# Developing Central Pattern Generator based periodic motions using tactile interaction

Fabio DallaLibera*, Takashi Minato†, Hiroshi Ishiguro†‡, Emanuele Menegatti*
* Department of Information Engineering (DEI),
University of Padua, Via Gradenigo 6/a, I-35131 Padova, Italy
† ERATO, Japan Science and Technology Agency,
Osaka University, Suita, Osaka, 565-0871, Japan
‡ Department of Adaptive Machine Systems,
Osaka University, Suita, Osaka, 565-0871 Japan

***Abstract*— Controlling robots by Central Pattern Generators (CPGs) is a widespread bio-inspired technique for the realization of robot motions. The numerous parameters of the CPGs, often specialized for a specific task, are usually set by automatic techniques like genetic algorithms or policy gradient. However, using these approaches leaves the users with little control on the resulting motion, which can be modified only by changing the evaluation function. Conversely manually setting each parameter gives the user full control over the motion, but identifying which parameters should be altered to obtain a desired effect is not intuitive and therefore developing motion requires time and effort. We present a system that allows programming the CPG parameters by interacting with the robot and in particular by intuitively touching the robot, giving the user full control on the resulting motion without requiring a direct modification of the parameter values.**

## I. Introduction

Many of the periodic movements of animals, like swimming, walking or chewing, are controlled by groups of neurons called Central Pattern Generators (CPGs) that can produce a rhythmic activity even in the absence of external inputs. When motor and sensory feedback is included the resulting entrainment between the robot and the environment brings several advantages in terms of stability and ability to adapt to environmental changes [1]. To achieve the same desirable characteristics of simplicity and robustness CPG had been proposed for the control of periodic motions of many kinds of robots, for instance hexapods [2], quadrupeds [3], [4], [5], bipeds [6], [7], [8], [9], snake robots [10], etc.

Often artificial CPGs consist of weakly coupled oscillators, where each oscillator consists of a couple of neurons representing extensor and flexor neurons [11]. As reported in [12] the determination of the numerous CPG parameters is still difficult because they depend both on the robot and on the environment. Usually CPGs parameters are set automatically by genetic algorithms [11], policy gradient [13], reinforcement learning [12], or similar techniques. Although there are several advantages, this gives the user little control over the resulting motions, in fact the only strategy available for controlling the resultant movement is changing the fitness/evaluation/reward function. Some features like the similarity to human movement cannot be easily expressed mathematically and therefore, for instance, when developing a walk motion for humanoid robots if the speed is used as a reward it is difficult to prevent the resultant motion from being fast and awkward. Conversely manually setting each of the parameters, for instance from a console or a slider-based graphical user interface, allows the user to finely control the motion. This is, however, time-consuming, and, especially in the case of oscillators which are not easy to predict, like Matsuoka's neuron [14], it can be very unintuitive and difficult.

In this paper we propose to use tactile interaction to set the CPG parameters. In particular we imagine the user to watch the movement of a robot equipped with touch sensors, touch the robot to modify the parameters, observe the change and repeat the process until the desired robot movement is satisfactory. More concretely, we assume each face of the robot links to be covered by a binary touch sensor. The touch pattern applied by the human operator over time on these sensors is used to modify the CPG parameters.

To allow this interactive process we need a highly predictable network of oscillators. In fact, if similar touches could lead to quite different consequences we can imagine that the usability of the system would be strongly negatively affected. Section II therefore presents the highly predictable oscillator network we employed. A strategy to convert user touches to CPG-parameter modifications must then be defined. In section III we illustrate our proposal, based on the assumption that to have an intuitive protocol the users should touch the part whose movement they want to modify.

We report experimental data relative to the development of a walking and a side stepping motion for a humanoid robot in section IV and conclude in section V presenting future works.

## II. CPG Network

As stated in the introduction we need the CPG network to be easily predictable, so that we can map touches to desired effects, and in turn identify the parameter change needed to obtain the desired effect.

Many alternatives are available for the choice of oscillator/neurons employed for the realization of the CPG, for instance sinusoidal oscillator [7], Hopf or adaptive frequency

Hopf oscillator [8], Rayleigh oscillator, Van Der Pol oscillators [15], FitzHugh-Nagumo oscillator [4], Hopfield and Hopfield with synaptic depression [16] or Matsuoka's [14].

For its simplicity, robustness and predictability we decided to employ the Hopf oscillator, which essentially provides a sinusoidal signal.

The predictability of the system is then strongly influenced by the oscillator interconnections. Among all the possible network interconnections, we can notice that essentially five structures are present in the literature:

1) chain [10], used mainly for snake robots
2) star [17], that is a "peacemaker"/ "clock" oscillator provides a synchronizing signal to all the others
3) tree [18], where essentially the oscillators are connected as a tree, from the proximal to the distal joints
4) connection between homologous joints [15], [3], i.e. joints with a similar function
5) full connection between the oscillators [5]

We decided to employ a star structure. This configuration is very general and task independent since no assumptions are made about which joints should be synchronized, a feature required by our system. Star structure is also highly predictable, since the peacemaker oscillator gives a synchronizing signal to all the oscillators controlling the joints but no unexpected effects occur due to the interaction between groups of oscillators as in the case of connection between homologous joints. Precisely, our implementation uses a directional coupling from the clock oscillator to the others. This assures the maximum system predictability. Tests on whether a bidirectional coupling can improve entrainment with the environment without affecting the ability to foresee the system behavior will be performed in future works.

