illustrate how our low-level planner can be used in a simple navigation task in a fully autonomous way. A representation map of the different positions of the obstacles and targets, acquired through a vision multi-object tracking module based on ARToolKitPlus (see Sect. 3.3), is turned into a potential field where obstacles and targets are represented by respectively positive and negative potentials (Khatib 1986). Note that the standard implementation of the potential fields was slightly modified to deal with multiple targets: the closer the robot is to a target, the more attractive it is (see Gay et al. 2010). The trajectory is then given by the gradient of the surface.

The field of view of the robot was enhanced by coupling the head oscillators with the rest of the body in a way that the head and eyes of the robot perform an oscillatory movement in phase with the crawling movement to scan the environment. The positions of the markers detected during one oscillation of the head and eyes are translated in the root reference frame of the robot using iKin (see Sect. 3.1) and used to construct a partial map of the direct surroundings of the robot. This partial map is the only information available to the robot to perform navigation. No external information (full map of the environment, self localization ...) is provided to the robot.

The command sent to the manager by the path planning module is to crawl with a certain desired angle of rotation. This angle correspond to the torso roll angle and is updated whenever it is required to follow the path. If the reaching module is active (i.e. if the module is launched), whenever the robot is close enough to target marker (that is when the

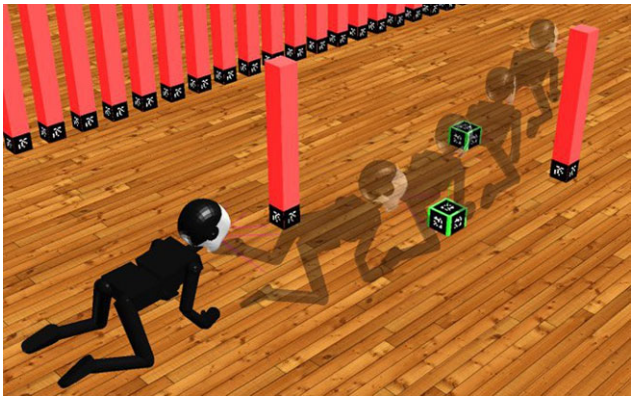


Fig. 27 Snapshots of the robot moving towards target object (*small*, *green boxes*), while avoiding obstacles (*tall*, *red boxes*). ARToolKit markers are used to track the object, with different markers for targets and obstacles. The information is sent to the path planning algorithm that computes a suitable path based on potential fields, this path is continuously updated. Movie available as an Online Resource, Movie 9

inverse kinematics module can find a solution), the behavior will be switched to reaching: the robot will stop crawling and all the steering commands will be ignored during the completion of the reaching.

Experiments were performed in randomly generated corridor-like environments with 10 goals and 15 obstacles. The distance between two obstacles or goals was constrained to be more than one meter so as to avoid conglomerates of objects and impossible situations. Results were promising with the robot being able to reach up to 9 goals out of 10, and avoid all obstacles. Figure 27 shows an example of the robot evolving among obstacles and targets (please refer to Gay et al. 2010 for more detailed results). Note that since the robot is updating its maps continuously, it can adapt to an environment with moving targets and obstacles. This simple implementation allows us to reproduce the behavior of an infant evolving in a complex and time-varying environment: the robot moves towards target and reaching for them while avoiding obstacles, in an autonomous way.

6 Discussion

We have presented here a control system that allows for the generation of both discrete and rhythmic movements based on the concept of motor primitives in biology. From a control point of view, this concept can be translated as a set of basis functions with open parameters that can be combined to generate complex trajectories. Thanks to these low-level motor primitives, the architecture that was developed allows for an extremely simple high-level control of the tasks, in the sense that the only parameters that need to be provided to the CPGs are the goals of the tasks (rather than full trajectories for instance). Such an approach is particularly well-fitted for behavior composed of stereotyped movements, such as locomotion and reaching for instance: once the nominal trajectories have been chosen, these can be easily adapted to the requirement of the task (goal of the target movement), to some behavioral choices (speed of locomotion, steering) or to environmental constraints (contact, close loop chains). In addition, the implementation using attractor dynamics is well-suited for feedback integration. The feedback can be local and act directly on the CPGs (as for instance the contact feedbacks for both drumming and crawling) or require some high-level processing and have a behavioral effect (as the high-level planning in crawling). Such strategies create a tight coupling between the controller and the environment, making the whole architecture more robust to modeling imprecisions, perturbations or time-varying environment. Note that the systematic design of feedback policies for trajectory generators is still an open, exciting research question.

