Multiple chaotic central pattern generators with learning for legged locomotion and malfunction compensation

Guanjiao Ren[a], Weihai Chen[a], Sakyasingha Dasgupta[b],
Christoph Kolodziejski[b], Florentin Wörgötter[b], Poramate Manoonpong[b,c,*]

[a]*School of Automation Science and Electrical Engineering, Beijing University of Aeronautics and Astronautics, Beijing, 100191 China*
[b]*Bernstein Center for Computational Neuroscience, III Physikalisches Institut - Biophysik, Georg-August-Universität Göttingen, 37077 Germany*
[c]*Mæsk Mc-Kinney Møller Institute, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark*

Abstract

An originally chaotic system can be controlled into various periodic dynamics. When it is implemented into a legged robot's locomotion control as a central pattern generator (CPG), sophisticated gait patterns arise so that the robot can perform various walking behaviors. However, such a single chaotic CPG controller has difficulties dealing with leg malfunction. Specifically, in the scenarios presented here, its movement permanently deviates from the desired trajectory. To address this problem, we extend the single chaotic CPG to multiple CPGs with learning. The learning mechanism is based on a simulated annealing algorithm. In a normal situation, the CPGs synchronize and their dynamics are identical. With leg malfunction or disability, the CPGs lose synchronization leading to independent dynamics. In this case, the learning mechanism is applied to automatically adjust the re-

*Corresponding author. TEL: +45 6550 8698, FAX: +45 6615 7697
Email address: `poma@mmmi.sdu.dk` (Poramate Manoonpong)

maining legs' oscillation frequencies so that the robot adapts its locomotion to deal with the malfunction. As a consequence, the trajectory produced by the multiple chaotic CPGs resembles the original trajectory far better than the one produced by only a single CPG. The performance of the system is evaluated first in a physical simulation of a quadruped as well as a hexapod robot and finally in a real six-legged walking machine called AMOSII. The experimental results presented here reveal that using multiple CPGs with learning is an effective approach for adaptive locomotion generation where, for instance, different body parts have to perform independent movements for malfunction compensation.

1. Introduction

Humans, mammals, insects, and other arthropods employ legs for movement. Common to all of them is that their walking pattern usually shows a high level of proficiency adapted to the different terrains of their natural habitat. Legged robots, on the other hand, have not yet achieved this level of performance.

Optimized biomechanics and (neural) control create these efficient and often very elegant walking patterns in animals and some robots have copied this strategy with varying levels of success. Many reports have demonstrated gait generations in animals which are achieved through oscillations originating from the spinal cord (vertebrate) or from different ganglions (invertebrate) [18, 9]. This is known as the concept of central pattern generators

2

(CPGs) and has been applied to different types of legged robots, such as the bipedal robot designed by Taga et al. [52] and Aoi and Tsuchiya [2], Aoi et al. [1], the quadruped robot *Tekken* by Fukuoka et al. [24], Kimura et al. [38], the hexapod robots by Arena et al. [3, 4], Inagaki et al. [34, 35] and in our previous works [42, 51, 46]. Bio-inspired amphibious robots [44, 33, 11] and snake-like robots [10] also employed this kind of control strategy. Further details on CPG-based locomotion control have been reviewed in Ijspeert [32].

CPG-based locomotion is directly inspired by the way animals control their movement. It has many advantages, such as distributed control, the ability to deal with redundancies, and fast control loops. It also allows modulation of locomotion by simple control signals [32, 39]. When applied to robot control, we do not need to know the precise mechanical model of a robot. We can also easily integrate sensory information and adjust the control signal due to the simple structure of a CPG. Therefore, CPG-based control has already become an effective approach to perform legged locomotion in robots.

However, there are several problems yet to be solved. Although previous CPG-based algorithms can generate sophisticated gait patterns and deal with irregularities of the terrain to some extent [39], the problem of leg malfunction compensation in CPG-based control is still a challenging task. A troubling control problem can arise from the fact that the main controller usually contains CPGs which always control all legs with identical frequency [30]. If a robot suffers from leg failures, the other – still functioning – legs cannot immediately tune their oscillations appropriately. In contrast, insects can adjust the frequency of each leg individually [5, 49]. If their legs are mal-

3

functioning or disabled, they can still perform proper locomotion by changing the oscillation frequencies of the legs independently. Such a phenomenon also appears in mammals. For example, a cat can walk with the hind legs over a treadmill belt while the fore legs rest on a stationary platform [48] even after spinal cord injury. This indicates that the cat can independently adjust each leg's movement to achieve stable locomotion.

Traditional robotic methods for compensating leg malfunction are complicated [56, 37]. They are mostly based on kinematics or dynamics models [23]. Robots usually have to detect where a malfunction happens, then replan the gait pattern and choose another proper foot contact point. For different legs, the different foot trajectories are recalculated using inverse kinematics. Hence, all situations have to be considered and, as such, the procedure is computational intensive.

In contrast to the traditional control methods, we develop a CPG-based control strategy not only to generate multiple gaits but also to deal with leg malfunction. Inspired by multiple oscillators found in the neural system of insects [5, 16, 15], we extend our previously proposed chaotic CPG controller [51] to multiple CPGs, according to the number of legs of the robot. The CPGs can be synchronized or desynchronized to produce uniform or non-uniform patterns, respectively. If all CPGs are synchronized, the neural outputs are the same. If they are desynchronized, the neural outputs can oscillate at different frequencies. Thus, if some joints are disabled, other legs can change their oscillation frequencies independently. A simulated annealing (SA) based approach [20, 6] is applied to our robots in order to learn a suitable combination of leg oscillation frequencies, allowing leg malfunction

4

compensation to be achieved automatically. Furthermore, the applications to a hexapod robot and a quadruped robot demonstrate the effectiveness of our proposed algorithm and its generalization properties. To verify our algorithm in a real world application, our hexapod robot AMOSII is employed to evaluate the control strategy and learning. The proposed methods allow AMOSII to perform multiple gaits and to adapt its locomotion in case of disabled legs. Therefore, the main contribution of this paper is a novel control strategy relying on multiple chaotic CPGs with an additional automatic learning mechanism for leg malfunction compensation.

This article is structured as follows. Section 2 presents the overall control algorithm where the chaotic CPG is briefly introduced as a single oscillator. After which, we show how to design multiple CPGs and also state how the multiple CPGs synchronize and desynchronize with each other. Section 3 introduces the learning algorithm (simulated annealing) and the principle of selecting a suitable combination of leg oscillation frequencies for malfunction compensation. Section 4 demonstrates the implementation of the proposed multiple CPGs and the learning strategy on simulated hexapod and quadruped robots. Section 5 introduces our real hexapod walking platform - AMOSII. The learning results obtained from simulation are applied to the robot and the effectiveness of the results is successfully verified. Section 6 discusses the results, and finally in Section 7 we present our conclusion.
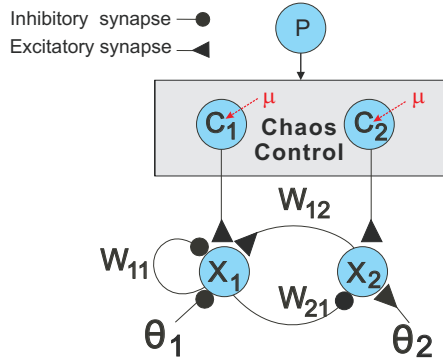
Figure 1: Single CPG with the chaos controller.

2. Multiple chaotic central pattern generators and synchronization mechanism

Our multiple CPGs-based locomotion controller is derived from the chaotic CPG controller, introduced in [51]. First, we describe a single CPG oscillator and then show how it can be extended to multiple CPGs. The synchronization and desynchronization mechanisms are also presented. The multiple CPGs generate either different periodic patterns independently, or they become synchronized and generate the same pattern. Here, they will be synchronized for basic locomotion generation and desynchronized for malfunction compensation.

2.1. Single chaotic CPG

The chaos control CPG unit is shown in Fig. 1. In this figure, x_1 and x_2 indicate the neurons that generate the oscillation, while c_1 and c_2 are the control inputs depending only on the period p with a control strength μ. w_{11}, w_{12}, w_{21} represent the synaptic weights and θ_1 and θ_2 indicate the

biases. Dynamics of the chaos control CPG can be exploited to generate complex patterns for legged robots, like chaotic leg motion and different walking patterns (multiple gaits). To achieve different walking patterns, we simultaneously add inputs to the two neurons, i.e., the control signals c_1 and c_2. These act as extra biases that depend on a single parameter p (the period of the output to be controlled). The output of the neurons is detected every p steps and the chaos is controlled to a p-period orbit by adjusting the control input. The discrete time dynamics of the activity (output) states $x_i(t) \in [0, 1]$ of the circuit satisfies:

$$x_i(t+1) = \sigma(\theta_i + \sum_{j=1}^{2} w_{ij} x_j(t) + c_i^{(p)}(t)) \quad \text{for } i \in \{1, 2\} \tag{1}$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$ is a sigmoid activation function with biases θ_i and w_{ij} is the synaptic weight from neuron j to i. Here, the weight and bias parameters are set as $w_{11} = -22.0, w_{12} = 5.9, w_{21} = -6.6, w_{22} = 0.0, \theta_1 = -3.4$ and $\theta_2 = 3.8$ [43], such that, if uncontrolled ($c_i^{(p)}(t) \equiv 0$) it shows chaotic dynamics. In order to obtain a given period p, the control input is given by:

$$c_i^{(p)}(t) = \mu^{(p)}(t) \sum_{j=1}^{2} w_{ij} \Delta_j(t) \tag{2}$$

It is calculated every p time steps, while for the other steps it is set to 0. In Eq. 2, $\Delta_j(t)$ indicates the activity difference between the current step and p-steps before:

$$\Delta_j(t) = x_j(t) - x_j(t-p) \tag{3}$$

and $\mu^{(p)}(t)$ is the control strength, which changes its value adaptively according to:

