Project no.: **027657**

Project full title: **Perception, Action & Cognition through Learning of Object-Action Complexes**

Project Acronym: **PACO-PLUS**

Deliverable no.: D6.7

Title of the deliverable: Publication of neuronally motivated disturbance compensation learning methods on a multi-joint arm

Contractual Date of Delivery to the CEC:	**31 Jan 2009**
Actual Date of Delivery to the CEC:	**01 Dec. 2008**
Organisation name of lead contractor for this deliverable:	**CSIC**
Author(s):	**Wörgötter, F. Manoonpong, P. and Schröder-Schetelig, J**
Participants(s):	**BCCN**
Work package contributing to the deliverable:	**WP6**
Nature:	**R/D**
Version:	**1.0**
Total number of pages:	**37**
Start date of project:	**1st Feb. 2006** **Duration: 48 month**

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This deliverable contains two publications on the creation of two different adaptive (hence learnable) neural forward models for disturbance compensation. Different from the plan, these models have been developed for a multi-joint walking machine (RunBot) and not for an arm, but the principles and the learning methods are transferable also to multi-joint arms. This is due to the fact that we are using generally applicable neural network learning methods, which do not rely on the actual mechanical structures on which they are employed.

Keyword list: Adaptive neuronal forward models. Error back-propagation.

Publication of neuronally motivated disturbance compensation learning methods on a multi-joint arm

altered to

Publication(s) of neuronally motivated disturbance compensation learning methods on a biped walking robot.

Wörgötter F., Manoonpong, P. and Schröder-Schetelig, J.

Index

Introduction

This deliverable consists of two submitted papers (Manoonpong and Wörgötter, 2009; Schröder-Schetelig et al., 2009) which use neural methods to learn and employ forward models for disturbance compensation.

We have chosen our dynamic walking robot RunBot instead of (the kinematically controlled) ARMAR to implement disturbance compensation because this problem is more difficult for dynamic than for kinematic machines. Furthermore, there are currently no neuronal control mechanisms implemented on ARMAR and its control relies so far still on conventional controllers.

Thus, this deliverable and the two submitted papers have to be considered as a contribution to basic research on neuronal control and learning with possible future use on ARMAR.

Future use of the powerful techniques shown in this deliverable will rely on a decision to implement some neuronal control methods on ARMAR, too.

Background: It has long been known that ego-motion will always lead to the (self-) stimulation an animal's sensor system. Normally, however, such self-induced stimuli will not be experienced and compensation mechanisms lead to stable percepts. This is especially pronounced for our vision and/or vestibular system, which receive continuously changing stimulation when we walk. In spite of this our world is perceived as stable and unmoving. Hence, some "error-correction" mechanisms are implemented in our brain to remove unwanted signals from self-stimulations. Powerful nausea can arise from the disturbance of such error-correction systems (often from direct sensor impairments like the blocking of a semi-circular canal). The mild form of travel sickness is associated to a mismatch between a stable visual percept and the conflicting vestibular perception of movement also upsetting the error-correction by the assumption of a wrong "set-point".

Novel Contribution: The idea roots back to von Holst and Mittelstaedt (1950), who proposed that motor commands copied within the central nervous system (efference copy) help to distinguish 'reafference' activity (afference activity due to self-generated motion) from 'exafference' activity (afference activity due to external stimulus). Based on biological findings, paper 1 shows two experimental studies using "RunBot" where such principles together with neural forward models are applied to RunBot's dynamic locomotion control. The main purpose of this paper is to discuss how the inherent dynamic properties of the different modules lead to the required signal processing. As a result, the first experiment shows that an efference copy can be applied to eliminate external and self-generated sensory noise. In the second experiment, we demonstrate that the robot can determine terrain condition changes through efference copies; i.e., it can detect a slope by the deviation of its own gait from the normal gait-pattern observed on flat ground.

In the second paper we present an application of the principle of efferent copies together with network learning use an error back-propagation algorithm in a small network to compensate for self-generated acceleration during walking. The difference to the actually measured acceleration is then used to stabilize the walking on terrains with changing slopes. While the idea is similar to the one of paper 1, the second paper uses an entirely different set of network methods and is – different from paper 1 – also able to be trained to the default situation (flat terrain) and then use the learned default for comparison with the actual situation for error calculating and compensation.