In a first application to interactive drumming, we have shown both the adaptivity and the robustness of the architecture. Indeed, once the CPGs have been appropriately designed, the robot can perform any score—up to the frequency upper limit imposed by the motors and the control bandwidth—in a robust way; we were able to run the demonstration for hours with random users modulating the score that the robot was playing. In addition, thanks to the contact feedback, the interactions with the drums were ensured to be safe for the robot for any frequency. Finally, it has been shown that simple visual tracking of objects permits the on line adaptation of the movement of the robot to a changing environment.

As was illustrated in the application to crawling, the usage of a unique dynamical system for both discrete and rhythmic movements eases the switch between two totally different behaviors (crawling and reaching in our case): a unique term controlling the amplitude (μ in the equations) allows for transiting between discrete and rhythmic behaviors thanks to the Hopf bifurcation of the system. In our example, we used intermediate positions to ensure that the constraints induced by the close kinematics are fulfilled, but one could imagine to use more sophisticated techniques to compute the desired trajectory of the different limbs. The crawling behavior can also be modulated on line, both by high-level commands (speed and steering) or by feedback information (stance/swing duration, steering to avoid obstacles and reach target objects). In addition, a new implementation with gain scheduling is under development, in order to modulate on line the compliance of the robot according to the needs of the task (e.g. by having a lower stiffness during the swing than the stance).

One could argue that a major limitation of the approach is that the space of possible trajectories is limited by the chosen dynamics. It is not exactly true, as it is possible to constantly update the discrete target of the movement in order to

obtain arbitrary movement (as was illustrated on Fig. 3(d), bottom panel), even though this would be merely a sort of by-pass of the motor primitives. This is analogous to automatic movements and complex movements that follows a precise kinematic plan. In addition, the dynamics of the motor primitives can be also modulated according to the specific need of the task, as was done for instance for crawling by changing the expression of the frequency to have an independent control of the duration of the swing and the stance (Righetti and Ijspeert 2006a). However, designing a dynamical system which solution is a predefined, desired trajectory is generally not an easy task. An interesting approach to this issue is the use of adaptive frequency oscillators (AFO) that is oscillators that can learn new frequencies through entrainment (Righetti et al. 2006). It can be used for frequency analysis (Buchli et al. 2008) and Righetti and Ijspeert (2006b) used this approach to express a complex trajectory into a sum of Hopf oscillators and successfully applied it to biped locomotion. Buchli and Ijspeert (2008) used this technique to develop an adaptive locomotion controller for compliant robots that can adapt to the body properties of the robot but also to different types of gaits.

Note that the system that we are presenting here could be easily integrated to the *dynamical systems approach to behavioral organization* proposed by Schoener et al. (1995). In this approach, both the high and low-level planners are represented by dynamical systems. Applications of this approach to robotics include Steinhage and Bergener (1998), Tuma et al. (2009) and Schoener and Santos (2001) for instance.

In this contribution, we have chosen to use a system with simple attractors properties and to adapt the trajectories to the requirements of the task if needed, through feedback or time-varying control commands. A different approach to movement generation using dynamical systems, often referred to as dynamical motor primitives (DMPs), allows for learning trajectories through human demonstration (e.g., Ijspeert et al. 2003; Gribovskaya and Billard 2008; Pastor et al. 2009; Kober and Peters 2010; Ude et al. 2010). The encoded trajectories, either discrete or rhythmic (but not both), can be modulated by feedback depending on the context of the tasks. The trajectories obtained this way can be more complex and human-like than the ones obtained with our system, but they are task-specific and require learning.

