ROBotic Open-architecture Technology for
Cognition, Understanding and Behavior

Cogsys
Cognitive Systems

Project No. 004370

RobotCub
Development of a Cognitive Humanoid Cub

Instrument:	Integrated Project
Thematic Priority:	Cognitive Systems

D 3.4 A controller architecture for the combination of rhythmic and discrete movements

Due date: **29/02/2007**
Submission Date: **29/02/2007**

Start date of project: **01/09/2004**

Duration: **60 months**

Organisation name of lead contractor for this deliverable: **EPFL**

Responsible Person: **Sarah Degallier, Ludovic Righetti and Auke Ijspeert**

Revision: **1.0**

Project co-funded by the European Commission within the Sixth Framework Programme (2004-2009)		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Introduction

In the framework of the RobotCub project, we are currently developing a functional model of the human motor system. This motor architecture has been designed to be easily integrated in the *iCub* Cognitive Architecture developed by the RobotCub consortium [15].

We present here the current implementation of a functional architecture that allows for the generation of discrete and rhythmic movements. The architecture consists in three independent modules (the planner, the manager and the generator) that can be used independently and easily replaced by other modules if needed.

We aim at designing an architecture that is suitable for generating complex, adaptive behaviors and for switching between these behaviors. This requires the ability to pertinently integrate sensory feedback and high level commands, but also the ability to deal with constraints.

The current implementation allows for an easy and fast online modulation of trajectories as well as the possibility of easily switching between behaviors according to sensory information; this will briefly be illustrated through two applications, namely *interactive drumming* and the switching between *crawling*, *reaching* on the fours and *reaching while crawling*. *Interactive drumming* has already been implemented on the real robot while switching between behaviors has been tested using the physics based simulator Webots^{TM}.

Note that as our particular interest is low level movement generation, we do not focus on the high cognitive abilities needed to define and choose the action. Such questions are treated by other labs in the framework of the RobotCub project[1].

[1] see `http://www.robotcub.org/misc/review3/papers.html` for a list of publications.

Chapter 1

Presentation of the architecture

Movement generation in humans appears to be processed through a three-layered architecture [6, 13], where each layer corresponds to a different level of abstraction in the representation of the movement. In our model, those levels are functionally defined as the planner, the manager and the generator.

The planner (i.e the motor cortex in humans) builds the mental representation of the task. The manager (the brain stem, the basal ganglia and the cerebellum in humans) is involved in the selection, timing and coordination of the appropriate behaviors (motor programs). Finally, the generator (the spinal cord) generates trajectories through central pattern generators, i.e. networks of neurons involved in the production of movement primitives.

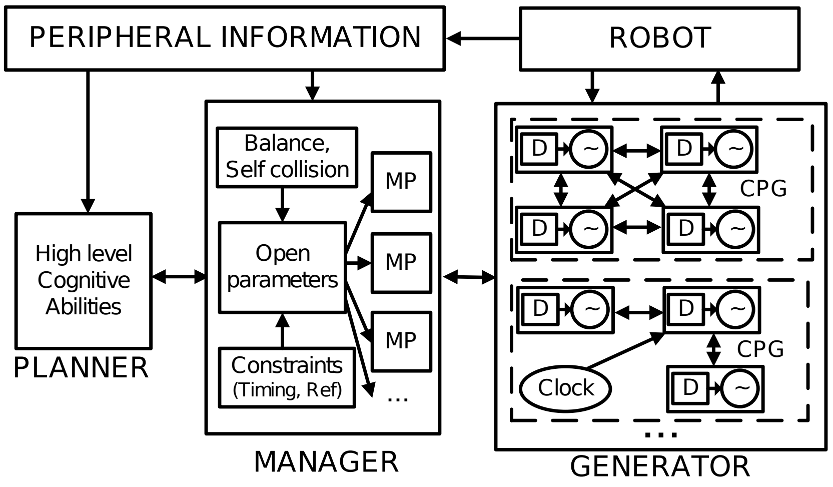


Figure 1.1: Schematic of the functional organization of the architecture. Both at the manager and at the generator levels, the movement is decomposed into modules, that is, respectively, into motor programs (MP), which represent behaviors, and into central pattern generators (CPG), which are networks of couple unit PGs responsible of the generation of discrete and rhyhthmic motor primitives. For the CPGs, two possible examples of networks are represented: the upper one correspond to the network used for locomotion and the lower one is an illustration of the possible usage of a clock.

Sensory feedback seems to be also distributed along the motor structure in the same three-layered fashion, see for instance [6, 13], accordingly to its degree of processing, namely (i) local information (cutaneous information, state of the muscles, load, etc.), that is required for fast, protective movements (reflexes), (ii) contextual information (balance, position of objects for reaching, timing, etc.), that triggers learned responses to the specific context (automatisms) and finally at the higher level, (iii) semantic information (signification of the environment, state of mind, etc.), that contributes to the mental representation of the context involved in voluntary movements.