More formally in our implementation each of the $n$ degrees of freedom of the robot is controlled by one oscillator and a further "clock" oscillator provides reference signals for these oscillators. Let us identify by $C_0$ the reference oscillator and by $C_j$, $1 \leq j \leq n$ the oscillators controlling the robot joints,

Using the complex number representation for the Hopf oscillator [19] we have for the $j$-th oscillator, $0 \leq j \leq n$

$$\begin{aligned} \dot{z_j} &= \gamma\left(\mu_j - |z_j|^2\right) z_j + i\omega_j z_j + F_j(t) \\ m_j &= \Re\{z_j\} + o_j \end{aligned} \tag{1}$$

In detail

- $z_j \in \mathbb{C}$ is the state of the oscillator
- $m_j \in \mathbb{R}$ is the control signal for the actuator
- $\gamma$ is a coefficient for the speed of recovery after perturbation [8]
- $\mu_j \in \mathbb{R}, \mu_j \geq 0$ controls the amplitude of the oscillation
- $\omega_j \in \mathbb{R}, \omega_j \geq 0$ controls the oscillation frequency
- $F_j(t)$ is an external perturbation signal
- $o_j$ is an offset value used to set the position around which the joint oscillates

Since we decided to restrict to periodic motions to ensure rational ratios between the frequencies of oscillation of each pair of joints we set

$$\omega_j = p_j \omega_0 \tag{2}$$

$1 \leq j \leq n$, $p_j \in \mathbb{N}$. In the current implementation no feedback signal is introduced, so $F_0(t)$ is zero (the main clock is not influenced by the external world) while for $1 \leq j \leq n$

$$F_j(t) = we^{i\phi_j} z_0^{p_j} \tag{3}$$

that is $F_j(t)$ consists essentially of the perturbation from the clock oscillator that permits a synchronization of the whole system. The reference signal $z_0$ is elevated to the power $p_j$ so the frequencies of the oscillator and of the perturbation are close. The similarity of the frequencies leads to an easier synchronization and a predictable phase between the $j$-th oscillator and the reference one [20]. The coefficient $w$ determines the coupling strength between $C_0$ and the other oscillators while the term $e^{i\phi_j}$ changes the phase difference between the clock oscillators and the other ones. In the current implementation $\mu_0 = 1$, $w = 0.1$ and $\gamma = 1000$.

## III. Touch protocol

As stated in the introduction we based our protocol on the assumption that touching the part whose movement is to be modified should be intuitive for human operators. From the previous section users can easily imagine that for each oscillator we can alter its amplitude (parameter $\mu_j$), its frequency (parameter $p_j$), its phase with respect to the reference oscillator (parameter $\phi_j$) and the offset position around which the joint moves (parameter $o_j$). In addition it is possible to use $\omega_0$ to change the global speed of the movement. Depending on the pressed touch sensors we decide which is the oscillator whose parameters should be modified. This is done by selecting the most distal joint that causes a movement of the pressed sensor in the direction normal to the sensor surface. We then decide which feature of the movement (amplitude, frequency, phase or offset) is affected depending on the pressure pattern. Expressly, if the user keeps pushing the sensor for a very long time the offset is modified. If the user pushes for a shorter time the amplitude is modified, while a single tap is used to change the phase. Two consecutive taps are used to change the frequency, and in particular if the touch is applied to the main body the reference oscillator frequency $\omega_0$ is changed.

More formally suppose the robot's main body (in the case of a humanoid robot, the torso) is fixed in space, and the vector denoted by $d_j$ represents the derivative of the position in the space of the center of the pushed sensor when the $j-th$ joint is rotated. Identify $n_s$ as the vector perpendicular to the pushed sensor surface. Denote by $j_1, j_2, ...j_q$ the indices of the $q$ joints that connect the robot's main body to the link where the sensor is located, in order from the most distal to the most proximal (see Fig. 1); we identify the joint $j_s$ such that $\rho_s =< n_s, d_{j_s} >\neq 0$ and $< n_s, d_{j_k} >= 0$ for $s < k \leq q$. If this joint doesn't exist we simply ignore the sensor pressure (unless it is on the main body, as will be specified later).

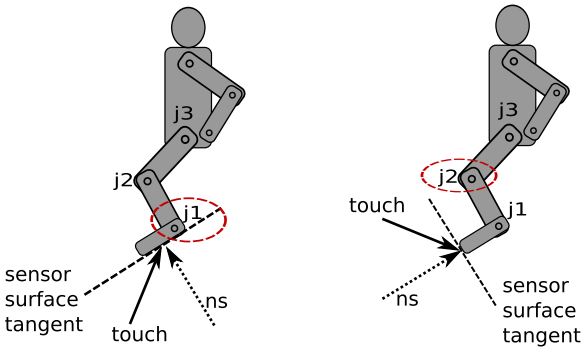



Fig. 1. Example of the determination of the joint whose parameters should be modified when a sensor is pressed. In the first case the parameters of joint $j_1$ are modified, in the second case, since the $\rho_1 =< n_s, d_{j_1} >$ is zero $j_2$ is selected.

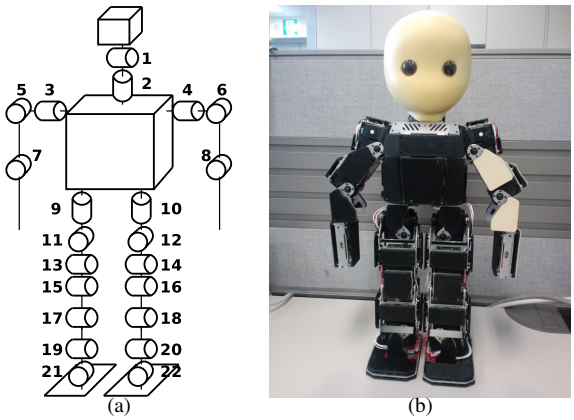



Fig. 2. Schema of the joints of the robot with their IDs (a) and photo of Vision 4G (b).