A major improvement of the current approach would be to use a model of the CPGs output combined with optimization techniques to better define the parameters of the CPGs. Indeed, it is important to note that while we have made a minimal usage of model-based techniques in the applications that we have presented, more complex task could be performed through the combination of motor primitives and model-based approach dealing with multiple constraints, as for instance whole-body control approaches such as in Sentis and Khatib (2005) or in the case of locomotion, such as Zico Kolter and Ng (2009), Kalakrishnan et al. (2010) or Zucker et al. (2010). Since our framework simply generates desired policies, it could be easily integrated with modern torque control techniques, in a similar way that Zico Kolter and Ng (2009) and Zucker et al. (2010) used splines. The main advantage of using differential equations over splines is that external signals can be embedded into the dynamics (e.g. for synchronization or to deal with perturbations). In particular, we think that our approach, combined with a high-level planning system, is particularly well-suited for locomotion, as the same system integrates the primitives needed for both rhythmic motion generation and posture control. Note that Kimura and his group obtained excellent results for locomotion based on CPGs and bio-inspired reflexes (Kimura et al. 2007; Maufroy et al. 2008), but here our goal is to provide a more general approach to movement generation.

In this article, our main focus was robotic application. However, another promising direction of research would be the investigation of the basic principles of coordination in humans. Indeed, as first postulated by Bernstein (1967), functional units (*synergies*) may exist that constrain movements, thus reducing redundancy. Uncovering such coordination structures would not only be beneficial to the study of human motor control, but it could also be used to simplify robotic controllers. For instance, it is known from motor control experiments that the amplitude and the frequency of a movement are linked; the pertinence of such dependencies for robotic applications, in particular for locomotion, could provide an interesting way to reduce the dimension of the control parameters.

7 Conclusion

The model that we have presented can be seen as a simple trajectory generator for both discrete and rhythmic movements that is easy to control and that can be modulated on line according to new control commands and/or feedback. Such a generator drastically reduces the planning as only the key characteristics of the movements need to be specified, namely the target of the discrete movements g_i, and the amplitude $\sqrt{[\mu_i]^+}$ and the frequency ω_i of the rhythmic one. In addition, the global attractiveness of the solutions ensures robustness against perturbations, but also the capacity of the system to adapt to changing environments through feedback information. It has been shown that it can be efficiently used for diverse applications on real robots such as drumming, crawling and reaching. The three main advantages of the approach are that (i) the planning phase is simplified thanks to the motor primitives, in the sense that the control commands that are required are reduced to the

key characteristics of the movement (the target for discrete movements and the amplitude and frequency for rhythmic movements), (ii) switching between behaviors is made easier by the fact that the same system can be used for all kind of tasks, either discrete and rhythmic, and (iii) the dynamics of the motor primitives can be modulated by sensory feedback in order to obtain an adaptive behaviors. In addition, this method has a low computational cost and is well-fitted for applications requiring fast control loops.

Acknowledgements We would like to thank Francesco Nori, Lorenzo Natale and Giorgio Metta from the Italian Institute of Technology for their help with the implementation of drumming and crawling on the iCub robot.

References

Bernstein, N. (1967). *The co-ordination and regulation of movements*. London: Pergamon.

Bizzi, E., Accornero, N., Chapple, W., & Hogan, N. (1984). Posture control and trajectory formation during arm movement. *The Journal of Neuroscience*, *4*(11), 2738–2744.

Bizzi, E., Cheung, V. C. K., d'Avella, A., Saltiel, P., & Tresch, M. (2008). Combining modules for movement. *Brain Research Reviews*, *57*(1), 125–33.

Buchli, J., & Ijspeert, A. J. (2008). Self-organized adaptive legged locomotion in a compliant quadruped robot. *Autonomous Robots*, *25*(4), 331–347.

Buchli, J., Righetti, L., & Ijspeert, A. (2008). Frequency analysis with coupled nonlinear oscillators. *Physica D*, *237*, 1705–1718.

Bullock, D., & Grossberg, S. (1988). The VITE model: a neural command circuit for generating arm and articulator trajectories. In J. Kelso, A. Mandell, & M. Shlesinger (Eds.), *Dynamic patterns in complex systems* (pp. 206–305). Singapore: World Scientific.

Capaday, C. (2002). The special nature of human walking and its neural control. *Trends in Neurosciences*, *25*(7), 370–376.

Cui, X., Zhu, Y., Zang, X., Tang, S., & Zhao, J. (2010). CPG based locomotion control of pitch-yaw connecting modular self-reconfigurable robots. In *Information and automation (ICIA), 2010 IEEE international conference on* (pp. 1410–1415).