We make the assumption that movements are generated in a modular way, both in terms of pattern generator (PG) (movement primitives) and of motor programs (MP) (i.e predefined, time-varying combinations of movement primitives) found in biology [5, 14, 7]. Such an assumption turns a high dimensional trajectory generation into a simple selection between predefined behaviors; it is thus well suited for fast adaptive behaviors. Thanks to the use of dynamical systems, those behaviors can be easily and smoothly modulated by simply changing the parameters.

We have built an architecture where each layer is an independent process that communicates with the others (see Fig. 1.1); such a structure allows for the distribution of the tasks relatively to their cognitive level, but also to their computational need. Indeed, the role of the generator consists in our case only in integrating the dynamical systems, a task which requires low computational needs and can be implemented on the DSP controllers of the robot with fast feedback loops. Such an approach ensures that the generation of the trajectories is not perturbed by highly demanding processes such as optimization and planning which are solved at higher levels (i.e. the planner in our case). The manager ensures the coherence of the movements (both in terms of spatial and time constraints) and the communication between the high and the low level processes.

1.1 The generator

The generator, which is responsible for the generation of the trajectories, is built on the concept of central pattern generators (CPGs), that we take in the sense of a network of unit generators (UGs) of basic movements called motor primitives.

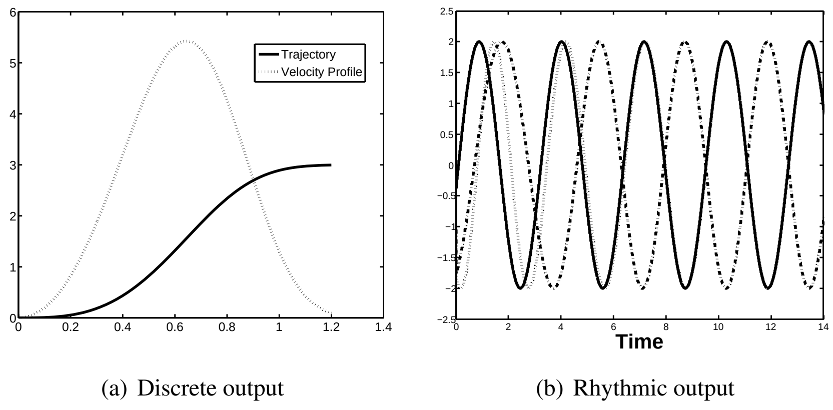
Unit generators are modeled in our architecture by dynamical systems; two types of primitives are defined, namely discrete and rhythmic, that correspond respectively to the solution of a globally attractive fixed point system and of a limit cycle system. The two main advantages of using such dynamical systems is that (i) the trajectories are generated online by integration, and thus their parameters can be modified smoothly on the fly, and (ii) the solutions obtained are robust to perturbations, and thus feedback can be easily included.

All trajectories (for each joint) are generated through a unique set of differential equations, which is designed to produce complex movements modeled as a periodic movement around a time-varying offset. More precisely, complex movements are generated through the superimposition and sequencing of simpler motor primitives generated by rhythmic and discrete unit generators. The discrete primitive is injected in the rhythmic primitive as an offset, although other combinations of them could be considered such as a sequencing or a simple addition of their output.

The discrete UG is modeled by the following system of equations

$$\dot{h}_i = d(u - h_i) \tag{1.1}$$
$$\dot{y}_i = h_i^4 v_i \tag{1.2}$$
$$\dot{v}_i = u^4 \frac{-b^2}{4} (y_i - g_i) - b\, v_i. \tag{1.3}$$

The system is critically damped so that the output y_i of Eqs 1.2 and 1.3 converges asymptotically and monotically to a goal g_i with a speed of convergence controlled by b, whereas the speed v_i converges to zero. The first equation ensures a bell-shaped velocity profile and is reset to zero at the end of each movement. The output of such a system is depicted on Fig. 1.2(a).



(a) Discrete output (b) Rhythmic output

Figure 1.2: (a) Discrete motor primitive generated by the discrete pattern generator (plain line) and its velocity profile (dotted line) (b) Rhythmic motor primitives generated by the rhythmic pattern generators - synchronization due to coupling can be clearly seen.

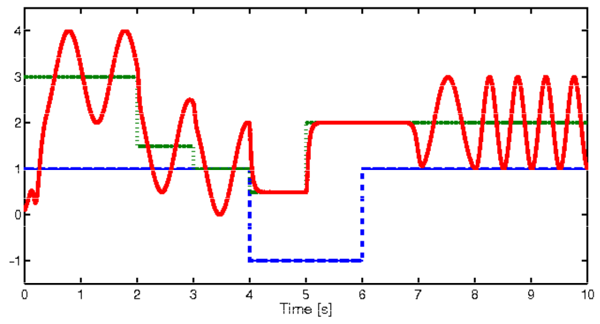


Figure 1.3: Using simple variations of the parameters m_i (dash line), g_i (dotted line) and ω_i (not represented on the figure), combinations of discrete and rhythmic movements can be easily generated.