Appendix: Paper 1: ***PoramateManoonpong and FlorentinWörgötter (2009) Efference Copies in Neural Control of Dynamic Biped Walking System structure, submitted to RAS.***

Appendix: Paper 2: ***Johannes Schröder-Schetelig, Poramate Manoonpong and FlorentinWörgötter (2009) Using Efference Copy and Forward Internal Model for Adaptive Biped Walking, submitted to AUTONOMOUS ROBOTS.***

Efference Copies in Neural Control of Dynamic Biped Walking

P. Manoonpong [a], F. Wörgötter [a,*]

[a] *Bernstein Center for Computational Neuroscience (BCCN), University of Göttingen, D-37073 Göttingen, Germany*

Abstract

In the early 1950s, von Holst and Mittelstaedt proposed that motor commands copied within the central nervous system (efference copy) help to distinguish 'reafference' activity (afference activity due to self-generated motion) from 'exafference' activity (afference activity due to external stimulus). In addition, an efference copy can be also used to compare it with the actual sensory feedback in order to suppress self-generated sensations. Based on these biological findings, we conduct here two experimental studies on our biped "RunBot" where such principles together with neural forward models are applied to RunBot's dynamic locomotion control. The main purpose of this article is to present the modular design of RunBot's control architecture and discuss how the inherent dynamic properties of the different modules lead to the required signal processing. We believe that the experimental studies pursued here will sharpen our understanding of how the efference copies influence dynamic locomotion control to the benefit of modern neural control strategies in robots.

Key words: Legged robots; Recurrent neural network; Adaptive walking; Dynamic walking; Internal model

1 Introduction

Neural networks have become a versatile tool in many application like pattern recognition, function approximation and others. Until recently networks, however, have not been used so often for the control of machinery (e.g., robots).

* Corresponding author. Tel.: +49 (0) 551 5176-528; fax: +49 (0) 551 5176-449.
Email addresses: poramate@nld.ds.mpg.de (P. Manoonpong), worgott@nld.ds.mpg.de (F. Wörgötter).

The difficulty in relaying network output to the end-effectors in a coordinated way and the complex structure of motor control networks may explain why they are still not much used for solving complex motor control problems so far. Recently, a few studies suggested, however, that small networks can be very powerful for addressing such problems. The works of Ijspeert et al. (2007) [12], Bem et al. (2003) [1], and Meyer et al. (2003) [24] have shown that in robots complex movement patterns like swimming and walking can be controlled and coordinated by neural network activity. The employed machines (lamprey or salamander like robots) are this way able to produce undulatory movements navigating through their environment. In our own studies, we have used an adaptive neural network to control a dynamic biped robot, called "RunBot" [7], [21]. This machine is able to walk and learn to adapt its posture and gait parameters to different terrains, e.g., when walking up a slope.

While this shows the power of network control, at least one important problem has so far not been addressed: All moving system are - on the sensor side - faced with noise. This could be random noise from various sources in the environment (external noise) but also disturbances which are introduced by the ego-motion (internal noise). For example, every step stimulates our own vestibular system in an unwanted way. Both noise sources mask other more relevant stimulus events and lead to reduced performance of the sensor system.

The brains of animals have developed strategies to compensate for these noise sources and the goal of this study is to show that these strategies can also be copied efficiently into robots allowing the machines to ignore external as well as internal noise. More than that: As internal noise is repetitive while walking it is predictable. This leads to the situation that the robot can recognize the disturbance. The comparison between the expected internal noise and the one actually measured can be used as an error signal which drives network learning as will be shown below. To this end we will employ the idea of "efferent copies".

Around the mid-19th century, von Holst and Mittelstaedt (1950) [9] demonstrated in animal models that motor commands are copied within the central nervous system (CNS). These copies help to distinguish 'reafference' (afference activity due to self-generated motion) from 'exafference' (afference activity due to changes in the external world). They can be also used for comparison with the actual sensory feedback in order to subtract self-generated sensations for maintaining stable perception. Similarly, Sperry (1950) [30] presented evidence which supported this idea. He showed that sensory areas receive discharge patterns (efference copy) with respect to the *expected* sensory feedback. In the early 1960s, Held (1961) [8] indicated that efference copies and the reafference generated by self-motion cannot be directly compared due to the different dimensionality between motor commands and sensory feedback. Therefore, he proposed a neural structure that transforms an efference copy signal into an expected sensory input to be able to compare it to the actual incoming sensory

signal. This neural transformation mechanism is known as "internal model" [35]. As described by Kawato (1999) [17], internal models or internal loops of biological systems are classified into three types: Inverse internal model (the system calculates a motor command from a desired trajectory/state information), forward internal model (the system predicts sensory consequences from efference copies), and integrated internal model (the system integrates both inverse and forward models).