Once $j_s$ is determined the phase of joint $j_s$ ($\angle z_{j_s}$) is used as a time reference. Expressly the pushing time $\tau_{j_s}$ is measured in terms of phase difference between the release time and the pushing time, counting for the phase resettings (i.e. the phase difference is considered non-negative, monotonically increasing and can be larger than $2\pi$). We distinguish the following cases:

- If $\tau_{j_s} > \Theta_O$ (the user pushes for a very long time) the offset parameter is changed according to the direction of the applied force, i.e. $o_{j_s,new} = o_{j_s,old} + sgn(\rho_{j_s})\Delta_O$, where $sgn$ is the sign function.
- If $\Theta_A < \tau_{j_s} \leq \Theta_O$ the amplitude parameter is updated by the value $sgn((\rho_{j_s} * m_{j_s})\Delta_A$ where $m_{j_s}$ is the value of the output at the pushing time.
- If the user pushes for a time $\tau_{j_s} \leq \Theta_A$, releases the sensor and doesn't push it for a time $\Theta_P$ then the phase parameter $\phi_{j_s}$ is updated such that in the following cycles the closest maximum of oscillation occurs at the pushing time, i.e. the quantity $-\angle(m_{j_s} * z_{j_s})$ is added to $\phi_{j_s}$, where $z_{j_s}$ and $m_{j_s}$ are considered at the pushing time.
- If the user pushes for a time $\tau_{j_s} \leq \Theta_A$, releases the sensor and before a phase change of $\Theta_P$ pushes the sensor again then $p_{j_s}$ is incremented or decremented respectively if this second pushing time $\tau_{j_s,2}$ is greater or lower than $\Theta_A$.
- Similarly, if the user pushes the robot main body for a time $\Delta\phi_0 \leq \Theta_A$ releases the sensor and before a phase change of $\Theta_P$ pushes the sensor again then $\omega_0$ is increased or decreased by the quantity $\Delta_{\omega_0}$ respectively if the second pushing time is greater or lower than $\Theta_A$.

In our implementation all the $\Delta$ and $\Theta$ values are constants, expressly $\Theta_O = \pi$, $\Theta_A = \frac{\pi}{6}$, $\Theta_P = \frac{2\pi}{3}$, $\Delta_O = \Delta_A = \frac{\pi}{12}$, $\Delta_{\omega_0} = 1$.

Given these definitions, we can expect the protocol we defined to be more intuitive than a simple manual parameter tuning. For instance, if users wants to change the center of oscillation of some joints they have to consider the position of the various motors in the kinematic chain, and decide which motor offsets should be modified. Then, depending on how each motor in mounted on the robot and on the established conventions, they need to evaluate whether the offset parameters of each joint should be increased or decreased. Conversely, with our approach the users are just required to keep pushing the robot parts in the direction they want the centers of oscillation to be moved.

## IV. Simulations

In order to validate the feasibility of the experiment we developed a crawling movement, a walking gait and a sidestep motion. To apply the touch protocol defined beforehand we need to distinguish between sensor pressure due to the gravity and pushes applied by the user. This module is currently not available in our system, so we decided to employ a simulator as a proof of concept. The user is able to interact with a 3D rendering of the robot (see figures 5, 5 and 6) where each link face simulates a touch sensor that can be clicked with the mouse (details of the simulator and advantages in using simulated touch sensors are reported in [21]). The simulator is based on ODE and models a customized version of Vision4G[1]. The robot photo and its joint schema are available in Fig. 2.

Although the results here presented as a proof of concept were obtained using a simulated robot, we aim our technique to be applicable to real robots. Since some CPG parameter configurations can lead to robot self-collisions that can damage the actuators, these collisions must be predicted online. The collision detection should therefore be very fast, but it can however be inaccurate. More precisely, while we should assure collision positions to be avoided, we are not required to recognize each collision free posture as such, i.e., we can reduce the set of allowed postures to increase the computation speed. In our implementation each link is approximated by a set of slightly bigger parallelepipeds, and the computation is limited to checking the collisions between those parallelepipeds.

[1]See `http://www.vstone.co.jp` and `http://www.ode.org` for Vision4G and ODE, respectively.

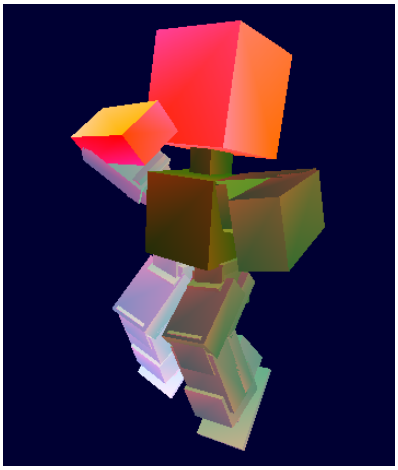


Fig. 3. Output of the collision prevention system. The figure shows the motion that would be generated if the collision prevention system were not enabled and highlights the colliding parts by a red color.

The simulator provides a graphical representation of the motion generated by the CPG and highlights the parts that would collide without the intervention of the prevention collision system, as depicted in Fig. 3. This information can be exploited by the user to refine the robot movement and get a motion that does not require any collision prevention.