De Rugy, A., & Sternad, D. (2003). Interaction between discrete and rhythmic movements: reaction time and phase of discrete movement initiation during oscillatory movements. *Brain Research*, *994*(2), 160–174.

Degallier, S., & Ijspeert, A. (2010). Modeling discrete and rhythmic movements through motor primitives: a review. *Biological Cybernetics*, *103*(4), 319–338.

Degallier, S., Santos, C. P., Righetti, L., & Ijspeert, A. (2006). Movement generation using dynamical systems: a humanoid robot performing a drumming task. In *IEEE-RAS inter. conf. on humanoid robots* (pp. 512–517).

Degallier, S., Righetti, L., & Ijspeert, A. (2007). Hand placement during quadruped locomotion in a humanoid robot: a dynamical system approach. In *IEEE-RAS international conference on intelligent robots and systems (IROS07)*.

Degallier, S., Righetti, L., Natale, L., Nori, F., Metta, G., & Ijspeert, A. (2008). A modular bio-inspired architecture for movement generation for the infant-like robot icub. In *Proceedings of the second IEEE RAS/EMBS international conference on biomedical robotics and biomechatronics, BioRob*.

Fitts, P. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, *47*(6), 381–391.

Fitzpatrick, P., Metta, G., & Natale, L. (2008). Towards long-lived robot genes. *Robotics and Autonomous Systems*, *56*(1), 29–45.

Frigon, S., & Rossignol, S. (2006). Experiments and models of sensorimotor interactions during locomotion. *Biological Cybernetics*, *95*(6), 607–627.

Gay, S., Degallier, S., Pattacini, U., Ijspeert, A., & Santos, J. (2010). Integration of vision and central pattern generator based locomotion for path planning of a nonholonomic crawling humanoid robot. In *Proceedings of the 2010 IEEE/RSJ international conference on intelligent robots and systems (IROS 2010)*, Taipei.

Gribovskaya, E., & Billard, A. (2008). Combining dynamical systems control and programming by demonstration for teaching discrete bimanual coordination tasks to a humanoid robot. In *Proceedings of 3rd ACM/IEEE international conference on human-robot interaction, HRI'08*, Amsterdam, 12–15 March 2008.

Grillner, S. (2006). Biological pattern generation: the cellular and computational logic of networks in motion. *Neuron*, *52*(5), 751–766.

Hersch, M., & Billard, A. (2008). Reaching with multi-referential dynamical systems. *Autonomous Robots*, *25*(1–2), 71–83.

Ijspeert, A., Nakanishi, J., & Schaal, S. (2002). Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of the IEEE/RSJ int. conference on intelligent robots and systems (IROS2002)* (pp. 958–963).

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In S. T. Becker & K. Obermayer (Eds.), *Neural information processing systems 15 (NIPS2002)* (pp. 1547–1554).

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., & Schaal, S. (2010). Fast, robust quadruped locomotion over challenging terrain. In *IEEE international conference on robotics and automation (ICRA10)*.

Kelso, J. A. S., Southard, D. L., & Goodman, D. (1979). On the nature of human interlimb coordination. *Science*, *203*(4384), 1029–1031.

Khatib, O. (1980). *Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacles*. PhD thesis, Ecole Nationale Supérieure de l'aéronautique et de l'espace, Toulouse, France.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, *5*(1), 90–98.

Kimura, H., Fukuoka, Y., & Cohen, A. H. (2007). Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *The International Journal of Robotics Research*, *26*(5), 475–490.

Kober, J., & Peters, J. (2010). Imitation and reinforcement learning. *IEEE Robotics & Automation Magazine*, *17*(2), 55–62.

Kose-Bagci, H., Dautenhahn, K., Syrdal, D. S., & Nehaniv, C. L. (2010). Drum-mate: interaction dynamics and gestures in human-humanoid drumming experiments. *Connection Science*, *22*(2), 103–134.

Matsuoka, K. (1985). Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, *52*, 367–376.

Maufroy, C., Kimura, H., & Takase, K. (2008). Towards a general neural controller for quadrupedal locomotion. *Neural Networks*, *21*(4), 667–681.