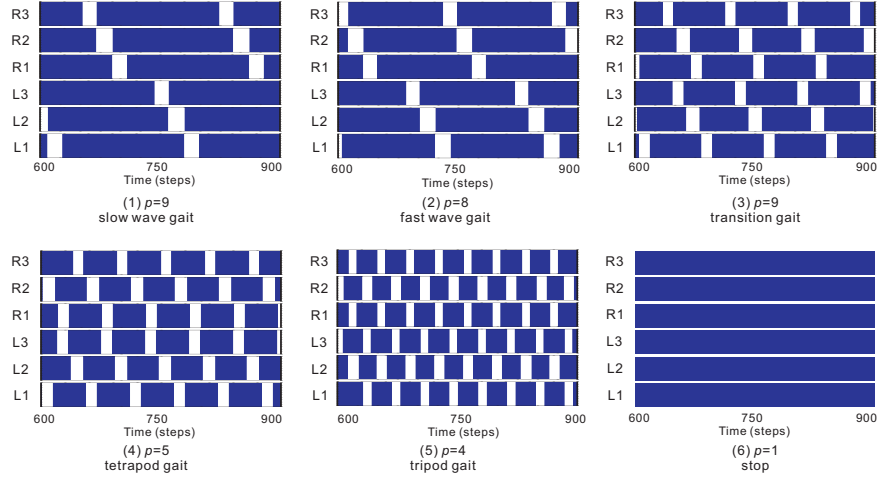$$\mu^{(p)}(t+1) = \mu^{(p)}(t) + \lambda \frac{\Delta_1^2(t) + \Delta_2^2(t)}{p} \tag{4}$$

7

Figure 2: Different hexapod gaits for changing p and the stop status ($p = 1$).

with an adaption rate λ, e.g. 0.05.

Thus, using this single chaotic CPG, different oscillation periods can easily be obtained just by changing the p value. After passing through some neural post-processing modules (see [51] for details), different gait patterns are produced for a hexapod robot; with an increase of p, the robot walks slower. Periods 9, 8, 6, 5, 4 indicate slow wave gait, fast wave gait, transition gait, tetrapod gait and tripod gait, respectively (see Fig. 2). A blue area means that this leg is in a support phase, i.e., it touches the ground, while a white area indicates the swing phase. Note that one time step is ≈ 0.037 s. If $p = 1$, there is no swing phase, i.e., the robot stops with all legs touching the ground. Periods 7 and 3 are unstable patterns while period 2 cannot generate a proper walking gait due to fast oscillation switching between two fixed points. Therefore, these periods are not used for locomotion generation.

Another useful function of this mechanism is the chaotic output. If we set

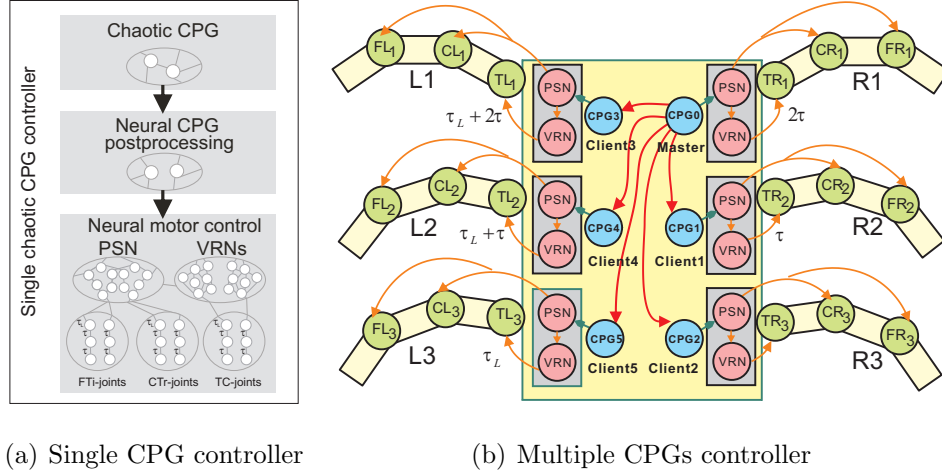(a) Single CPG controller   (b) Multiple CPGs controller

Figure 3: Single chaotic CPG (a) and multiple chaotic CPGs (b) for a multi-legged robot.

the control signal $c_i^{(p)}(t) \equiv 0$, the neural CPG circuit shows chaotic dynamics, which can be applied for self-untrapping, e.g., when a leg falls into a hole. It is important to note that in principle chaotic dynamics is exploited not only for the self-untrapping but also mainly for obtaining a stable period from a large number of unstable periods embedded in chaotic dynamics. Thus, without chaotic dynamics, different stable periods cannot be obtained in this scheme. Usually, the outputs of the CPG are passed to motor neurons, which activate the leg joints, through two hierarchical neural modules: neural CPG post processing and neural motor control (see Fig. 3(a) for an abstract diagram and [51] for the complete neural circuit). Since the neural CPG post processing and motor control have already been presented in our previous studies [42, 51], we only discuss them briefly here.

The neural CPG postprocessing, which directly receives the CPG outputs,

shapes the CPG signals to allow for smooth leg movements. Subsequently, the post-processed CPG outputs are transmitted to the neural motor control module. This module (see Fig. 3(a)) consists of one phase switching network (PSN) and two velocity regulating networks (VRNs). The PSN is a generic feed-forward network, which can reverse the phase of the processed CPG outputs with respect to a given input [42, 51]. It is implemented to achieve a proper phase shift between the CTr- and FTi-joints and allow for sideward walking [42]. The VRNs are also simple feed-forward networks, with each VRN controlling the three ipsilateral TC-joints on one side. Since the VRNs act qualitatively like a multiplication function [40], they have the capability to increase or decrease the amplitude of the TC-joint signals and even reverse them with respect to their control input. Controlling the TC-joint signal in this way results in various walking directions, such as forward/backward or turning left/right (see [42] for walking experiments). Finally, the outputs of the PSN and VRNs are sent to the motor neurons through delay lines. The ipsilateral lag is determined by a delay τ (i.e., 16 time steps or ≈ 0.6 s, see Fig. 3(a)) and the phase shift between both left and right sides are given by another delay τ_L (i.e., 48 time steps or ≈ 2 s, see Fig. 3(a)). This setup leads to biologically motivated leg coordination, since the legs on each side perform phase shifted waves of the same frequency [54]. The frequency of the waves is defined by the period p of the CPG, resulting in different gaits. In addition, sensory feedback can be integrated into the controller by targeting the VRNs; thereby leading to sensor-driven orienting behavior [42, 41].

2.2. Multiple chaotic CPGs

The single chaotic CPG is able to handle most situations where all joints of the robot are functional. However, if one or more of the joints are disabled, the robot cannot use the same gait to stay on its trajectory. In other words, the robot cannot compensate for leg malfunction with a single CPG. In contrast, real insects can control their locomotion to continue with their trajectory even though some legs are malfunctioning or damaged [27]. To do this, their legs show different frequencies in order to obtain effective walking patterns.

Inspired by this, we extend the original single CPG to multiple CPGs where the other modules (i.e., CPG post processing, PSN and VRN networks) are also replicated at each leg. The outputs of the PSN and VRNs of each leg are still sent to the corresponding motor neurons through the fixed delay lines, as described above. The control structure developed for hexapod locomotion is shown in Fig. 3(b). In this figure, the blue circles represent the CPGs. The right front neuron is the called master CPG, while other are called client CPGs. The neural motor control modules are depicted in gray rectangles. Each module consists of one PSN and one VRN which are depicted by pink circles. Green circles are motor neurons. The small dark green arrows indicate the signals from the CPG post processing modules (not shown here but see [51]). Red lines indicate the synchronization mechanism. Orange lines indicate delay lines transmitting the outputs of the PSN and VRN to the motor neurons. After delaying the signals by corresponding time steps, the outputs of the motor neurons are sent to the leg joints.

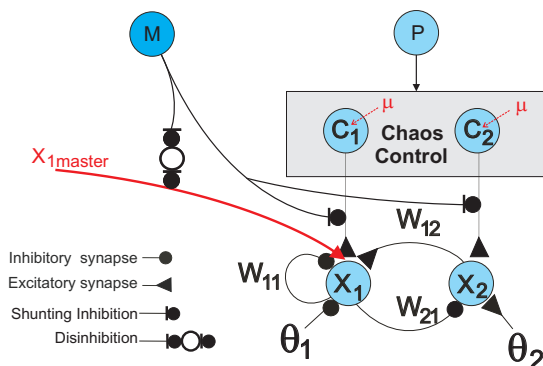The client CPGs can synchronize to the master in order to keep pace with

Figure 4: The inner structure of the client CPG.

the oscillation frequency. When synchronized, the controller generates the same outputs as if there was only one chaos control CPG. If some legs are disabled, the six CPGs automatically lose synchronization and can oscillate at different frequencies. A similar approach can be also applied to other walking schemes, e.g., quadruped locomotion (see below).