Concerning the rhythmic UG, it is modeled by a system with a Hopf bifurcation:

$$\dot{x}_i = a\left(m_i - r_i^2\right)(x_i - y_i) - \omega_i z_i \tag{1.4}$$

$$\dot{z}_i = a\left(m_i - r_i^2\right) z_i + \omega_i (x_i - y_i) + \sum k_{ij} z_j + u_i) \tag{1.5}$$

$$\omega_i = \frac{\omega_{down}}{\mathrm{e}^{-bz_i} + 1} + \frac{\omega_{up}}{\mathrm{e}^{bz_i} + 1} \tag{1.6}$$

where $r_i = \sqrt{(x_i - y_i)^2 + z_i^2}$. When $m_i > 0$, Eqs. 1.4 and 1.5 describe an Hopf oscillator whose solution x_i is a periodic signal of amplitude $\sqrt{m_i}$ and frequency ω_i with an offset given by g_i (see Fig. 1.2(b)) . A Hopf bifurcation occurs when $m_i < 0$ leading to a system with a globally attractive fixed point at (g_i,0). The term $\sum k_{ij} z_j$ controls the couplings with the other joints j; the k_{ij}'s denote the gain of the coupling between joints i and j. The expression used for ω_i allows for an independent control of the speed of the ascending and descending phases of the periodic signal (see [11] for more details). Finally the term u_i is a control term generated by feedback information (for a possible implementation of the feedback, see Sec. 2.2 or [12]) .

Qualitatively, by simply modifying on the fly the parameters g_i and m_i, the system can switch between purely discrete movements ($m_i < 0, g_i = g(t)$), purely rhythmic movements ($m_i > 0, g_i =$cst), and combinations of both ($m_i > 0, g_i = g(t)$) (see [2] for more details). Simple inputs can thus lead to relatively complex trajectories as illustrated on Fig. 1.3. More elaborate movements can be achieved by sending time-varying parameters to the system and by integrating feedback signals to the generator.

Thanks to the use of limit cycle systems, the different unit generators of each joints can be coupled to form a network that will allow for intra and interlimb coordination. Such networks, that we call central pattern generators (CPGs), are well suited to ensure fixed time relations between the different rhythmic outputs[1], a feature which is particularly convenient for generating different gaits for locomotion for instance [11]. Moreover, a reference limit cycle system can be added in the system to serve as a clock to cope with phase resettings due to bifurcations. We have used such a clock in drumming application to serve as a metronome (i.e. as an absolute time referential).

1.2 The manager

The manager is built upon the concept of motor program, which is defined as "*a set of muscle commands which are structured before a movement begins and which can be sent to the muscle with the correct timing so that the entire sequence is carried out in the absence of peripheral feedback*" by Marsden et al. [7]. This concept is a nice way of explaining the rapidity with which we react to stimuli and the stereotypy present in human movements. Moreover, the notion of generalized motor program (MP), that is motor programs with open parameters, allows the generation of movements adapted to the environment (see [13] for instance).

Functionally speaking, the manager is mainly responsible for sending the right parameters (in joint space) to the generator, at the right timing. We define a (generalized) motor program (MP) as a sequence of parameters sent to the generator to produce the desired trajectories, that is in our case $\vec{g}(t)$, $\vec{m}(t)$, $\vec{\omega}(t)$ and the couplings between the oscillators (i.e. the topology of the network). Some of the parameters are fixed (the coupling between the limbs for crawling for instance), others are open and need to be defined relatively to the environment and the task (the desired angles in reaching). An inverse kinematics is also needed to transform task space goals into target joint angles.

Every time a MP is launched by the manager, the first command sent corresponds to a predefined initial position. The parameters are then sent at regular time intervals to the generator. At the end of the sequence, a command corresponding to a final target position is sent. This makes the switching between tasks easier, as will be illustrated with crawling and reaching. A MP can be elicited either by the planner (voluntary movements) or by the contextual sensory information (automatisms).

At the manager level, the role of the feedback is mainly to adapt the movement to the contextual constraints as the position of an object to be reached or balance issues. The manager can adapt the movement through the parameters of the GMPs, for instance by activating a repulsor in the adequate direction to prevent a self collision.

1.3 The planner

In the current implementaiton, the planner is a simple GUI that allows the user to specify the task the robot has to perform. However, higher cortical abilities can be implemented to define the task to be performed. The output that should be given to the manager is the target goal (or the target trajectory) in cartesian space.

We are currently extending the architecture so that the planner output to the manager can be any trajectory in task space so that any partner in the consortium could use the manager and the generator

[1]The interested readers are referred to the work of Golubitsky for the construction of networks of coupled oscillator, see for instance [4]

for low level movement generation. The given trajectory will be converted to joint space commands by the manager and then executed by the generator. Note that if the trajectory are defined in joint space, low level movement generation can be achieved by using the genertor only.