Based on the biological findings described above, several robot experiments have been performed applying efference copy and internal model concepts employing these ideas for arm control [25], visuo-acoustic coordination [29] as well as leg control [5],[19] (see the Discussion section for details). These studies show that the efference copy principle together with an appropriate internal model can be successfully applied to a wide range of robot control problems. The work presented here extends this line of research to problems in dynamic walking control. In the present study we conduct two experiments on "RunBot": 1) The first experiment shows that an efference copy can be applied to eliminate external and self-generated sensory noise. Normally such perturbations destabilize the activation parameters for the gait and cause unstable walking. This can be successfully avoided by using an efference copy signal. 2) In the second experiment, we demonstrate that the robot can detect a slope by the deviation of its own gait from the normal gait-pattern observed on flat ground. This deviation signal can be used for learning the new parameter set, applicable to slope-walking.

The employed networks in general consists of three components: A network for basic walking, a learning control network as well as an efferent copy and internal model building network. In total this leads to a somewhat higher complexity of the network structure. As compared to the original control network of RunBot [7] the three modules, however, can be understood one by one, which makes network design simple. Thus, the main purpose of this article is not only to present the applications of the efference copy for dynamic locomotion control. In addition to this, some emphasis is put on the modular design and the aspect how the inherent dynamic properties of the different modules lead to the required signal processing.

In the following section, we give a general overview of the RunBot system. Afterwards biomechanics and adaptive reflex neural locomotion control forming the system's basic behavior are presented in brief where the complete descriptions can be found in our previous publications [7], [21]. Sections 3 and 4 show experimental studies for the application of the efference copy for improving locomotion control and determining terrain condition changes which is the main contribution of the article. Discussion and conclusions are provided in Sections 5.

2 RunBot system

The RunBot system (see Fig. 1) [21] uses the design principle of multiple nested loops to couple its biomechanics with adaptive reflex neural locomotion control through an environment. Employing this hierarchical architecture, RunBot exhibits the self-stabilizing and passive properties [7] reflected by its biomechanics. It can stably walk with different speeds regulated through its reflexive neural control [7]. Furthermore, it can adapt its gaits to different terrains by means of a neural learning process using adaptive neural control [21]. An overview of biomechanics and adaptive reflex neural locomotion control are provided in the following, for more details see [21].

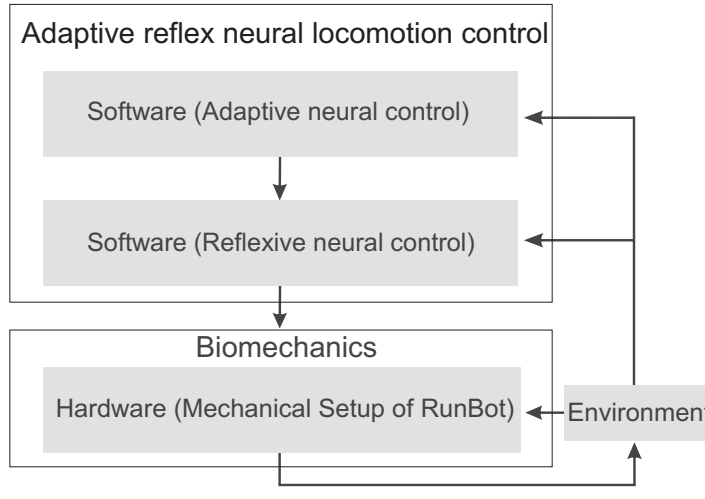

Fig. 1. The RunBot system. It is divided into three levels (Biomechanics, Reflexive neural control, and Adaptive neural control) organized as a hierarchical structure and coupled via the environment.

2.1 Biomechanics

RunBot is a planar dynamic biped robot (see Fig. 2). It consists of four actuated joints: left hip, right hip, left knee and right knee. Each joint is driven by a modified servo motor where the built-in Pulse Width Modulation (PWM) control circuit is disconnected, while its built-in potentiometer is used to measure the joint angles (S). RunBot has no actuated ankle joints, resulting in very light feet and efficiency for fast walking. Its feet were designed having a small circular form (4.5 cm long). Each foot is equipped with a ground contact sensor (G). A mechanical stopper is implemented on each knee joint to prevent it from going into hyperextension. Approximately seventy percent of the robot's weight is concentrated on its trunk and the parts of the trunk are assembled in a way that its center of mass is located forward of the hip axis.