Utilizing our system a crawling motion was developed by a single user in 56 minutes by 57 amplitude changes, 39 phase changes, 22 offset changes and 2 frequency changes. The same user then realized a side-step movement in 29 minutes, using 31 amplitude changes, 4 frequency changes, 18 phase changes and 56 offset changes. Finally an open-loop walking motion was obtained in 34 minutes. This required 60 amplitude changes, 15 frequency changes, 28 phase changes and 132 offset changes. Though a single experiment with a single user cannot provide any statistical evidence, we can notice that the number of commands per minute highly increased during the motion development. In detail in the first experiment 2.14 commands/minute were provided, in the second the frequency raised to 3.7 commands/minute and by the third motion the number of command/minute reached 6.9. This suggests that human operators can easily adapt to the system. We can in fact exclude learning of the robot dynamics, since the same simulator and robot models were previously used by the user in plenty of experiments.

Table I reports the final values obtained for the CPG parameters for the three motions. The final value for $\omega_0$ is 3.256 for the crawling, 5.256 for the side step and 1.256 for the walking motion. Figures 4, 5 and 6 report screenshots of the developed motions. Videos are available at `http://robotics.dei.unipd.it/~fabiodl/papers/material/humanoids09/`. A further tuning, for instance to assure perfect movement symmetry or to maximize the crawling or walking speed could be performed but this is outside the scope of this paper. We would also like to stress that the purpose of the experiment is not to achieve the fastest locomotion speed which is possible but to validate the feasibility of the approach by developing a motion that is satisfactory for the user (in terms, for instance, of similarity to human movements). We therefore didn't even measure the walking velocity and ignored speed comparisons with walking gaits obtained with other methodologies. For comparison, however, Fig. 7 reports a crawling movement obtained with a genetic algorithm (population size 20, 60 generations, real value encoding, roulette wheel selection, mutation probability 1). The average speed over one minute was employed as evaluation function. We can see that the algorithm finds a shape for the legs that minimizes the friction with the ground, and uses the head as a support point to proceed forward by large arm movements. While this solution can lead to a good speed, it definitely looks awkward to humans. A difference in the smoothness of the two motions can be deduced quantitatively by observing the roll and pitch in the two cases (see Fig 8). The ranges of variation for the roll and the pitch of the robot are 15.4 and 14.5 degrees respectively for the motion obtained by direct interaction with the CPG and 34.7 and 23.9 for the motion optimized by the genetic algorithm. Obviously the pitch could be improved by introducing another term into the evaluation function that penalizes the high pitch and roll excursions, although deciding the weighting between the two terms is far from being a trivial task. Furthermore we can imagine that users desire symmetries in the motion of the limbs. This would require another evaluation term, whose weight would be again difficult to guess. The difficulty in choosing the evaluation function weights highlights the richness and complexity of the evaluation that human user can provide: many factors are intuitively weighted, with priorities that are difficult to express as a weighted combination of evaluation functions. Indeed, developing systems able to directly exploit the human user evaluation appears very appealing, and the proof of concept results presented in this section seems to suggest that CPG parameter tuning by touch could represent a way to intuitively realize motions using a very simple and robust systems.

## V. Conclusions and Future Works

In this paper we proposed the idea of using touch to set the parameters of a CPG. More precisely we devised a protocol to map user touches to desired motion modifications and we designed a CPG network that allows to predict which parameter change performs the desired movement alteration. Since online modification of the CPG parameters can easily cause self collisions we then shortly described the simple and fast self collision prevention system that was employed in the realization of the proposed system.

We reported data relative to experiments conducted with a simulated humanoid robot. Three different motions, namely a crawling motion, a walking gait, and a side step motion were developed using the proposed approach, which revealed to be feasible for the creation of motions from scratch. Further analysis with a high number of unexperienced users