Michel, O. (2004). Webots tm: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, *1*, 39–42.

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *International conference on robotics and automation (ICRA 2009)*.

Righetti, L. (2008). *Control of legged locomotion using dynamical systems*. PhD thesis, EPFL, Lausanne.

Righetti, L., & Ijspeert, A. (2006a). Design methodologies for central pattern generators: an application to crawling humanoids. In *Proceedings of robotics: science and systems*, Philadelphia, USA.

Righetti, L., & Ijspeert, A. (2006b). Programmable central pattern generators: an application to biped locomotion control. In *Proceedings of the 2006 IEEE international conference on robotics and automation*.

Righetti, L., & Ijspeert, A. (2008). Pattern generators with sensory feedback for the control of quadruped locomotion. In *Proceedings of the 2008 IEEE international conference on robotics and automation (ICRA 2008)* (pp. 819–824).

Righetti, L., Buchli, J., & Ijspeert, A. (2006). Dynamic hebbian learning in adaptive frequency oscillators. *Physica D*, *216*(2), 269–281.

Ronsse, R., Sternad, D., & Lefèvre, P. (2009). A computational model for rhythmic and discrete movements in uni- and bimanual coordination. *Neural Computation*, *21*(5), 1335–1370.

Ronsse, R., Vitiello, N., Lenzi, T., van den Kieboom, J., Carrozza, M., & Ijspeert, A. (2010). Human-robot synchrony: flexible assistance using adaptive oscillators. *IEEE Transactions on Biomedical Engineering*, (99), 1. doi:10.1109/TBME.2010.2089629

Schaal, S., Kotosaka, S., & Sternad, D. (2000). Nonlinear dynamical systems as movement primitives. In *International conference on humanoid robotics (Humanoids00)* (pp. 117–124). Berlin: Springer.

Schoener, G. (1990). A dynamic theory of coordination of discrete movement. *Biological Cybernetics*, *63*, 257–270.

Schoener, G., & Kelso, J. A. S. (1988). Dynamic pattern generation in behavioral and neural systems. *Science*, *239*(4847), 1513–1520.

Schoener, G., & Santos, C. (2001). Control of movement time and sequential action through attractor dynamics: a simulation study demonstrating object interception and coordination. In *Neurons, networks, and motor behavior*.

Schoener, G., Dose, M., & Engels, C. (1995). Dynamics of behavior: theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, *16*(2–4), 213–245.

Sentis, L., & Khatib, O. (2005). Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, *2*(4), 505–518.

Sproewitz, A., Pouya, S., Bonardi, S., van den Kieboom, J., Moeckel, R., Billard, A., Dillenbourg, P., & Ijspeert, A. (2010). Roombots: reconfigurable robots for adaptive furniture. *IEEE Computational Intelligence Magazine*, special issue on "Evolutionary and developmental approaches to robotics".

Steinhage, A., & Bergener, T. (1998). Dynamical systems for the behavioral organization of an anthropomorphic mobile robot. In *Proceedings of the fifth international conference on simulation of adaptive behavior on from animals to animats 5* (pp. 147–152). Cambridge: MIT Press.

Tsagarakis, N., Metta, G., Sandini, G., Vernon, D., Beira, R., Becchi, F., Righetti, L., Santos-Victor, J., Ijspeert, A., Carrozza, M., & Caldwell, D. (2007). iCub—the design and realization of an open humanoid platform for cognitive and neuroscience research. *International Journal of Advanced Robotics*, *21*(10), 1151–1175. Special Issue on Robotic platforms for Research in Neuroscience.

Tuma, M., Iossifidis, I., & Schoner, G. (2009). Temporal stabilization of discrete movement in variable environments: an attractor dynamics approach. In *Robotics and automation, 2009. ICRA '09. IEEE international conference on* (pp. 863–868).

Turvey, M. (1990). Coordination. *The American Psychologist*, *45*(8), 938–953.

Ude, A., Gams, A., Asfour, T., & Morimoto, J. (2010). Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, *26*(5), 800–815.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*, 25–57.