1.4 Link with the *iCub* cognitive architecture

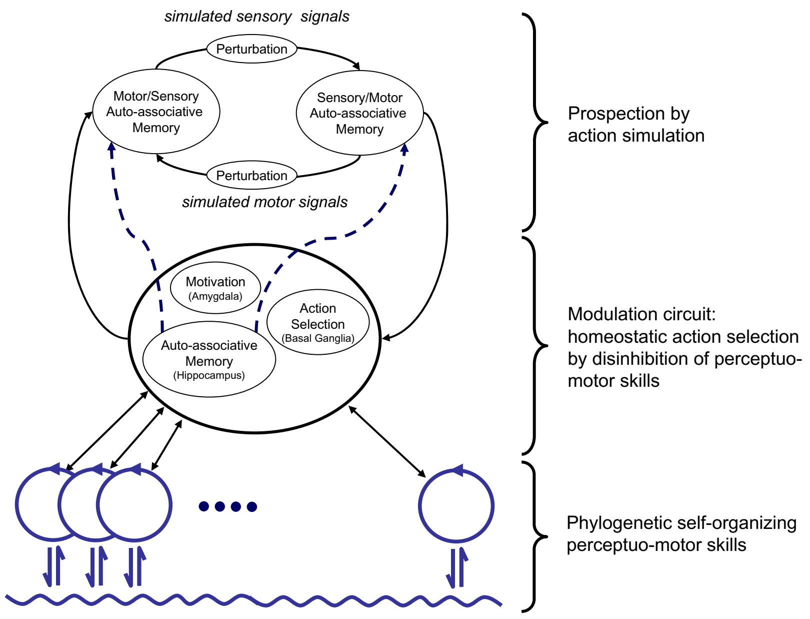

simulated sensory signals
Perturbation
Motor/Sensory Auto-associative Memory
Sensory/Motor Auto-associative Memory
Perturbation
simulated motor signals
Motivation (Amygdala)
Action Selection (Basal Ganglia)
Auto-associative Memory (Hippocampus)
Prospection by action simulation
Modulation circuit: homeostatic action selection by disinhibition of perceptuo-motor skills
Phylogenetic self-organizing perceptuo-motor skills

Figure 1.4: The *iCub* cognitive architecture (figure taken from [15]).

Even if defined only in terms of low level movement generation, the architecture that we have presented is closely related to the *iCub* cognitive architecture depicted on Fig. 1.4 (see [15]). Indeed, functionally speaking, the planner is part of the *prospection by simulation* layer, the manager of the *modulation circuit* and the generator of the *phylogenetic self-organizing perceptuo-motor skills*.

Chapter 2

Applications

We present here two possible applications of the system. We first apply the system to interactive drumming and to reachingon the fours and crawling, as these tasks require combination of primitives, but also synchronicity, switching between discrete and rhythmic movements and fast online modulations of parameters.

2.1 Drumming

Interactive drumming is an interesting task combining discrete and rhythmic movements because it requires the usage of all of the four limbs, precise timing, coordination between limbs and also the online modulation of the trajectories subject to constraints. As it does not require high level cognitive abilities or balance issues, it is well suited for a first test of the architecture on the real robot.

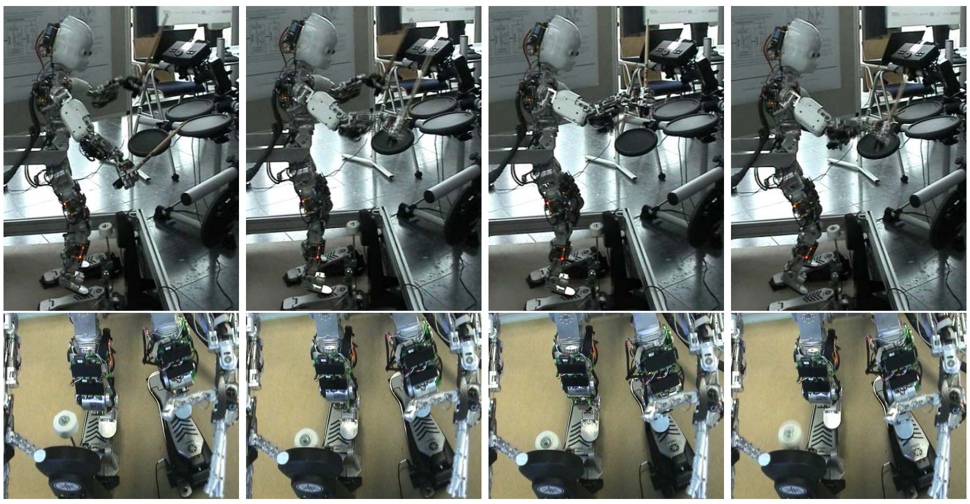


Figure 2.1: Snapshots of the *iCub* drumming at CogSys.

Drumming has been implemented on the real *iCub* and presented as a demonstration at the CogSys 2008 conference in April (see Fig. 2.1 for some pictures of the setting). 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. The sticks are grasped by the hands which remain fixed afterwards. We control actively four joints for each limb and one for the head.