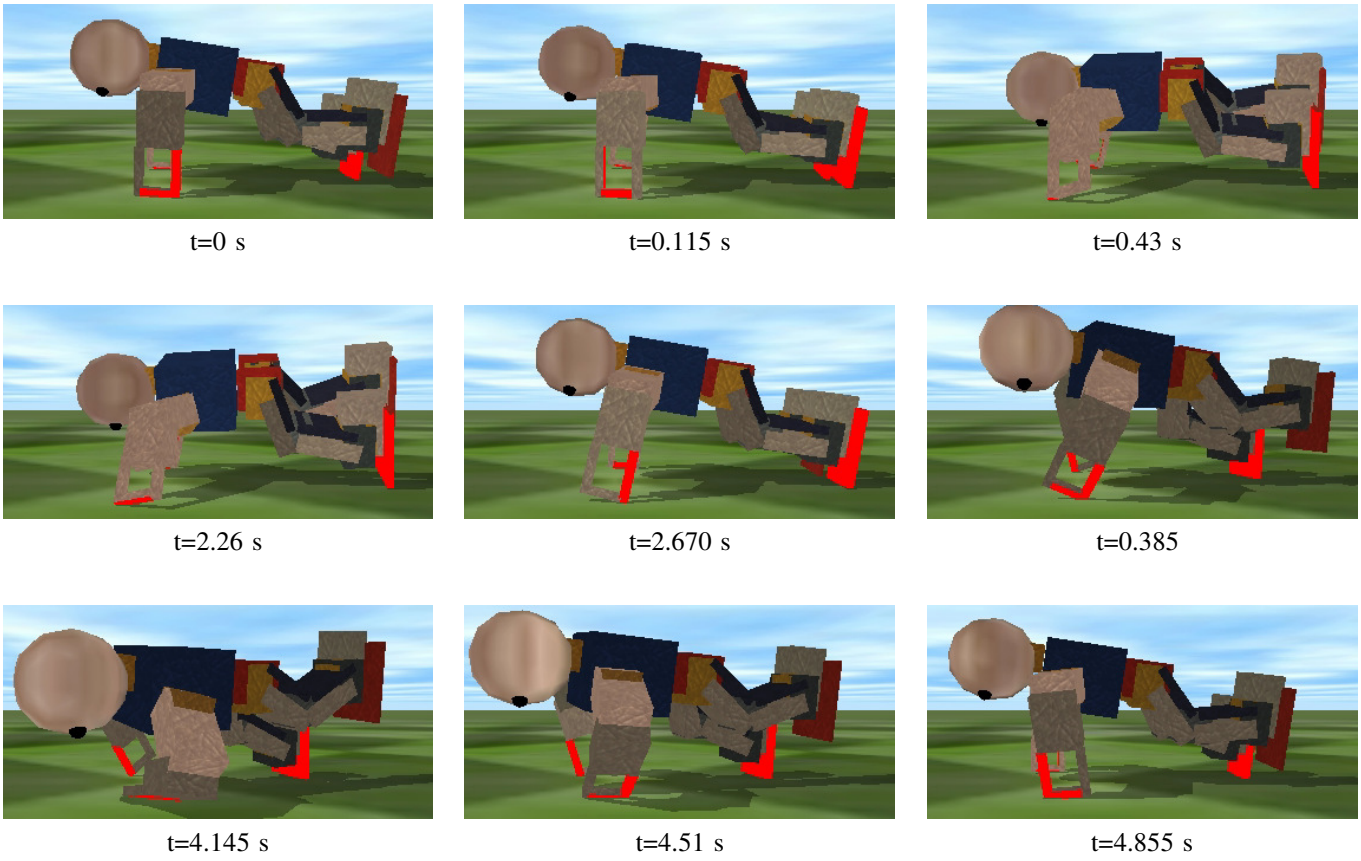



Fig. 4. Execution of the crawling movement.

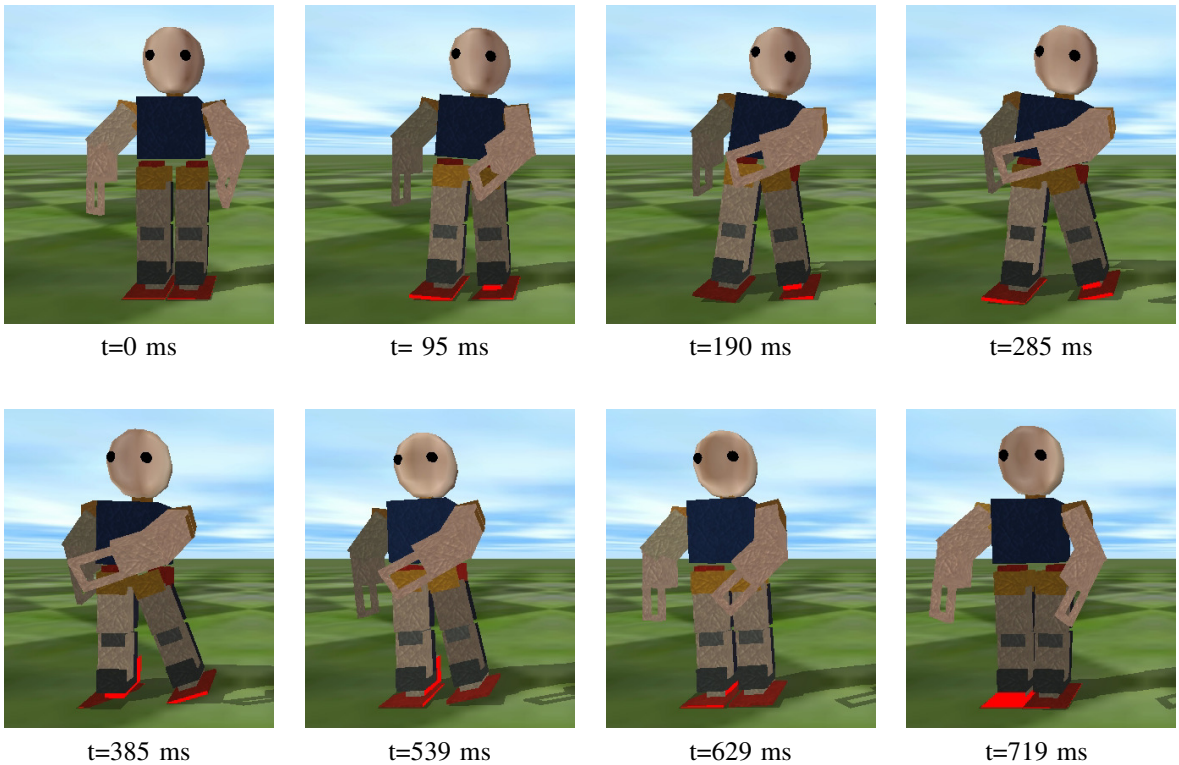



Fig. 5. Execution of the sidestepping movement.