2.3. Synchronization and desynchronization

The inner structure of the master CPG is like that of our original chaotic CPG (Fig. 1), but the inner structure of the client CPGs is different. We add a synchronization mechanism to each client CPG as shown in Fig. 4. It is similar to the master CPG except for the synchronization mechanism where the M-neuron is introduced to enable or disable the signal from the master CPG. When a client CPG needs to synchronize to the master CPG, the M-neuron becomes active (i.e., value of 1) shunting the synaptic weight from the inputs (c_1, c_2). Thus, the outputs of the network are uncontrolled and, the output from the master CPG ($X_{1master}$, see Fig. 4) that was inhibited before

12

is passed to the client due to disinhibition. This results in the output of the client-CPG oscillating at the same frequency as the master CPG. When the client CPG needs to oscillate at its own frequency, the M-neuron becomes inactive (i.e., 0) switching off the inhibition by cutting down the connection from the master CPG. The control inputs (c_1 and c_2) can again pass through to the network and, as a consequence, all the CPGs oscillate at different frequencies.
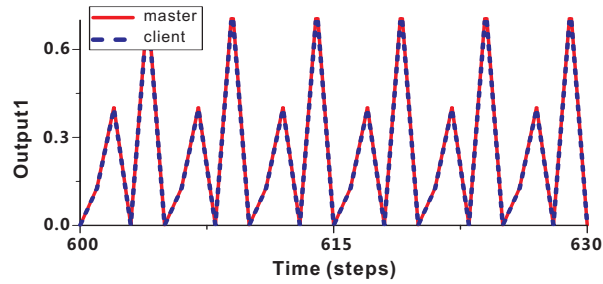
The following equations describe the details of the client CPG. With all legs functional, α is set to 1 such that the legs move with the same default frequency. If some legs malfunction, they are automatically set to 0 such that each leg can oscillate at its own frequency and learning will find a proper combination of oscillation frequencies of the different legs for malfunction compensation (see the next section). The outputs of the two neurons satisfy:

$$
\begin{aligned}
x_1(t+1) &= \sigma(a_1(t)) + \alpha(x_{1master} - \sigma(a_1(t))) \\
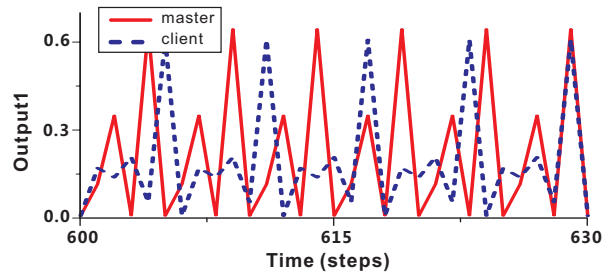x_2(t+1) &= \sigma(a_2(t))
\end{aligned}
\tag{5}
$$

where the activity satisfies:

$$
a_i(t) = \theta_i + \sum_{j=1}^{2} w_{ij} x_j(t) + c_i^{(p)}(t)
\tag{6}
$$

and α is the synchronization parameter. It is set to 1 if the M-neuron is active and 0 if it is inactive. The outputs of the first neuron for synchrony and asynchrony are shown in Fig. 5. In this figure, the master CPG has period 5 and the client CPG has period 6. Note that one time step is $\approx$ 0.037 s.

(a) Synchronous

(b) Asynchronous

Figure 5: The outputs of the CPG network for (a) synchrony and (b) asynchrony.

3. Learning for leg period adaptation

The desynchronization of the multiple chaotic CPGs enables each leg to oscillate at its own frequency. To adapt the leg period automatically for malfunction compensation, here we apply simulated annealing [20, 6] as our search algorithm to obtain feasible solutions. The algorithm is suitable to our task since it can be used for global optimization problems and discrete search space problems. The complete learning process is described as follows:

1) The robot starts to walk in a forward direction. While walking, its yaw
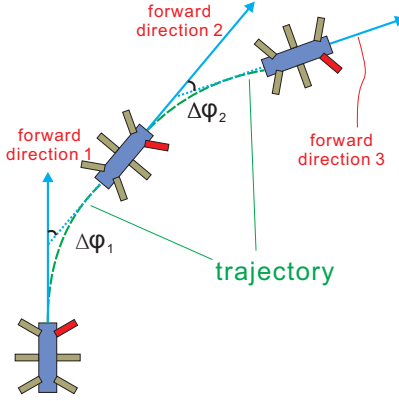
Figure 6: The method of deviation measure in the learning algorithm.

angle is recorded to determine the heading direction of the robot. 2) After a certain time window (400 time steps), the current yaw angle is measured and subtracted from the last or the initial one. The angle difference $\Delta\varphi$ is used for estimating a deviation, which is illustrated in Fig. 6. In red we depict the disabled leg. The green dashed line indicates the walking trajectory and the blue arrows point to the forward heading direction. $\Delta\varphi_1$ and $\Delta\varphi_2$ are the deviation angles, which are the yaw angle differences at certain steps. If there is no joint disabled, $\Delta\varphi$ should be approximately equal to zero, otherwise it indicates the degree of deviation. When a deviation occurs, i.e., when some joints are disabled, all CPGs automatically lose synchronization and oscillate independently. 3) The oscillation period of each leg is stochastically changed, and the deviation is re-evaluated. 4) This process is repeated until the deviation is below a specific threshold and therefore leg malfunction compensation is achieved.

Table. 1 depicts the learning process. In this table, $E_{1,2,...,n}$ denote the

Table 1: Learning algorithm.

Initialize $C(1) = [4, 4, 4, 4, 4, 4]; \Delta\varphi = 0.0; E_1 = 0.0$

Repeat:

At trials n

(1) randomly pick a leg l, $l \in [R1, R2, R3, L1, L2, L3]$

(2) change the period of leg l to a random value, $P(l) \in [4, 5, 6, 8, 9]$

(3) compare this combination of leg period, $C'(n)$, to the walking records

(4) repeat (1) to (3) until $C'(n)$ is a new combination of leg periods

(5) run the robot

(6) calculate the evaluation function and its variation

$$E_n = \Delta\varphi$$

$$\Delta E = E_n - E_{n-1}$$

(7) decide the combination of leg periods

    if $\Delta E < 0$ **then**

        $C(n) = C'(n)$

    else

        if $x \leq e^{-\beta\Delta E}$ **then**

            $C(n) = C'(n)$

        else

            $C(n) = C(n - 1)$

        end if

    end if

Until: The evaluation function E_n is less than a required value E_{req}

---

evaluation function in different trial. The function is calculated from the deviation angle, $\Delta\varphi$. E_{req} indicates the required minimum evaluation function. $\Delta E = E_n - E_{n-1}$ is the difference of the evaluation function between two trials. β is the adaption rate. X is a random value between 0 and 1. $C'(n)$ and $C(n)$ indicate the randomly selected combination of leg periods and the chosen combination of leg periods in the nth trial, respectively. If a deviation which is larger than a threshold occurs, the learning loop starts. A random leg is selected and one of the five oscillation periods (period 4, 5, 6, 8, 9) are randomly assigned to this leg. Then the combination of periods is compared to the previous states. If this particular combination of periods has already been performed, the trial is aborted, and another random leg is selected and randomly assigned to another period. Once a combination has been selected, the robot moves forward and the deviation angle $\Delta\varphi$ is measured. We set $E_n = \Delta\varphi$; if E_n is less than our required evaluation E_{req}, which means the deviation has already been compensated, the learning process is stopped and the loop is exited. If E_n is less than the evaluation of the last trial, i.e., $\Delta E = E_n - E_{n-1} < 0$, which means the deviation angle in this trial is less than last time, this combination of periods is stored and the next trial starts. If $\Delta E > 0$, this combination of periods is stored with probability $X < e^{-\beta\Delta E}$, which is in the range from 0 to 1 and decreases as ΔE increases. For example, if ΔE is very large, which means the deviation is much larger than last time, the acceptance probability $e^{-\beta\Delta E}$ is very close to 0. Thus, it is much more probable to return to the combination of periods of the last trial. Conversely, if the deviation is very small, it is more probable to keep the new combination of periods, and a new loop will start based on this new

combination of periods. The learning loop will be exited once the robot can walk straight even with disabled joints. With this learning process, the robot can automatically learn to find a feasible combination of its leg periods. In this implementation, the search algorithm is conducted continuously until a suitable solution is found.

In the learning process, β is an important parameter which creates a tradeoff between acceptance probability and convergence speed (small $\beta \rightarrow$ higher acceptance probability & slower learning and vice versa). We additionally observe that for large β the learning algorithm will often find inappropriate solutions. In this work, β is selected empirically (usually to 0.5) to balance this trade off. This, together with the use of a cost function (the "deviation angle") and a small, finite set of leg periods assures convergence of the simulated annealing method [6].

4. Implementation on different walking robots

4.1. Simulated six legs (6 CPGs)

LPZROBOTS[1] was employed as a simulation environment. For testing our multiple CPGs and the learning algorithm, a six CPGs controller was implemented on a simulated hexapod robot as shown in Fig. 7. The controller is updated with a frequency of 27Hz. With all the legs initially working well, we switched on the M-neuron to synchronize the client-CPGs to the master CPG. At the same time, the master CPG was set to periods 4, 5, 6, 8, 9, respectively (see. Fig. 2). All legs performed at the same frequency and

---

[1]It is based on the Open Dynamics Engine (ODE). For more details of the LPZROBOTS simulator, see `http://robot.informatik.uni-leipzig.de/software/`.
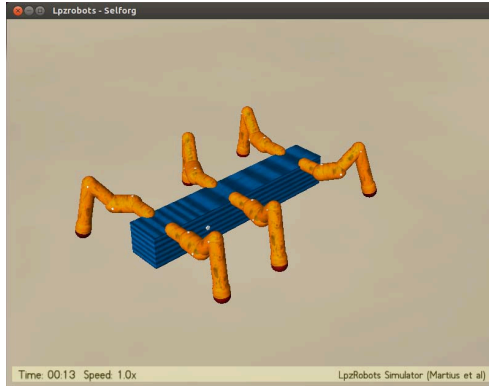
Figure 7: A hexapod robot in LPZROBOTS.

achieved tripod gait, tetrapod gait, transition gait, fast wave gait and slow wave gait, respectively. In synchrony, the robot performed just as if there was only one CPG.