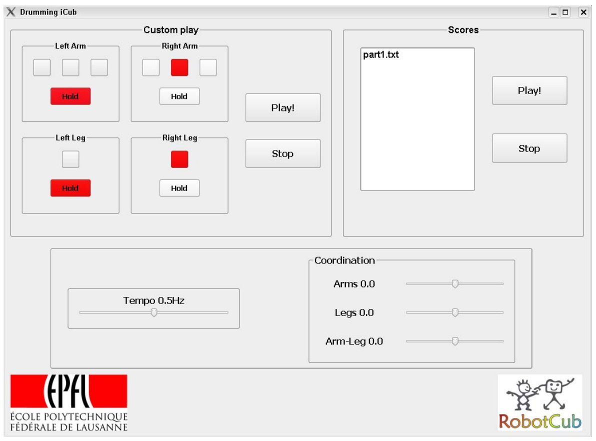


Figure 2.2: A picture of the GUI we used for drumming. The user can either specify a score to be played (right part of the GUI) or create a score online by specifying the drums on which the robot has to beat (left part). Coordination between the limb and the frequency of beating can also be modified online.

The planner is simply implemented as a GUI (see Fig. 2.2) that allows a user to define the score that the robot has to play; more precisely, the user can specify either a predefined score or the drum that has to be beaten next by each of the limbs. There is a ”Hold” position corresponding to the mode where the limb does not beat anything. Moreover, the coordination between the limbs (i.e. their phase relation) and the frequency of the beating can be modified on the fly.

At the manager level, there is a unique motor program for each limb (MP) whose parameters are controlled through the GUI. The manager is then responsible to translate the commands sent by the planner in terms of joint space parameters for the generator. For now on, the target discrete postures for hitting each drum are predefined; the integration of visual localization of the drums is planned as a future work. Those parameters are sent at a specific timing corresponding to the beat (i.e. the tempo) of the score, this beat being given by the clock at the generator level.

Concerning the generator, each dof is controlled by the discrete and rhythmic pattern generators that we have presented in chapter 1. The couplings between the dofs, which are specified through a configuration file, are the following:

- for the legs, both hips are unilaterally coupled to the clock and bilaterally coupled to the knees
- for the arms, both shoulders are unilaterally coupled to the clock and bilaterally coupled to the elbows

The bilateral couplings between limbs allows for a precise synchronization between them. After a Hopf bifurcation, one can observe a phase resetting of the oscillators; the clock can be seen as a metronome that ensures that the limbs stay in synchronization with the absolute tempo despite those phase resettings.

The implementation of the real *iCub* has successfully shown that the architecture was well-suited to allow for the online modulation of trajectories subject to time constraints (Fig. 2.3) as well as for the generation of synchronized movements between the limbs (Fig. 2.4). See [9] for a movie of the robot drumming.

On Fig. 2.3, it can be seen that the parameters are modulated in real time and that those modulations end up in a smooth adaptation of the generated trajectories. Moreover, the modifications occurs at specific times corresponding to the end of a beat thanks to the manager that deals with time constraints.

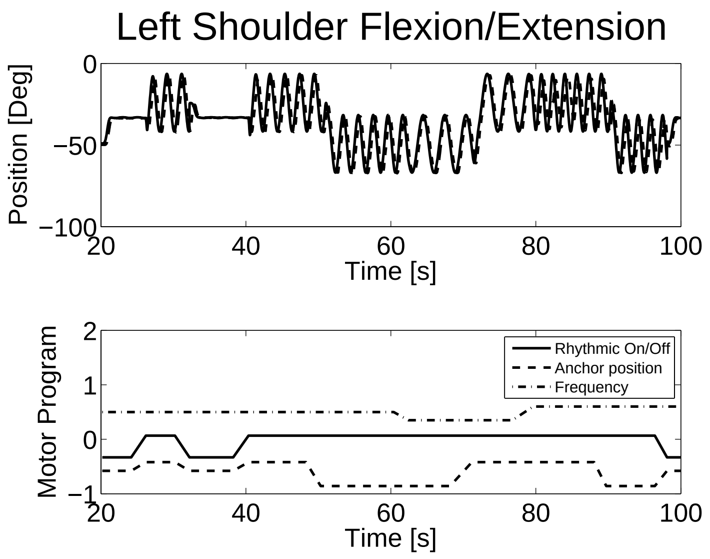


Figure 2.3: Up: Trajectories generated by the generator for one arm obtained with iCub when drumming. Plain lines are desired trajectories and dotted lines are the actual trajectories. Down: Corresponding parameters sent by the manager to the generator.

On Fig. 2.4, trajectories from the two legs are shown to illustrate coordination between limbs. It can be seen that the limb stay synchronized even when the frequency is changed (Fig. 2.4(a)). Moreover, when the coordination of the legs is changed, the transition is fast and the trajectories remain smooth (Fig. 2.4(b)).