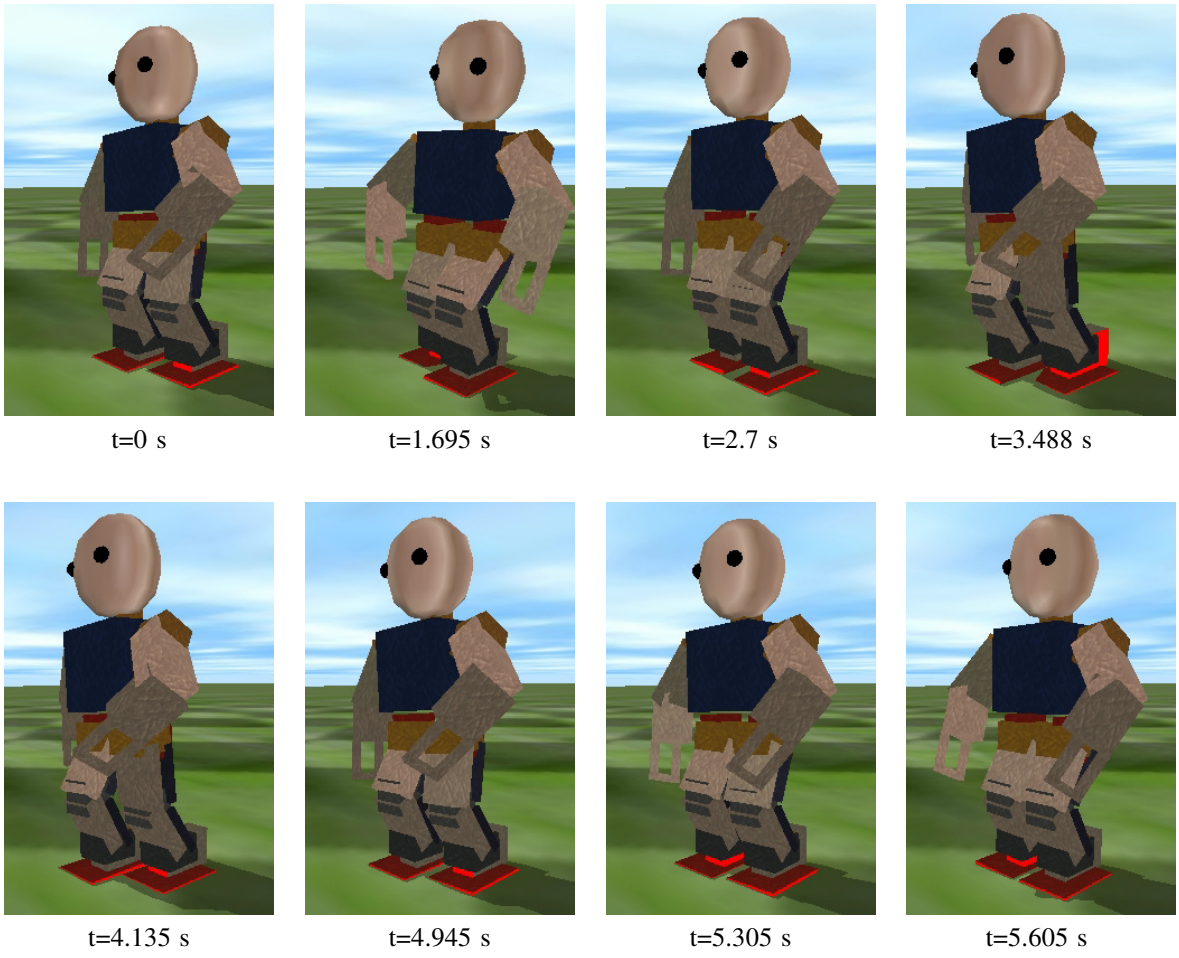



Fig. 6. Execution of the walking movement.

TABLE I

CPG PARAMETER SETTINGS OBTAINED FOR THE THREE MOTIONS.

| Parameters / Joint | Crawling | | | | Walking | | | | Side Step | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mu_j$ | $p_j$ | $\phi_j$ | $o_j$ | $\mu_j$ | $p_j$ | $\phi_j$ | $o_j$ | $\mu_j$ | $p_j$ | $\phi_j$ | $o_j$ |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0.52 | 1 | 0.66 | 0.78 | 0.17 | 1 | 0.4 | -0.7 | 0.35 | 1 | 0 | -0.7 |
| 4 | 0.26 | 1 | 1.61 | -0.52 | 0 | 1 | 0 | 0.35 | 0.17 | 1 | 1.86 | 0.17 |
| 5 | 0 | 1 | 9.72 | 0.52 | 0 | 1 | 0 | 1.22 | 0 | 1 | 0 | 1.05 |
| 6 | 0.52 | 1 | 2.31 | 0 | 0 | 1 | 0 | -1.05 | 0.35 | 1 | 4.4 | -1.4 |
| 7 | 0.26 | 1 | 2.20 | 0.78 | 0.17 | 1 | 0.49 | -0.17 | 0 | 1 | 0 | 0 |
| 8 | 0.78 | 1 | 5.69 | -0.52 | 0.17 | 1 | 0 | -0.35 | 0 | 1 | 5.5 | 0 |
| 9 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 11 | 0 | 1 | 5.68 | 0 | 0 | 1 | 1.64 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0.17 | 1 | 0 | 0 |
| 13 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 14 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 15 | 0.26 | 1 | 1.28 | -1.05 | 0 | 1 | 0 | -0.52 | 0 | 1 | 0 | 0 |
| 16 | 0.26 | 1 | 0 | 1.05 | 0.17 | 1 | 0.31 | 0.35 | 0 | 1 | 0 | 0 |
| 17 | 0.26 | 1 | 0 | 0 | 0 | 1 | 0 | 0.52 | 0 | 1 | 0 | 0 |
| 18 | 0.26 | 1 | 0 | 0 | 0 | 1 | 0 | -0.52 | 0 | 1 | 0 | 0 |
| 19 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 20 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 21 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 22 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