To simulate leg malfunction, we disabled the movement of one or more legs by setting the outputs of the three joints of the leg to constant values. As a consequence, the affected leg could not move normally but only sustain part of the body weight intermittently (depending on the other legs' status). After disabling, the robot could not stay on a straight trajectory. In the simulation, an orientation sensor was implemented on the robot to measure its yaw angle. If the orientation sensor detected a deviation, the six CPGs automatically lost their synchronization and oscillate independently. Afterward, the learning process began as described in Section 3. For every 400 time steps, we calculate the deviation angle by subtracting yaw at the start from the end of this time window. This deviation was evaluated against the current combination of leg periods and the robot tested different combinations until a suitable one is found. As a result, leg malfunction was compensated

NO.	R1	R2	R3	L1	L2	L3	Deviation Angle (degree)	Decision
0	4	4	4	4	4	4	37.8604	Start
1	4	4	4	5	4	4	21.4578	Keep
2	4	4	9	5	4	4	64.1167	Return to No.1
3	4	4	4	5	9	4	14.0168	Keep
4	4	4	4	5	6	4	9.5276	Keep
5	4	5	4	5	6	4	9.6890	Keep
6	4	5	4	5	6	8	21.8642	Return to No.5
7	4	5	4	5	6	5	0.2451	End

Figure 8: The learning process for one scenario (see text for details).

for, and the robot maintained its forward walking in a straight line. It is important to note that in our implementation the disabled leg was not taken into account for period change since changing its period does not affect the walking behavior. Thus, its period was kept fixed.

As an example, Fig. 8 illustrates the learning process for one scenario. In this figure, the first column indicates the nth trial. R1 = right front, R2 = right middle, R3 = right hind, L1 = left front, L2 = left middle, L3 = left hind. These six columns show the periods of the corresponding legs. The deviation in this trial is shown in degree. The last column shows the decision to keep this new period combination or return to a former one. In this scenario, the R1 leg was disabled (depicted in red) and its period was ignored in the learning process. Initially, the robot walked with tripod gait, i.e., every CPG oscillated with period 4. When the right front leg (R1) was disabled, the robot performed a right turning curve since it cannot provide enough propelling force to balance the body. After 400 time steps, the robot deviated to the right with an angle of deviation $\approx 37.86°$. In step 1, the

left front leg (L1) was randomly selected and randomly changed to period 5. After 400 time steps, the deviation angle was $\approx 21.46°$. This combination of periods was kept since the deviation angle decreased. In step 2, the right hind leg (R3) was selected and changed to period 9 and the deviation was $64.12°$. As $\beta = 0.5$, the acceptance probability of this combination of periods satisfies:

$$P = e^{-\beta \Delta E} = e^{-0.5 \times (64.12 - 21.46)} = 5.45 \times 10^{-10}, \tag{7}$$

which is very small. As a result, this trial was aborted and the robot returned to step 1. Hence, step 3 was derived from step 1 rather than step 2. The deviations in step 3 and step 4 decreased so they were kept. In step 5, the deviation was a little larger than in step 4. However, the probability of keeping this combination of periods was close to 1, and it was kept. This shows the advantage of the simulated annealing method: it accepts worse situations which provide an opportunity to approach a better solution. Step 6 was aborted. In step 7, we obtained a final solution through changing the left hind leg (L3) from period 4 (in step 5) to period 5. Final deviation was only $\approx 0.25°$.

According to this, a possible solution to cope with the problem of the R1 leg malfunction was a combination of the following periods: R2 = 5, R3 = 4, L1 = 5, L2 = 6 and L3 = 5. The joint angles of the 18 actuators performed as shown in Fig. 9. In this figure, the outputs of the FTi- joints of hind legs (e.g., Leg L3 and R3) were reversed in order to push the leg outwards to stabilize the body. Note that one time step is ≈ 0.037 s.

The foot contact forces were recorded to see how the robot interacts with the environment (see Fig. 10). In this figure, the signals have a range from 0

(a) Leg R1: disabled (b) Leg R2: p5 (c) Leg R3: p4
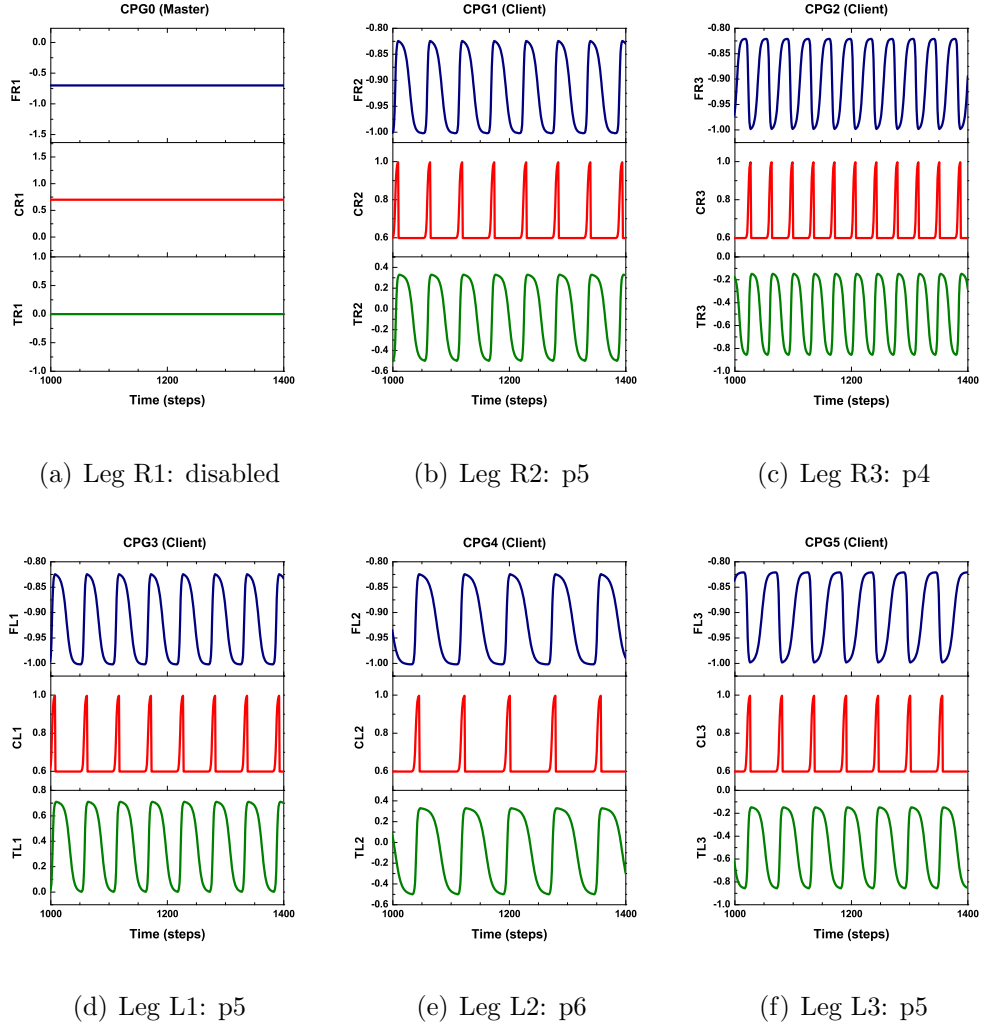
(d) Leg L1: p5 (e) Leg L2: p6 (f) Leg L3: p5

Figure 9: Outputs of the 18 motor neurons. The FTi-, CTr- and TC- joints are depicted in blue, red and green, respectively. Leg R1 (a) was disabled (i.e., the outputs of the three joints were kept at a constant value) while L2 (e) oscillated with period 6. The oscillation of R2 (b), L1 (d) and L3 (f) were decreased to period 5 and R3 (c) stayed on period 4.
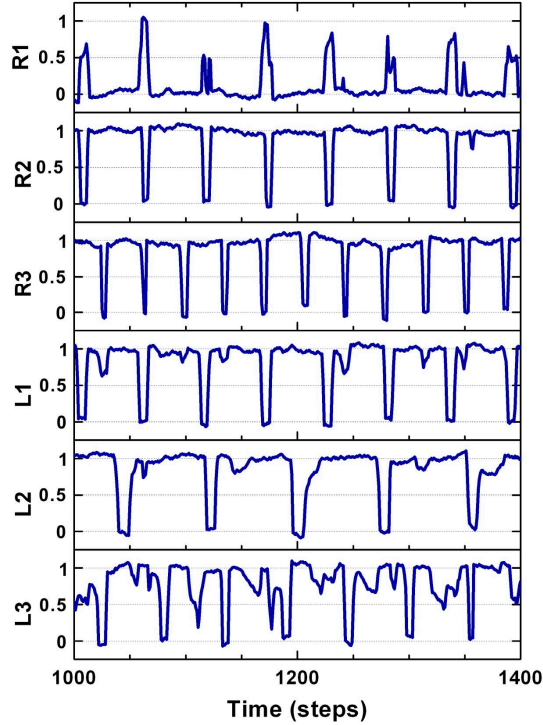
22

Figure 10: The foot contact signals of the six legs. Here, the leg R1 was disabled.

to 1. Positive 1 means the leg fully touches the ground (stance phase), while negative 1 means the leg is in the air (swing phase). The functional legs performed similar patterns relating to the input oscillation: for example, the R3 leg touched the ground with p4 and the L2 leg touched the ground with p6. However, the disabled R1 leg still touched the ground intermittently to support the body, especially when the neighboring legs attempted to swing (e.g., R2 and L1), even though its joints were fixed.