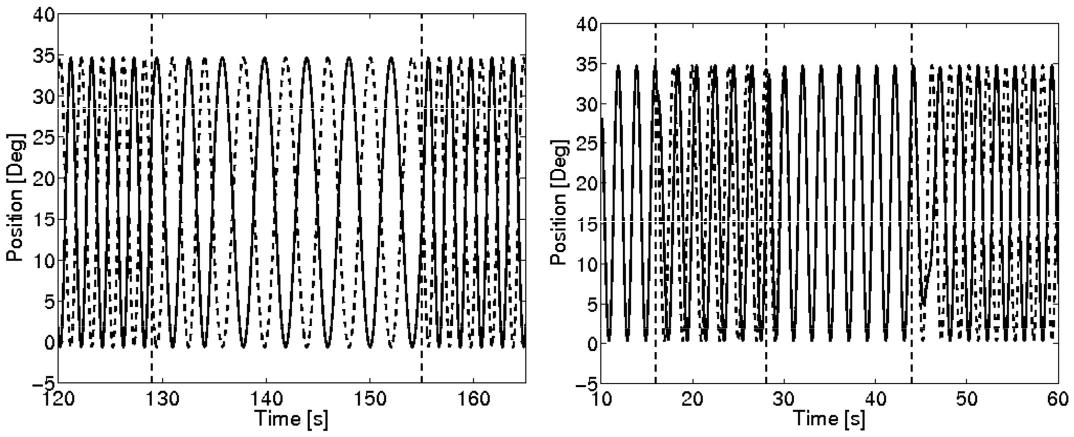
2.2 Crawling

In this application, we want to test the ability of the architecture to switch and combine behaviors. Contrarily to the drumming task, here behaviors are triggered by sensory information, i.e. no planner is involved. More precisely, we define three tasks (motor programs): *reaching*, *crawling* and *reaching while crawling*; each of this task is triggered by color marks on the ground, i.e. a red mark on the ground launches *reaching*, a blue mark *reaching while crawling* and no mark *crawling*. No visual processing is considered here; the position and color of the mark are directly provided to the robot. The robot crawls in an environment where it has to switch between those three behaviors according to marks arbitrarily placed on the ground. Combinations of *crawling* and *reaching* have been tested in simulation using the ODE-based software Webots™.

At the manager level, the onset of the different behaviors is controlled by two variables corresponding to the information relative to the mark and to the current state of the leg (in cases where the robot is crawling), i.e. a reaching movement starts only when the mark is reachable and when the robot is a position stable for such a movement (for instance the arm should be in the swing phase when crawling). An inverse kinematics (IK) algorithm is used to get the angles needed to reach marks on the ground; only three joints per arm are used to avoid redundancies.

At the generator level, we use the system defined in subsection by Eqs 1.2- 1.6 to generate the trajectories for each behavior. We define here the needed specific settings; each behavior is simply triggered through the specification of the amplitudes $\vec{m}$ and the offsets $\vec{g}$ by the manager.

Crawling ($\vec{m} > \vec{0}, \vec{g} = \vec{0}$). Analysis of crawling infants have shown that most infants crawl on



(a) Modulation of the frequency (b) Modulation of the coordination

Figure 2.4: (a) The left leg (plain line) and the right leg (dash line) are in anti-phase. This phase shift remains constant even when frequency of the system is modified (at 130s and 155s, vertical dash line). Moreover, the convergence to the new frequency in less than a cycle. (b) The left and the right legs (resp. plain and dash lines) are in phase at the beginning of the movement. Then at time 15s, 28s, and 44s (vertical dash lines) the phase shift of the right leg relatively to the clock is modified. The trajectory converges in less than a cycle to the desired one.

hands and knees, using a walking trot gait [1, 12]. The couplings are thus set so to obtain a trot gait and g is set to a fixed value (0 here) so to obtain a purely rhythmic movement.

Reaching ($\vec{m} < \vec{0}, \vec{g} \neq \vec{0}$). Once a mark is close enough to be reachable, *crawling* is turned off by setting $\vec{m}$ to a negative value at an appropriate time for each limb. Then the IK algorithm is used to get the target position $\vec{g}$ which is sent to the generator.

Reaching while crawling ($\vec{m} > \vec{0}, \vec{g} = \vec{g}(t)$). When a mark is reachable and when the arm is in an appropriate state (i.e. when the arm has not support the body, that is during the swing phase), the desired position g is sent to the generator by the manager; the actual position of the system is then compared to the desired position so to reach the correct position through a modification of the offset. $\vec{m}$ is kept to a positive value so that the resulting movement is rhythmic with a time varying offset.

Feedback. A phase dependent sensory feedback is included in the rhythmic PG to make the crawling locomotion more robust and adaptive to the environment, as we did previously in [10]. Information from the touch sensors located on the hands and knees of the robot is used to modulate the onset of the swing and stance phases, as mammals do [3]. The transition from stance to swing phases is delayed as long as the other limbs cannot support the body weight and is triggered sooner when the limb leaves unexpectedly the ground. Analogous policies are used for the swing to stance transition. More precisely, the term u_i of Eq 1.5 is defined as

$$u_i = \begin{cases} -\text{sign}(y_i)F & \text{fast} \quad \text{transitions} \\ -\omega x_i - \sum k_{ij} y_j & \text{stop} \quad \text{transition} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where F ($= 200$ in our case) controls the speed of the transition. Fig. 2.6 shows the activation of the feedback depending on the phase of the limb and the resulting modification of the phase space of the oscillator.