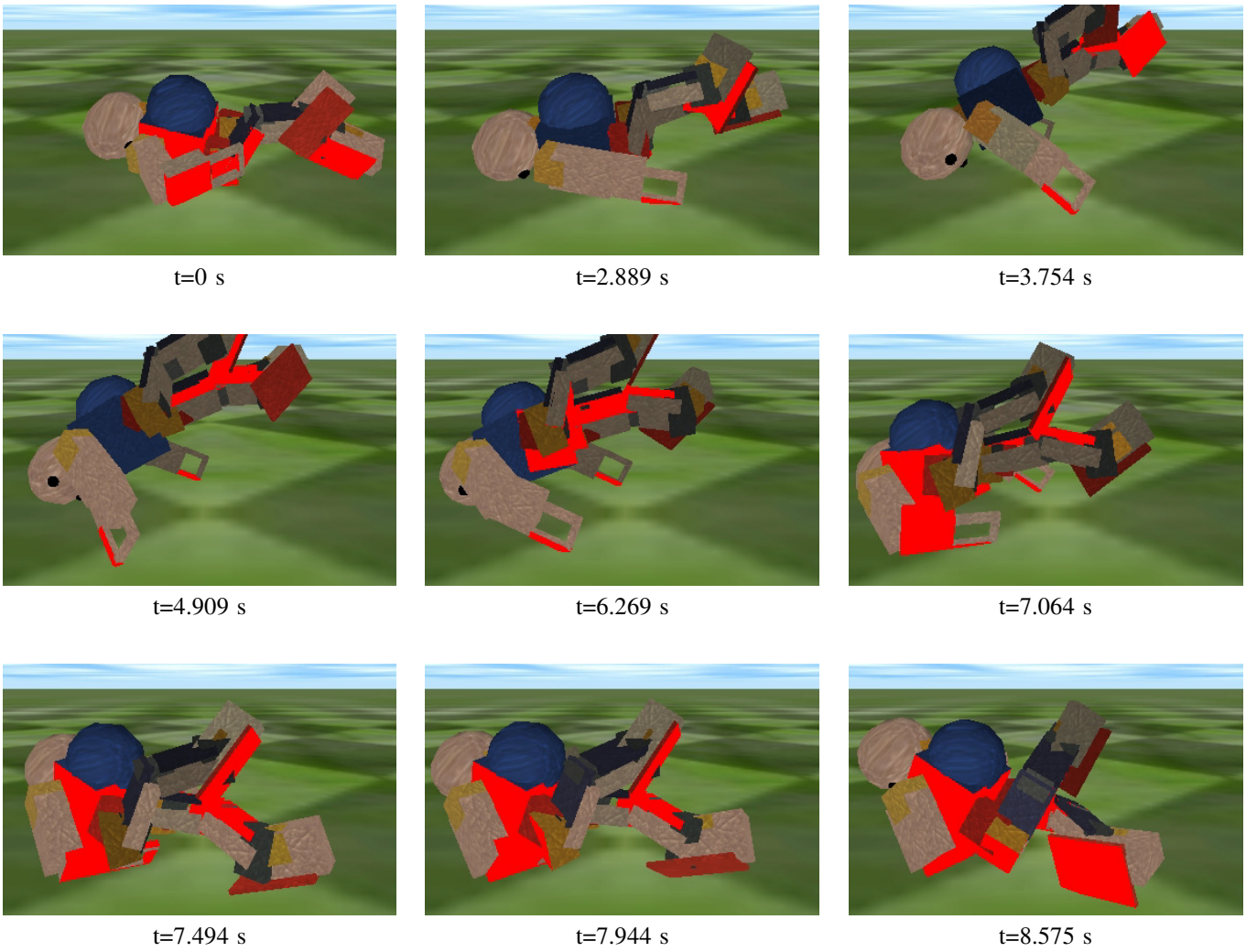



Fig. 7. Execution of a crawling movement obtained with a genetic algorithm.

should be performed to provide statistical evidence of the the advantages of the proposed system over classical approaches.

Future works will involve the implementation of the proposed system on a real robot. We can expect the approach to be easily applicable for the development of slow motions, which allow the user to directly interact with the real hardware. For fast motions, we can simply stop the robot motion execution, touch the robot to alter the movement and restart the motion execution. The user would then be required to wait few cycles to have the CPG stabilize again, observe the new motion and if necessary apply further modifications.

When applying our technique to the real hardware some aspects need however to be tackled. First of all, discrimination between user touches, self touches, and pressures due the environment must be performed to avoid unwanted modification of the motion. Furthermore, the actual implementation does not include feedback from the sensors or the gyroscope. Since we can expect entrainment with the environment and therefore a better stability and motion variation, future research must be aimed at including feedback without reducing the generality of the approach. Finally, since the trajectories generated by Hopf oscillators are essentially just sinusoidal other types of easily predictable oscillators will be considered to broaden the possible movement repertories.

## VI. Acknowledgements

The authors would like to thank Derek Tia for the comments on the first draft of the paper.

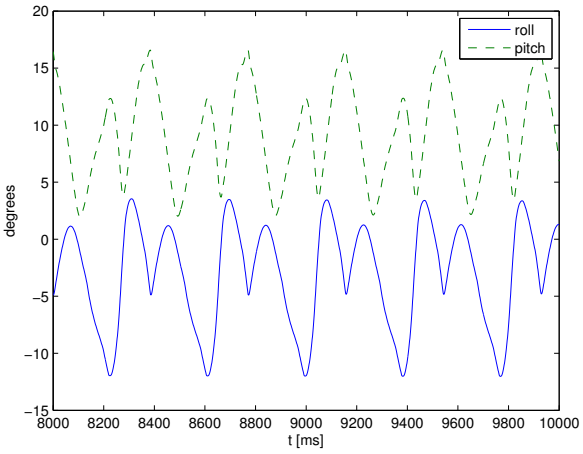
## References

[1] K. Doya and S. Yoshizawa, "Adaptive synchronization of neural and physical oscillators," vol. 4, pp. 109–116, 1992.