Other scenarios for learning to compensate leg malfunction were also

23

Leg status	Disabled leg(s)	Example of periods after learning	Average deviation angle (deg) + SD	Trials + SD
	R1	4, 5, 4, 5, 6, 5	3.61 ± 2.10	2.8 ± 1.99
	R2	4, 4, 4, 6, 4, 4	3.52 ± 2.00	1.9 ± 1.51
	R3	4, 4, 4, 9, 4, 4	3.45 ± 2.45	7.9 ± 9.05
	L1	4, 6, 4, 4, 4, 4	4.57 ± 2.30	2.4 ± 1.69
	L2	4, 4, 6, 4, 4, 4	4.88 ± 1.98	2.6 ± 2.11
	L3	4, 6, 4, 4, 4, 4	4.47 ± 1.66	4.6 ± 4.72
	R2, R3	4, 4, 4, 5, 8, 9	4.24 ± 2.69	8.0 ± 5.31
	R1, R3	4, 4, 4, 6, 4, 9	2.64 ± 2.05	6.1 ± 3.94
	R1, R2	4, 4, 4, 8, 9, 6	4.35 ± 1.59	7.6 ± 2.20
	R1, L3	4, 5, 4, 4, 5, 4	3.29 ± 2.16	10.9 ± 5.84
	R2, L3	8, 4, 4, 4, 8, 4	4.17 ± 2.29	10.7 ± 6.77
	R1, L2	4, 4, 4, 4, 4, 4	0.83 ± 0.04	0.0 ± 0.0
	R1,R3,L1	4, 4, 4, 4, 5, 5	4.39 ± 2.09	7.7 ± 6.51
	R1,R3,L3	4, 4, 4, 4, 9, 4	5.08 ± 1.29	3.6 ± 3.83
	R1,R3,L2	4, 4, 4, 4, 4, 9	2.01 ± 1.61	5.7 ± 5.92
	R1,R2,L1	4, 4, 5, 4, 8, 8	4.67 ± 2.50	4.9 ± 3.08
	R1,R2,L3	4, 4, 4, 5, 9, 4	3.81 ± 2.31	5.8 ± 4.04
	R1,R2,L2	4, 4, 4, 4, 4, 9	2.98 ± 2.21	3.5 ± 2.46
	R2,R3,L1	4, 4, 4, 4, 8, 5	2.49 ± 2.07	3.8 ± 2.52
	R2,R3,L3	4, 4, 4, 6, 5, 4	4.94 ± 2.37	3.7 ± 2.10
	R2,R3,LM	4, 4, 4, 4, 4, 8	5.03 ± 1.93	3.3 ± 1.55

Figure 11: Combination of periods after learning for the hexapod. The first two columns illustrate the functional and disabled legs. The third column is the combination of periods after learning. In red we depict the disabled legs. The last two columns represent the deviation angle and how many trials are on average required with standard deviations (SD).

24

tested and the results are depicted in Fig. 11. For each scenario, 10 trials had been averaged. In this diagram, all simulations started from period 4. In all cases, the learning state space for the hexapod consists of $5^6 = 15625$ different controllers. In this figure, six scenarios for one leg's malfunction, six scenarios for two legs' malfunction and nine scenarios for three legs' malfunction were tested. These scenarios include most of the situations that might occur (see the first two columns), excluding situations that have a symmetrical counterpart in this Table. For example, the situation with the L2 and L3 legs disabled is not depicted here because we can deduce it from the scenario that has the R2 and R3 legs disabled. In Fig. 11, the best resulting period combination of each scenario is presented.

The actual time needed to conduct a real robot experiment can be approximately estimated. As one trial takes 400 time steps and the control frequency for a robot experiment is 27 Hz, it costs $400/27 \approx 14.8$ seconds to try one combination of periods. From the results shown in Fig. 11, the learning time for most of the scenarios can be normally controlled within 20 trials, i.e., it takes about 4 to 5 minutes to finish one learning experiment.

4.2. Simulated four legs (4 CPGs)

The multiple chaotic CPGs and the learning algorithm presented here are not limited to hexapod robots but can also be applied to different walking robots. To verify this, we simulated a quadruped robot in LPZROBOTS (see Fig. 12). For the controller of this robot, we used 4 CPGs with, as before, one CPG set as master CPG and the other three as clients. The controller is updated with a frequency of 27Hz. Synchronization and learning mechanisms were the same as for the 6 CPGs' controller. All trials started
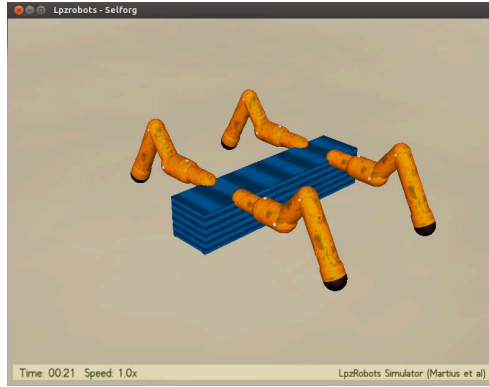
25

Figure 12: A simulated quadruped robot in LPZROBOTS.

from period 4, i.e., a trot gait for the quadruped robot. In this simulation, we disabled the R1, R2, L1 and L2 legs individually to test the learning process for leg malfunction compensation (see Fig. 13). Similar to Fig. 11, the average deviation angles (deg) and trials are shown together with the standard deviations (SD). R1 = right front, R2 = right hind, L1 = left front, L2 = left hind. Every scenario was tested 10 times. In all cases, the learning state space for the quadruped consists of $5^4 = 625$ different controllers. The experimental results show that the controller can find feasible combinations of periods in the different conditions of leg malfunction for the quadruped robot. This shows that our multiple CPGs and learning mechanism can be applied to not only a hexapod robot but also to a quadruped robot, indicating a certain potential for generalization. In other words, this generic approach can deal with not only small (i.e., quadruped) but also large (i.e., hexapod) learning state space.

Leg status	Disabled leg	Example of periods after learning	Average deviation angle (deg) + SD	Trials + SD
	R1	4, 4, 8, 6	3.19 ± 2.52	4.4 ± 2.06
	R2	6, 4, 9, 8	3.87 ± 2.33	4.8 ± 3.84
	L1	5, 8, 4, 4	5.50 ± 1.53	4.3 ± 3.29
	L2	5, 8, 5, 4	4.94 ± 2.45	4.7 ± 3.66

Figure 13: Learning results for quadruped locomotion. All columns are presented similar to Fig. 11.

4.3. Influence of β

An important parameter which needs to be discussed in our learning method is the annealing factor β. It determines the learning time and whether it can converge globally. In our learning process, if β is set to 0 the acceptance probability is 1, i.e., every possible change of leg periods is acceptable. Therefore, the search is performed by random permutations.

If β increases to positive infinity, the acceptance probability approaches zero. Therefore, the current combination of periods will be discarded only if the deviation is larger than the previous one, i.e., the learning algorithm does not accept worse solutions. In this condition, the simulated annealing method is reduced to a greedy search method.

Fig. 14 shows the number of trials when we set different annealing factors β. The blue bar represents random permutations ($\beta = 0$) and the green bar represents approximately a greedy search ($\beta = 10$). The red bars indicate results from the simulated annealing method when we use five different β.
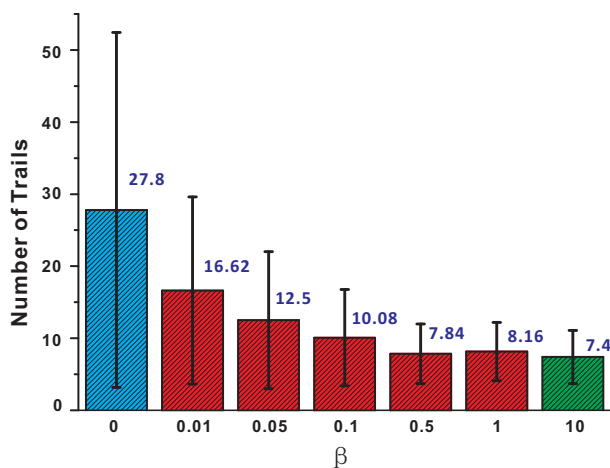
Figure 14: The average number of trials with different β (see text for details).

Every situation (i.e., every β) was tested for 50 times and the mean value is depicted by the blue numbers upon the bars. In this figure, the R1 and R2 leg were disabled while other legs were functional.

When $\beta = 0$ (random permutation), we observed the longest search time. With increase of β, search time decreased, however, if β was too large (larger than 10 here), learning might get stuck at a worse solution. For example, in a scenario where R1 and L2 were disabled, the greedy search got stuck at a solution, where the remaining legs performed with periods R2=4, R3=4, L1=4 and L3=4. In this situation, the deviation angle remained at about 9.24°, which is larger than the predefined threshold of 8°. No matter how we changed a leg's oscillation frequency, the deviation angle increased such that all combinations were discarded. However, as shown in Fig. 11, the combination of R1=8, R3=4, L1=4 and L2=8 (line 11) would have been a suitable solution (found by simulated annealing, $\beta = 0.5$).