Crawling and reaching have been tested in simulation using the ODE-based software Webots^{TM}. We performed simulation of the three different behaviors with and without feedback. First we note that the feedback makes the crawling more stable and adaptive to the environment (i.e. the robot can

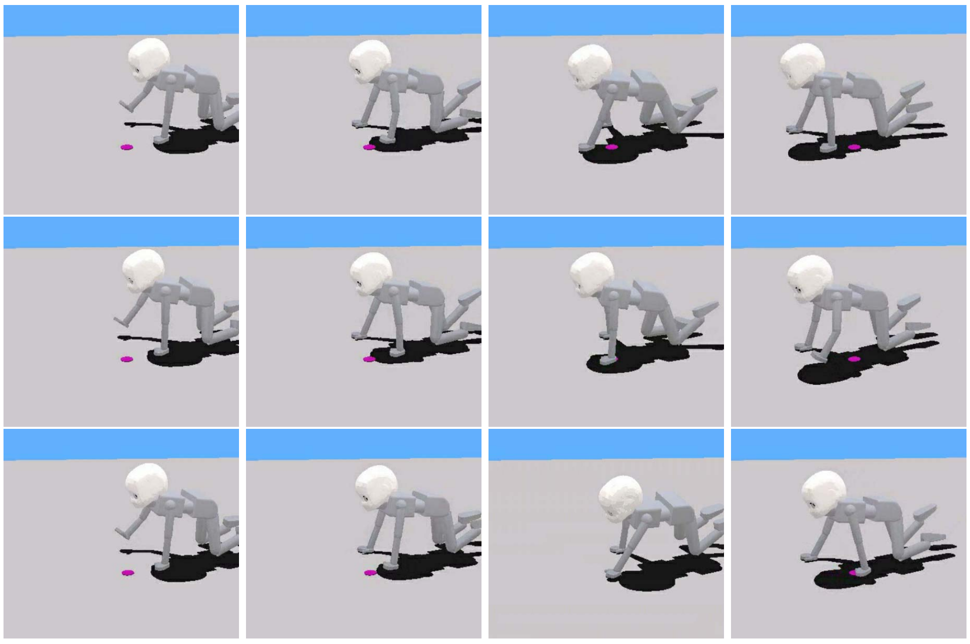
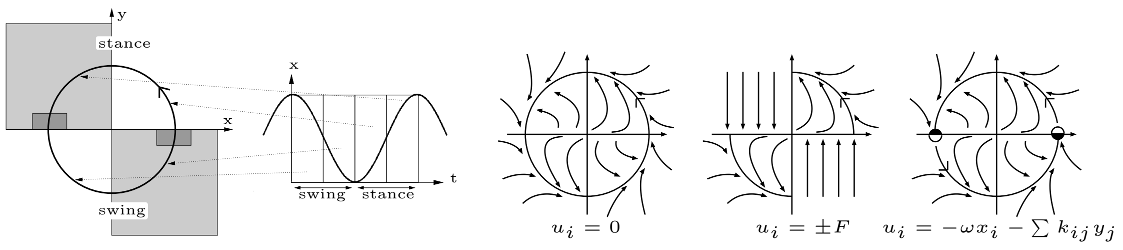


Figure 2.5: Snapshots of the different behaviors. Upper line: crawling only, middle line: reaching while crawling, lower line: crawling then reaching.

crawl on inclined slopes and uneven terrain), for details on the improvement of crawling by feedback refer to [10]. Second, thanks to our modular approach, the integration of the different behaviors and the feedback is successful (see Fig. 2.5 for snapshots of the three different behaviors).

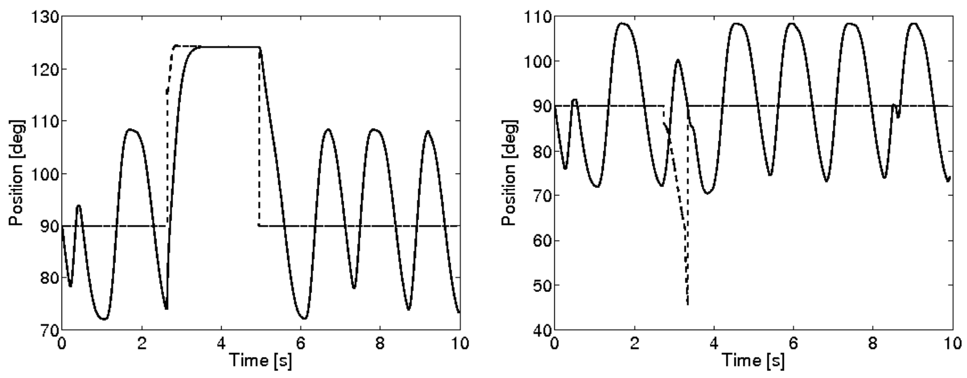
On Fig. 2.7, the trajectories of the left shoulder corresponding to the snapshots of Fig. 2.5 are shown together with the parameters sent by the manager. Both trajectories with and without feedback are presented. It can be seen that with feedback the trajectories are constantly modulated accordingly to the local sensory information showing fast adaptive behaviors. Specially at the beginning of the crawling, the amplitude of the trajectory is modulated to account for the initial posture of the robot which does not correspond to the usual crawling posture.