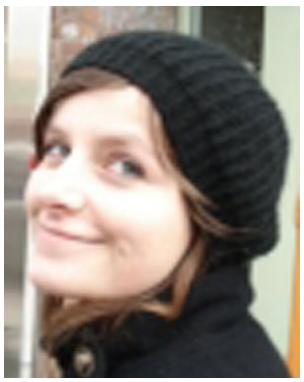
Wagner, D., & Schmalstieg, D. (2007). Artoolkitplus for pose tracking on mobile devices. In *Proceedings of 12th computer vision winter workshop (CVWW'07)*.

Williamson, M. (1999). *Robot arm control exploiting natural dynamics*. PhD thesis, MIT Department of Electrical Engineering and Computer Science.

Won, J., & Hogan, N. (1995). Stability properties of human reaching movements. *Experimental Brain Research*, *107*(1), 125–136.

Zico Kolter, J., & Ng, A. Y. (2009). Task-space trajectories via cubic spline optimization. In *Proceedings of the 2009 IEEE international conference on robotics and automation*, Kobe, Japan (pp. 2364–2371). New York: IEEE Press.

Zucker, M., Bagnell, J. A. D., Atkeson, C., & Kuffner, J. (2010). An optimization approach to rough terrain locomotion. In *IEEE conference on robotics and automation*.



Sarah Degallier has a B.Sc./M.Sc. in Mathematics from the Ecole Polytechnique fédérale de Lausanne (2005) and a doctorate (Dr. in Science) from EPFL, under the supervision of Prof. Auke Ijspeert, in November 2010. She now works as a research associate for the CNBI Lab at EPFL. Her main research interests are nonlinear dynamical systems, locomotion, reaching and, more generally, humanoid robotics.

Ludovic Righetti received a Diploma (eq.M.Sc.) in Computer Sciences from the Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, in March 2004 and a doctorate (Dr. in Science) from EPFL, under the supervision of Prof. Auke Ijspeert, in November 2008. His research interests include robot control, nonlinear dynamical systems and complex systems in general, animal locomotion and computational neurosciences.

Sebastien Gay graduated in 2007 as Informatics Engineer at the National Institute of applied Sciences (INSA) in Lyon (France). He received a MSc with honours in Artificial Intelligence from the University Claude Bernard in Lyon in 2007. He joined the Biorobotics Laboratory (formerly BIRG) for the first time in 2007 to carry out his master project about reconfiguration of modular robots. He started his Ph.D. in April 2009, hosted by both the Biorobotics Laboratory at EPFL in Lausanne and the VisLab at the IST in Lisbon. He was integrated in the RobotCub project, a European attempt build an infant like humanoid robot to study cognition and it's development in children, which ended in January 2010. He is

now attached to the Amarsi project about designing adaptive locomotion modules to develop rich motor skills in compliant legged robots. He's main research interests are the coordination of vision and locomotion in legged robots, stabilization of compliant legged robots using vision, locomotion on rough terrain.

Auke Ijspeert is an associate professor at the EPFL (the Swiss Federal Institute of Technology at Lausanne), and head of the Biorobotics Laboratory (BioRob). He has a B.Sc./M.Sc. in Physics from the EPFL, and a Ph.D. in artificial intelligence from the University of Edinburgh (with John Hallam and David Willshaw as advisors). He carried out postdocs at IDSIA and EPFL (LAMI) with Jean-Daniel Nicoud and Luca Gambardella, and at the University of Southern California (USC), with Michael Arbib and Stefan Schaal. He then became a research assistant professor at USC, and an external collaborator at ATR (Advanced Telecommunications Research institute) in Japan. In 2002, he came back to the EPFL first as a SNF assistant professor, and since October 2009 as an associate professor (with tenure). His research interests are at the intersection between robotics, computational neuroscience, nonlinear dynamical systems, and applied machine learning. He is interested in using numerical simulations and robots to get a better understanding of animal locomotion and movement control, and in using inspiration from biology to design novel types of robots and locomotion controllers. With his colleagues, he has received the Best Paper Award at ICRA2002, the Industrial Robot Highly Commended Award at CLAWAR2005, and the Best Paper Award at the IEEE-RAS Humanoids 2007 conference. He is an associate editor for the IEEE Transactions on Robotics, and has acted as guest editor for the IEEE Transactions on Biomedical Engineering, Autonomous Robots, and Biological Cybernetics.