In summary, greedy search has a fast convergence speed but is more susceptible to getting stuck at a solution which might be worse than the solution obtained by SA. That is, the average deviation angle from the greedy search might be still larger than the one obtained from the simulated annealing. Employing random permutations will converge globally but costs more time. The simulated annealing technique lies between greedy search and random permutations. In principle, it is globally convergent and the search time is short if we employ an appropriate annealing factor [28]. Finally, SA is able to find a suitable combination of periods for compensating leg malfunction. Therefore, the simulated annealing method is a good choice for our learning process.

5. Real robot experiments

5.1. The walking machine platform AMOSII

In order to test our algorithm in a physical system, the six-legged walking machine AMOSII[2] is employed (see Fig. 15(a)). It has identical leg structure with three linkages (coxa, femur, and tibia, see Fig. 15(b)). Each leg has three joints: the thoraco-coxal (TC-) joint enables forward (+) and backward (−) movements, the coxa-trochanteral (CTr-) joint enables elevation (+) and depression (−) of the leg, and the femur-tibia (FTi-) joint enables extension (+) and flexion (−) of tibia. Compared to a real insect [18], the tarsus is ignored in the current design. In general the tarsus is for absorbing outside

---

[2]AMOSII was developed by Bernstein Center for Computational Neuroscience at Georg-August-Universität Göttingen in collaboration with Fraunhofer Institute IAIS, Germany.
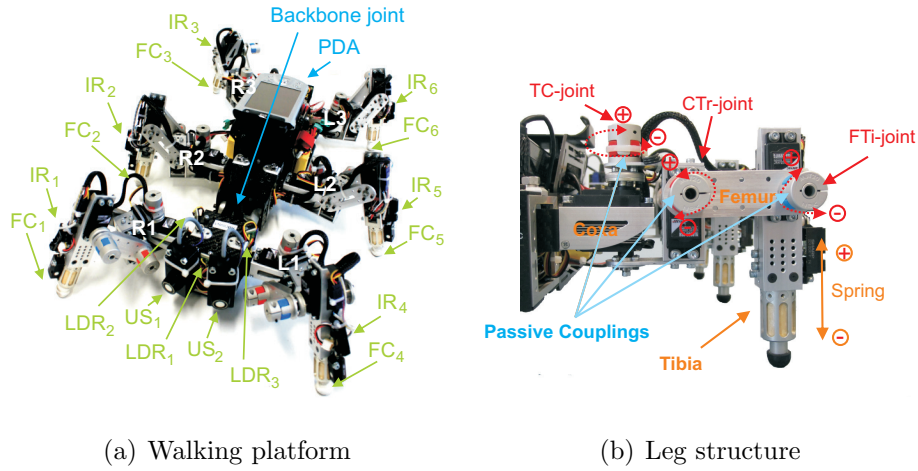
(a) Walking platform   (b) Leg structure

Figure 15: (a) Biologically inspired walking machine platform AMOSII with sensors. (b) Leg structure inspired by a cockroach leg.

impact forces and to stick the leg to a walking surface [12]. Nevertheless, a spring is installed in the leg to substitute part of the function of the tarsus; i.e., absorbing the impact force during touchdown on the ground. In addition, a passive coupling is installed at each joint (see Fig. 15(b)) in order to yield passive compliance and to protect the motor shaft. The body consists of two parts: two front legs on the front part, and the middle and hind legs on the hind part. The two body parts are connected by an active backbone joint which enables the rotation around the lateral or transverse axis. This backbone joint is mainly used for climbing which is not the main focus here (but see Goldschmidt et al. [25]). All leg joints as well as the backbone joint are driven by digital servomotors.

The robot has six infrared sensors ($IR_{1,...,6}$) at its legs, six force sensors ($FC_{1,...,6}$) in its tibiae, three light dependent resistor sensors ($LDR_{1,2,3}$) ar-

ranged in a triangle shape on the front body part, and two ultrasonic sensors ($US_{1,2}$) at the front body part (see Fig. 15(a)). The force sensors are for recording and analyzing the walking patterns. The infrared sensors are used for detecting obstacles near the legs and the ultrasonic sensors are used for detecting obstacles in front. The light dependent resistor sensors serve to generate positive tropism like phototaxis. We use a Multi-Servo IO-Board (MBoard) installed inside the body to digitize all sensory input signals and generate a pulse-width-modulated signal to control servomotor position. The MBoard can be connected to a personal digital assistant (PDA) or a personal computer (PC) via an RS232 interface. For the robot walking experiments presented here, the MBoard is connected to a PC on which the neural controller is implemented. Electrical power supply is provided by batteries: one 11.1 V lithium polymer 2,200 mAh for all servomotors and two 7.4 V lithium polymers for the electronic board (MBoard) and for all sensors.

5.2. Experimental results

The real robot experiments were conducted to test the validity of the proposed multiple CPGs and learning algorithm. The controller is implemented with the update frequency of 27Hz. The robot was placed in front of a tunnel which was 300 cm long and 120 cm wide (see Fig. 16). Three scenarios were tested: in the first, second, and third experiment, we disabled the R1 leg, the R1 and R3 legs, and the R1, R3, and L2 legs, respectively. Using default periods (i.e., all legs moved with period 4), the robot deviated to the right and hit the right board in all three experiments, i.e., it failed to pass the tunnel. For the R1 leg disabled, the robot deviated to the right by 48 cm after travelling 53 cm. We reset the robot to the initial

position and implemented the learned suitable combination of leg periods, i.e., 5, 4, 5, 6, 5 (obtained directly from the simulation, see line 1 in Fig. 11, in the sequence of R2, R3, L1, L2, L3). These periods enabled the robot to pass the tunnel with a small deviation of approximately 42 cm. For the R1 and R3 legs disabled using the default periods, the robot deviated 47 cm after travelling 120 cm. We used the learned combination of 4, 6, 4, 9 (see line 8 in Fig. 11, in the sequence of R2, L1, L2, L3) and observed only a small deviation of 5 cm. For the R1, R3, and L2 legs disabled using the default periods, the robot deviated 45 cm after travelling 105 cm. In this situation, we used the learned periods 4, 4, 9 obtained directly from the simulation (see line 11 in Fig. 11, in sequence of R2, L1, L3), resulting in a deviation of 14 cm. The deviations are also shows as angles in Table 2. Experimental snapshots and gait diagrams are shown in Fig. 16. The experimental video can be seen in the supplementary material and at our website `http://manoonpong.com/MultiCPGs/supple_video.wmv`. The video shows that the results from our simulations can directly transfer to the real robot resulting in the malfunction compensation in the robot's locomotion.

The experiments demonstrate the effectiveness of the proposed multiple CPGs with the learning mechanism. Additionally, as it is flexible, it can be easily extended to configurations other than the six CPGs implemented here; it can also be extended to 4 CPGs for the quadruped robot as shown in the simulation and 8 CPGs for a possible octopod robot in future research.
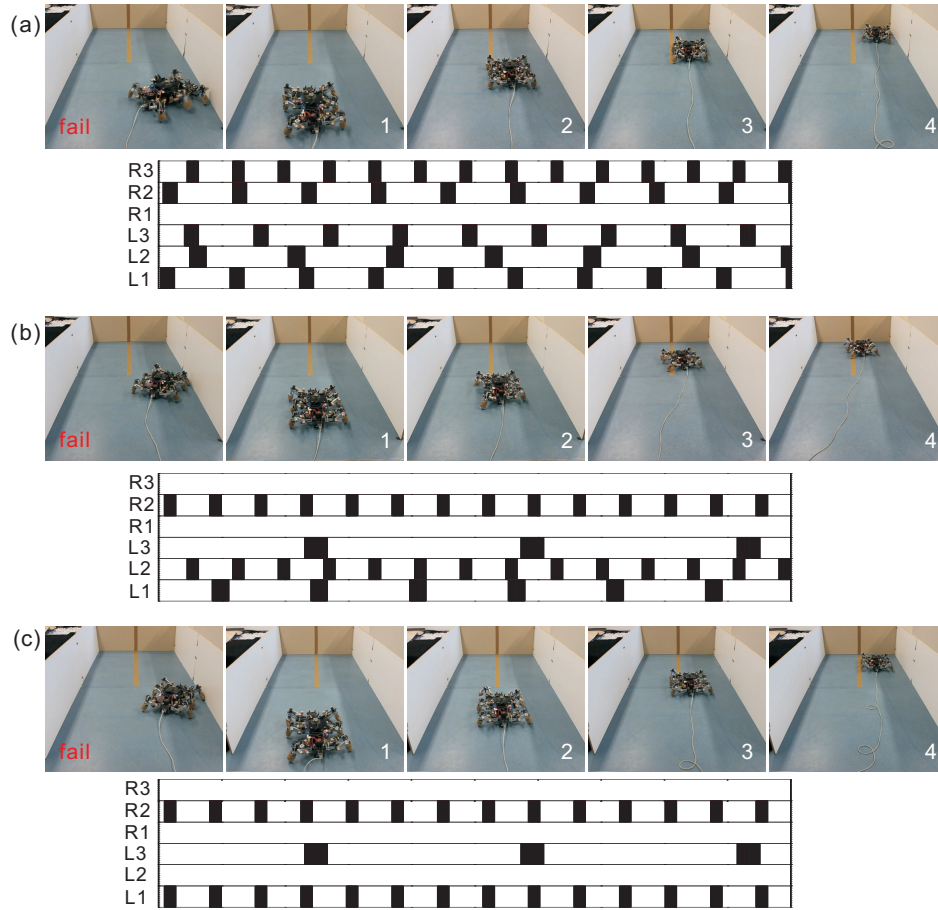
Figure 16: Three scenarios of the real robot experiments. For each subfigure, the upper panel shows one snapshot of a fail situation (before learning) and four snapshots of a success situation (after learning). The lower panel shows the gait (i.e., suitable leg frequencies) after learning. The gait is observed from the motor signals of the CTr-joints. A black area means that the leg touches the ground, while a white area indicates that the leg is in the air. (a) R1 leg disabled. (b) R1 and R3 legs disabled. (c) R1, R3, and L2 legs disabled.