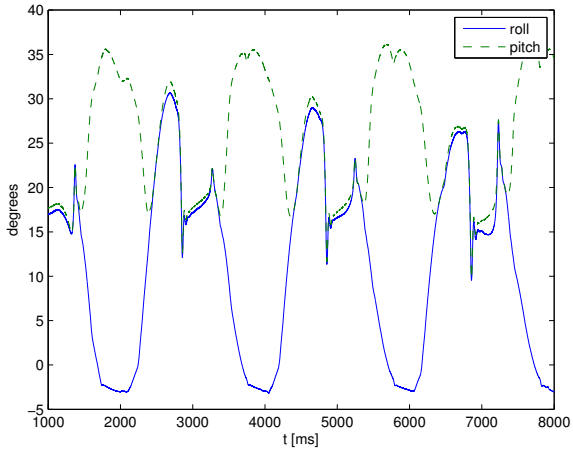
[2] M. A. Lewis, A. H. Fagg, and G. Bekey, "Genetic algorithms for gait synthesis in a hexapod robot," *Zheng, ed. Recent Trends in Mobile Robots*, pp. 317–331, 1994.

[3] H. Kimura, Y. Fukuoka, Y. Hada, and K. Takase, "Three-dimensional adaptive dynamic walking of a quadruped - rolling motion feedback to cpgs controlling pitching motion," in *2002 IEEE Intl. Conf. on Robotics and Automation (ICRA 2002)*, Washington, USA, 2002, pp. 2228–2233.

[4] J. J. Collins and S. A. Richmon, "Hard-wired central pattern generators for quadrupedal locomotion," *Biological Cybernetics*, vol. 71, no. 5, 1994.




(a)




(b)

Fig. 8. Pitch and roll for the crawling motion obtained with the proposed approach (a) and for the crawling motion obtained by a genetic algorithm (b).


[5] K.Tsujita, K.Tsuchiya, and A.Onat, "Decentralized autonomous control of a quadruped locomotion robot," *Artificial life and robotics*, vol. 5, pp. 152–158, 2003.

[6] G. Taga and H. Yamaguchi, Y. Shimizu, "Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment," *Biological Cybernetics*, vol. 65, pp. 147–159, 1991.

[7] J. Morimoto, G. Endo, J. Nakanishi, S.-H. Hyon, G. Cheng, D. C. Bentivegna, and C. G. Atkeson, "Modulation of simple sinusoidal patterns by a coupled oscillator model for biped walking," in *2006 IEEE Intl. Conf. on Robotics and Automation (ICRA 2006)*, Orlando, USA, 2006, pp. 1579–1584.

[8] L. Righetti and A. Ijspeert, "Programmable central pattern generators: an application to biped locomotion control," in *2006 IEEE Intl. Conf. on Robotics and Automation (ICRA 2006)*, Orlando, USA, 2006, pp. 1585–1590.

[9] A. C. de Pina Filho, M. S. Dutra, and L. S. C. Raptopoulos, "Modeling of a bipedal robot using mutually coupled rayleigh oscillators," *Biological Cybernetics*, vol. 92, no. 1, pp. 1–7, 2005.

[10] A. Crespi, A. Badertscher, A. Guignard, and A. Ijspeert, "Amphibot i: An amphibious snake-like robot," *Robotics and Autonomous Systems*, vol. 50, no. 4, pp. 163–175, 2005.

[11] H. Inada and K. Ishii, "Behavior generation of bipedal robot using central pattern generator(cpg) (1st report: Cpg parameters searching method by genetic algorithm)," in *2003 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems(IROS 2003)*, vol. 3, Las Vegas, USA, 2003, pp. 2179–2184.

[12] Y. Nakamura, T. Mori, M. aki Sato, and S. Ishii, "Reinforcement learning for a biped robot based on a cpg-actor-critic method," *Neural Networks*, vol. 20, no. 6, 2007.

[13] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot," *International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[14] K. Matsuoka, "Sustained oscillations generated by mutually inhibiting neurons with adaptation," *Biological Cybernetics*, vol. 52, p. 367376, 1985.

[15] P. Roy and Y. Demiris, "Analysis of biped gait patterns generated by van der pol and rayleigh oscillators under feedback," in *Proc. 3rd Intl. Symp on Adaptive Motion in Animals and Machines (AMAM 2005)*, Ilmenau, Germany, 2005.

[16] A. L. Taylor, G. W. Cottrell, and W. B. Kristan, "Analysis of oscillations in a reciprocally inhibitory network with synaptic depression," *Neural Computation*, vol. 14, pp. 561–581, 2002.

[17] S. Schaal, S. Kotosaka, and D. Sternad, "Nonlinear dynamical systems as movement primitives," in *IEEE-RAS 1st Intl. Conf. on Humanoid Robots (Humanoids 2000)*, Cambridge, USA, 2000, pp. 117–124.

[18] L. Jalics, H. Hemami, and Y. F. Zheng, "Pattern generation using coupled oscillators for robotic and biorobotic adaptive periodic movement," in *1997 IEEE Intl. Conf. on Robotics and Automation (ICRA 1997)*, vol. 1, Albuquerque, USA, 1997, pp. 179–184.

[19] A. Kern and R. Stoop, "Nonlinear dynamics of cochlear information processing," *Proc. Int. Workshop of Nonlinear Dynamics of Electronic Systems (NDES-04), Evora, Portugal*, pp. 190–193, 2004.

[20] K. J. Pikovsky A., Rosenblum M., *Synchronization: A Universal Concept in Nonlinear Science*. Cambridge University Press, Cambridge, 2001.

[21] F. Dalla Libera, T. Minato, H. Ishiguro, E. Pagello, and E. Menegatti, "Developing robot motions by simulated touch sensors," *SIMPAR 2008, LNAI*, vol. 5325, pp. 242–253, 2008.