Concerning behaviors, we notice that simple parameters change allows to generate trajectories complex enough to fulfill the task. Thanks to the robustness of dynamical systems, the trajectories resume to simple crawling after reaching the mark in both tasks. The generated trajectories are smooth



(a) Feedback activation zone (b) Oscillator phase spaces

Figure 2.6: Phase space of an oscillator with its activation zone for the feedback (light gray for switch and dark gray for stop controls) and the correspondence with the x variable of the oscillator is shown on the left figure. Right figure shows the schematic phase plot of the oscillator for the different types of feedback.



(a) Reaching without crawling (b) Reaching while crawling

Figure 2.7: Results of the simulation for both reaching while crawling and reaching when crawling is stopped and resumed after. The experiments were done with and without feedback integration. The upper graphs show the left shoulder joint (plain line with feedback, dashed line without), the lower graphs show the corresponding parameters (m plain line with feedback, dashed line without. g dash-dot line with feedback, dotted line without). Refer to the text for discussion of the results.

despite the discontinuous nature of the parameters defining the GMP, thanks to the use of dynamical systems.

Results obtained in simulation have shown that the architecture allows for smooth transitions between motor behaviors; in both *reaching while crawling* and *reaching*, the trajectory smoothly resume to crawling after the mark has been reached. Fig. 2.5 shows some snapshots of the three tasks; for the corresponding movies, see [8].

Conclusion

We have presented here an architecture for the generation and the superimposition of discrete and rhythmic movements. This architecture is based on the concepts of motor programs and of central pattern generators and motor primitives.

Applications have shown that this architecture is well suited for both the specification on the fly of a behavior as well as the switch between behaviors accordingly to sensory information. Moreover, the architecture is suitable for the online generation of complex trajectories and can serve as the low level basis for any type of motor behaviors by adapting the planner layer.

We are planning to extend the architecture, notably by adding more feedback loops as vision, force sensing and sound. Moreover, the manager is going to be improved so to deal with constraints such as equilibrium and self collision avoidance. We are also working on making the architecture general enough to be usable by the different partners of the consortium. In the future, the generator module is going to be implemented at the DSP level to be able to have faster control loop.

Bibliography

[1] K.E. Adolph, B. Vereijken, and A.B. Denny. Learning to crawl. *Child Development*, 69:1299–1312, 1998.

[2] S. Degallier, C. P. Santos, L. Righetti, and A. Ijspeert. Movement generation using dynamical systems: a humanoid robot performing a drumming task. In *IEEE-RAS Inter. Conf. on Humanoid Robots*, pages 512–517, 2006.

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

[4] M. Golubitsky, I. Stewart, and A. Torok. Patterns of synchrony in coupled cell networks with multiple arrows. *SIAM J. Appl. Dynam. Sys.*, 4(1):78–100, 2005.

[5] S. Grillner. Neurobiological bases of rhythmic motor acts in vertebrates. *Science*, 228(4696):143–149, 1985.

[6] E. R. Kandel, J.H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. Mc Graw Hill, 2000.

[7] C.D. Marsden, P.A. Merton, and H. Morton. The use of peripheral feedback in the control of movements. *Trends Nerosci.*, 7:253–258, 1984.

[8] Movie of Crawling. http://birg2.epfl.ch/users/degallier/movies_BioRob/crawl.mpeg.

[9] Movie of Drumming. http://birg2.epfl.ch/users/degallier/movies_BioRob/icubdrum.mpg.

[10] L. Righetti and A.J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation ICRA*. Accepted.

[11] L. Righetti and A.J. Ijspeert. Design methodologies for central pattern generators: an application to crawling humanoids. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[12] L. Righetti, A. Nylén, K. Rosander, and A.J. Ijspeert. Kinematics of crawling in infants. In *preparation*.

[13] R.A. Schmidt and T.D. Lee. *Motor control and learning: A behavioral emphasis*. Human Kinetics, Champaign, IL, USA, 2005.

[14] M.C. Tresch, Ph. Saltiel, and E. Bizzi. The construction of movement by the spinal cord. *Nature Neuroscience*, 2:162–167, 1999.

[15] D. Vernon, G. Metta, and G. Sandini. The icub cognitive architecture: Interactive development in a humanoid robot. *IEEE International Conference on Development and Learning*, 2007.