33

Table 2: The deviation angle (deg) before learning and after learning.

Scenarios	Disabled leg(s)	Before learning	After learning
1	R1	$\approx 42°$	$\approx 7.97°$
2	R1, R3	$\approx 21°$	$\approx 0.95°$
3	R1, R3, L2	$\approx 23°$	$\approx 2.67°$

6. Discussion

Our contribution here is an extension of our previous works [51, 46, 45]. In Steingrube et al. [51], we proposed a single chaotic CPG for generating multiple gaits and complex behaviors, but excluded leg malfunction compensation. In Ren et al. [46], we investigated three chaotic CPGs with manual frequency tuning for malfunction compensation of a hexapod robot only. In Ren et al. [45], only a short introduction to the work is given. The complete technical details, analysis, or experimental results presented here have not been published in our previous papers [51, 46, 45] or elsewhere. In general, implementing such a CPG-based control strategy does not require the precise mechanical model. Furthermore, it does not need to calculate inverse kinematics, which requires a lot computational resources.

The strategy of using multiple CPGs is inspired by the way insects control their locomotion. There were already experiments demonstrating that each insect leg is controlled by an independent oscillation center [5]. Additionally, they showed that legs are able to respond to their corresponding ganglions to generate rhythmic motion after amputation [49, 17]. Our control strat-

34

egy is similar; the multiple CPGs are a decentralized system and based on a modular concept, with each leg controlled by an independent oscillator. Although our CPG model, exhibiting chaotic dynamics, and learning used here do not directly match biological findings, they form a powerful approach for robot locomotion control and learning. The multiple chaotic CPGs perform as open-loop controllers and only an orientation sensor is required and used as sensory feedback for learning. This results in a minimalistic system where learning can efficiently and automatically find a proper combination of oscillation frequencies of the different legs for malfunction compensation. Note that our approach is mainly developed to benefit technical systems like the walking robots used here. In real insects certain hard-wired sensori-motor loops are used instead of the learning mechanisms implemented in this study. For a robot however, the modeling of any such loop requires carefully addressing quite complex neuro-biological design issues [24, 33]. Here, we demonstrate that learning can replace hard-wired sensorimotor loops, which for robotics substantially reduces design demands and we show that learning can indeed take the role of (evolutionary designed) sensori-motor structures regardless of the robot's specific embodiment (e.g., six- and four-legged robots shown here).

Some other approaches also successfully showed fault tolerance. For example, Bongard et al. [7] presented an active process that allowed a robot to generate successful motor patterns for locomotion, before and after damage, through autonomous and continuous self-modeling. This algorithm was tested on a four legged starfish robot. Schilling et al. [49] improved the "Walknet" by adding an analog-selector for dealing with leg amputations.

An Artificial Immune System (AIS) was implemented on the OSCAR robot in order to detect leg anomaly and a Swarm Intelligence for Robot Reconfiguration (SIRR) method was applied to rearrange the body shape and to regulate the leg behaviors [36, 22, 21]. Spenneberg et al. [50] realized fault tolerance locomotion of their octopod robot SCORPION by changing its gait to a hexapod gait. Yang and Kim [56] developed a fault tolerance mechanism based on a modeling and planning strategy to achieve stable walking with leg loss. Christensen et al. [8] applied learning to automatically adjust a quadruped robot's gait such that fault tolerance and morphology optimization were realized. Compared to many of these approaches, we emphasize here a simple but robust CPG-based mechanism with learning for both multiple gait generation (see Fig. 2) and leg malfunction compensation (see Fig. 11). The mechanism deals with only one parameter of each CPG for adaptation and requires only an orientation sensor as sensory feedback for learning. Therefore, it converges faster compared to, e.g., the results shown in [8]. We have also shown that it is easy to combine the learning mechanism with the multiple CPGs controller which is transferable to different platforms (see Fig. 11 and 13).

For our robot's locomotion control, especially for the malfunction compensation experiments, the center of mass of the body was maintained at a low level. In this configuration, the body sometimes touches the ground thus being able to support part of the body weight. Because of this, the remaining legs do not need to carry as much load and as a result the robot can move in a stable manner. This is similar to the way insects (e.g., cockroaches) perform their locomotion [47]. This configuration is especially effective when some of

the robot's legs are disabled. In this situation, the functional legs are responsible for propelling the robot while the body is used to support the robot. If the robot meets with rough terrain, it can raise its body by depressing the legs to overcome the terrain irregularity. To compensate leg malfunction during walking on rough terrain, additional posture control might be required for balancing. However, investigating this issue is beyond the scope of this work.

When legs are disabled (one, two or even three), the resulting trajectory deviation can be compensated for by the proposed algorithm. The learning process starts as soon as the deviation exceeds the predefined threshold. Eventually the learning algorithm will find a suitable combination of periods. As the controller is modular, it is flexible and general [31, 53] and it can be extended from one CPG to the four or six CPGs presented here. Furthermore, the modularity of the system offers the future possibility of combining it with other neural sensory processing modules to generate various sensor-driven behaviors (like obstacle avoidance, phototropism, not discussed here, see [42, 51]). The modules are inhibited in case of leg malfunction, thereby not interfering with the learning process for leg malfunction compensation.

7. Conclusion

In this paper, we focus on the locomotion control of legged robots using neuron-inspired central pattern generators for the purpose of automatic malfunction compensation. This is a continuation of our previous investigations, where we had already demonstrated that a chaotic CPG could generate complex behaviors, such as different walking gaits as well as novel self-untrapping

37

mechanism [51, 46].

The main advantage of CPG based control is that it does not require a precise mathematical model of leg structure. However, with a single CPG, the robot cannot compensate for leg malfunction. As such, taking inspiration from multiple oscillators found in the real neural systems of insects, here we first extended our single chaotic CPG to multiple chaotic CPGs. The different CPGs can now synchronize to generate uniform frequency or desynchronize to oscillate independently. Here, robot leg malfunction causes the different CPGs to oscillate asynchronously. In this setup, in order to deal with the leg malfunction, an on-line learning mechanism based on the simulated annealing (SA) technique is applied. The SA algorithm allows the robot to automatically find a suitable combination of periods (frequency) for each leg and thereby considerably reduce any deviation from the original movement trajectory caused due leg malfunction. Furthermore, we show that such a SA based learning algorithm is capable of converging to a solution within acceptable time without getting stuck at a worse combination of periods. We clearly demonstrate the effectiveness and generalization of our learning algorithm using both simulations and real robot experiments.

In this study, although our control approach allows walking robots to learn to compensate for leg malfunction, individual joint malfunctions and body damage compensation have not been addressed. In the future, we will investigate these issues which might require more complex sensory information [29, 57] as well as additional mechanisms, such as advanced force control [49, 26] and suitable muscle models [55] to enable leg compliance and adaptation [19]. Furthermore, cognitive abilities such as memory [13] and decision

making [14] will be also applied for goal-directed navigation learning. In addition, due to the employed chaotic CPGs controller which in principle can reliably obtain a stable (complex) periodic pattern out of a large (often infinite) number of unstable periodic patterns [51], we will also use the controller to deal with other motor tasks, like manipulation which requires a variety of complex patterns.

Acknowledgment

References

[1] S. Aoi, Y. Egi, R. Sugimoto, T. Yamashita, S. Fujiki, K. Tsuchiya, Functional roles of phase resetting in the gait transition of a biped

robot from quadrupedal to bipedal locomotion, IEEE Transactions on Robotics (2012) 1–16.

[2] S. Aoi, K. Tsuchiya, Stability analysis of a simple walking model driven by an oscillator with a phase reset using sensory feedback, IEEE Transactions on Robotics 22 (2006) 391–397.

[3] P. Arena, L. Fortuna, M. Frasca, Multi-template approach to realize central pattern generators for artificial locomotion control, International journal of circuit theory and applications 30 (2002) 441–458.

[4] P. Arena, L. Fortuna, M. Frasca, G. Sicurella, An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 34 (2004) 1823–1837.

[5] U. Bässler, A. Büschges, Pattern generation for stick insect walking movements-multisensory control of a locomotor program, Brain Research Reviews 27 (1998) 65.

[6] D. Bertsimas, J. Tsitsiklis, Simulated annealing, Statistical Science 8 (1993) 10–15.

[7] J. Bongard, V. Zykov, H. Lipson, Resilient machines through continuous self-modeling, Science 314 (2006) 1118–1121.

[8] D.J. Christensen, J.C. Larsen, K. Stoy, Fault-tolerant gait learning and morphology optimization of a polymorphic walking robot, Evolving Systems 5 (2014) 21–32.

[9] A. Cohen, P. Holmes, R. Rand, The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: a mathematical model, Journal of Mathematical Biology 13 (1982) 345–369.

[10] A. Crespi, A. Ijspeert, Online optimization of swimming and crawling in an amphibious snake robot, IEEE Transactions on Robotics 24 (2008) 75–87.

[11] A. Crespi, K. Karakasiliotis, A. Guignard, A.J. Ijspeert, Salamandra robotica ii: an amphibious robot to study salamander-like swimming and walking gaits, IEEE Transaction on Robotics 29 (2013) 308–320.

[12] H. Cruse, V. Dürr, M. Schilling, J. Schmitz, Principles of insect locomotion, in: P. Arena, L. Patané (Eds.), Spatial Temporal Patterns for Action-Oriented Perception in Roving Robots, Springer, Berlin, Germany, 2009, pp. 43–96.

[13] S. Dasgupta, F. Wörgötter, P. Manoonpong, Information dynamics based self-adaptive reservoir for delay temporal memory tasks, Evolving Systems 4 (2013) 235–249.

[14] S. Dasgupta, F. Wörgötter, J. Morimoto, P. Manoonpong, Neural combinatorial learning of goal-directed behavior with reservoir critic and reward modulated hebbian plasticity, in: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC2013), IEEE, Manchester, UK, 2013, pp. 993–1000.

[15] S. Daun-Gruhn, A. Büschges, From neuron to behavior: dynamic equation-based prediction of biological processes in motor control, Biological cybernetics 105 (2011) 1–18.

[16] S. Daun-Gruhn, T. Tóth, An inter-segmental network model and its use in elucidating gait-switches in the stick insect, Journal of computational neuroscience 31 (2011) 43–60.

[17] F. Delcomyn, Factors regulating insect walking, Annual review of entomology 30 (1985) 239–256.

[18] F. Delcomyn, Walking robots and the central and peripheral control of locomotion in insects, Autonomous Robots 7 (1999) 259–270.

[19] R. Der, Self-organized acquisition of situated behaviors, Theory in Biosciences 120 (2001) 179–187.

[20] K.A. Dowsland, J.M. Thompson, Simulated annealing, in: G. Rozenberg, T. Bäck, J.N. Kok (Eds.), Handbook of Natural Computing, Springer, London, UK, 2012, pp. 1623–1655.

[21] A. El Sayed Auf, N. Dudek, Hexapod walking as emergent reaction to externally acting forces, in: Proceedings of Robotica - 9th Conference on Mobile Robots and Competitions, IEEE, Castelo Branco, Portugal, 2009, pp. 67–72.

[22] A. El Sayed Auf, F. Mösch, M. Litza, How the six-legged walking machine oscar handles leg amputations, in: Workshop on Bio-inspired Cooperative and Adaptive Behaviours in Robots, Springer, Rome, Italy, 2006, pp. 115–124.

[23] M. Erden, K. Leblebicioglu, Torque distribution in a six-legged robot, IEEE Transactions on Robotics 23 (2007) 179–186.

[24] Y. Fukuoka, H. Kimura, A. Cohen, Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts, The International Journal of Robotics Research 22 (2003) 187–202.

[25] D. Goldschmidt, F. Hesse, F. Wörgötter, P. Manoonpong, Biologically inspired reactive climbing behavior of hexapod robots, in: Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2012), IEEE, Vilamoura, Algarve, Portugal, 2012, pp. 4632–4637.

[26] M. Görner, G. Hirzinger, Analysis and evaluation of the stability of a biologically inspired, leg loss tolerant gait for six-and eight-legged walking robots, in: Proceedings of 2010 IEEE International Conference on Robotics and Automation (ICRA2010), IEEE, Anchorage, AK, 2010, pp. 4728–4735.

[27] D. Graham, The effect of amputation and leg restraint on the free walking coordination of the stick insect (carausius morosus), Journal of comparative physiology 116 (1977) 91–116.

[28] V. Granville, M. Krivanek, J.P. Rasson, Simulated annealing: a proof of convergence, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (1994) 652–656.

[29] R. Héliot, B. Espiau, Multisensor input for cpg-based sensory—motor coordination, IEEE Transactions on Robotics 24 (2008) 191–195.

[30] S. Hooper, Central pattern generators, Current Biology 10 (2000) R176–R179.

[31] G. Hornby, S. Takamura, T. Yamamoto, M. Fujita, Autonomous evolution of dynamic gaits with two quadruped robots, IEEE Transactions on Robotics 21 (2005) 402–410.

[32] A. Ijspeert, Central pattern generators for locomotion control in animals and robots: a review, Neural Networks 21 (2008) 642–653.

[33] A. Ijspeert, A. Crespi, D. Ryczko, J. Cabelguen, From swimming to walking with a salamander robot driven by a spinal cord model, Science 315 (2007) 1416–1420.

[34] S. Inagaki, H. Yuasa, T. Arai, Cpg model for autonomous decentralized multi-legged robot systemgeneration and transition of oscillation patterns and dynamics of oscillators, Robotics and Autonomous Systems 44 (2003) 171–179.

[35] S. Inagaki, H. Yuasa, T. Suzuki, T. Arai, Wave cpg model for autonomous decentralized multi-legged robot: Gait generation and walking speed control, Robotics and Autonomous Systems 54 (2006) 118–126.

[36] B. Jakimovski, B. Meyer, E. Maehle, Self-reconfiguring hexapod robot oscar using organically inspired approaches and innovative robot leg amputation mechanism, in: International Conference on Automation, Robotics and Control Systems (ARCS), Curran Associates Inc., Orlando, USA, 2009, pp. 62–69.

[37] A. Johnson, G. Haynes, D. Koditschek, Disturbance detection, identification, and recovery by gait transition in legged robots, in: Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2010), IEEE, Taipei, Taiwan, China, 2010, pp. 5347–5353.

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

[39] P. Manoonpong, U. Parlitz, F. Wörgötter, Neural control and adaptive neural forward models for insect-like, energy-efficient, and adaptable locomotion of walking machines, Frontiers in neural circuits 7 (2013) Online available.

[40] P. Manoonpong, F. Pasemann, H. Roth, Modular reactive neurocontrol for biologically-inspired walking machines, International Journal of Robotics Research 26 (2007) 301–331.

[41] P. Manoonpong, F. Pasemann, F. Wörgötter, Reactive neural control for phototaxis and obstacle avoidance behavior of walking machines, International Journal of Mechanical Systems Science and Engineering 1 (2007) 172–177.

[42] P. Manoonpong, F. Pasemann, F. Wörgötter, Sensor-driven neural control for omnidirectional locomotion and versatile reactive behaviors of walking machines, Robotics and Autonomous Systems 56 (2008) 265–288.

[43] F. Pasemann, Complex dynamics and the structure of small neural networks, Network: Computation in neural systems 13 (2002) 195–216.

[44] L.N. Patel, A.F. Murray, J. Hallam, Evolving multi-segment'super-lamprey' cpg's for increased swimming control, in: Proceedings of Symposium on Artificial Neural Networks (ESANN, pp. 461–466.

[45] G. Ren, W. Chen, S. Dasgupta, C. Kolodziejski, F. Wörgötter, P. Manoonpong, Fault tolerant locomotion of a hexapod walking robot under neural control based on multiple chaotic cpgs and learning, in: Conference Abstract: Bernstein Conference 2013, Tübingen, Germany. doi:10.12751/nncn.bc2013.0153.

[46] G. Ren, W. Chen, C. Kolodziejski, F. Wörgötter, S. Dasgupta, P. Manoonpong, Multiple chaotic central pattern generators for locomotion generation and leg damage compensation in a hexapod robot, in: Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2012), IEEE, Vilamoura, Algarve, Portugal, 2012, pp. 2756–2761.

[47] R.E. Ritzmann, R.D. Quinn, M.S. Fischer, Convergent evolution and locomotion through complex terrain by insects, vertebrates and robots, Arthropod Structure & Development 33 (2004) 361–379.

[48] S. Rossignol, Locomotion and its recovery after spinal injury, Current opinion in neurobiology 10 (2000) 708–716.

[49] M. Schilling, H. Cruse, P. Arena, Hexapod walking: an expansion to

walknet dealing with leg amputations and force oscillations, Biological cybernetics 96 (2007) 323–340.

[50] D. Spenneberg, K. McCullough, F. Kirchner, Stability of walking in a multilegged robot suffering leg loss, in: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), IEEE, New Orleans, LA, USA, 2004, pp. 2159–2164.

[51] S. Steingrube, M. Timme, F. Wörgötter, P. Manoonpong, Self-organized adaptation of a simple neural circuit enables complex robot behaviour, Nature Physics 6 (2010) 224–230.

[52] G. Taga, Y. Yamaguchi, H. Shimizu, Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment, Biological cybernetics 65 (1991) 147–159.

[53] V.K. Valsalam, R. Miikkulainen, Evolving symmetry for modular system design, IEEE Transactions on Evolutionary Computation 15 (2011) 368–386.

[54] D.M. Wilson, Insect walking, Annual Review of Entomology 11 (1966) 103–122.

[55] X. Xiong, F. Wörgötter, P. Manoonpong, A simplified variable admittance controller based on a virtual agonist-antagonist mechanism for robot joint control, in: Proceedings of 16th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR), AAMI, Sydney, Australia, 2013, pp. 989–996.

[56] J. Yang, J. Kim, Fault-tolerant locomotion of the hexapod robot, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 28 (1998) 109–116.

[57] L. Zhang, J. Hallam, J. Christensen-Dalsgaard, Effects of asymmetry and learning on phonotaxis in a robot based on the lizard auditory system, Adaptive Behavior 20 (2012) 159–171.