In addition, it has an upper body component (UBC), which can be actively moved to shift the center of mass backward or forward for walking on different terrains, e.g., level floor versus up or down a ramp. It leans backwards during walking on a level floor (see Fig. 2b) and this position is also suitable for walking down a ramp [22]. On the contrary, it will lean forwards (reflex action) when RunBot falls backwards or after it successfully learned to walk up a ramp (see Fig. 2b). The corresponding reflex is controlled by an accelerometer sensor (AS) functioning as its simple vestibular system. The AS is installed on top of the right hip joint. In addition, one infrared (IR) sensor is implemented at the front part of RunBot pointing downwards to detect a ramp. Here, the IR sensor serves as a simple vision system, which can distinguish between a level floor with black color and a painted ramp with white color (see Fig. 2b). This sensory signal is used for adaptive control. All sensory and motor signals are converted through a AD/DA converter board (USB-DUX [1]) with the update frequency of 250 Hz.

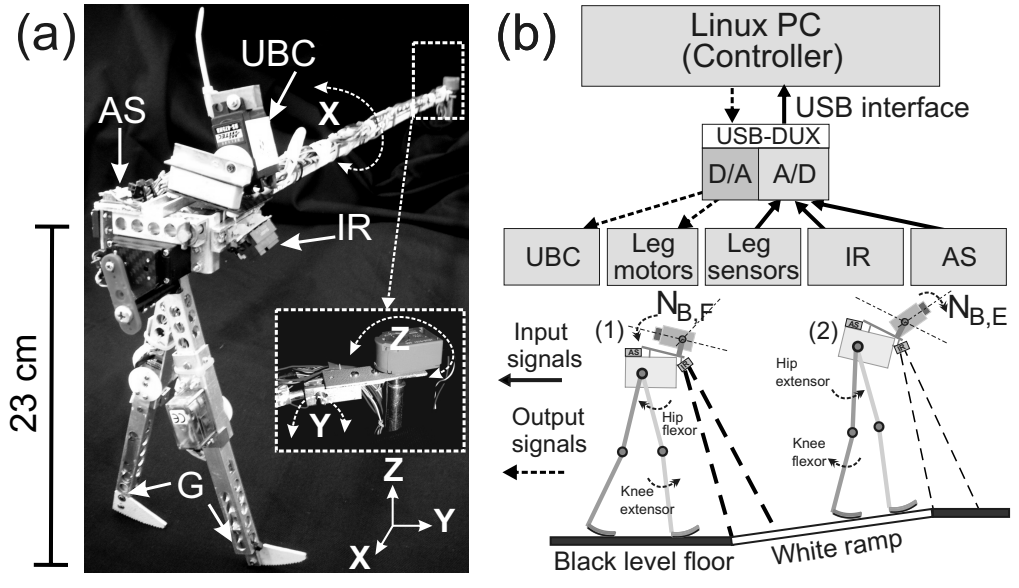

Fig. 2. (a) The planar dynamic robot RunBot. UBC, upper body component; IR, infrared sensor; AS, accelerometer sensor; G, ground contact sensor. (b) Schematic set-up of the RunBot system. Leg sensors consist of joint angle and ground contact switch sensors, leg motors are the motors of the left and right hip and knee joints. The detection range of the IR sensor for slope sensing is shown in the lower figure where the thick dashed ray of the IR sensor (1) indicates that the sensor gives a high output signal while the thin dashed ray (2) means a low signal. Hence the sensor responds more strongly to the white color. $N_{B,F}$, body flexor (leaning backwards); $N_{B,E}$, body extensor (leaning forwards).

We constrain RunBot in the sagittal plane by a boom of one meter length. RunBot is attached to the boom via a freely rotating joint in the x-axis, while

[1] http://www.linux-usb-daq.co.uk.

the boom is attached to the central column with freely rotating joints in the y and z axes (see Fig. 2a). The mechanical design of RunBot has the following special features that distinguish it from other powered biped robots and that facilitate high-speed walking and exploitation of natural dynamics: (a) small, curved feet allowing for rolling action; (b) unactuated, hence light, ankles; (c) lightweight structure; (d) light and fast motors; (e) proper mass distribution of the limbs; and (f) properly positioned mass center of the trunk. Utilizing all these properties, RunBot can perform self-stabilization of gaits and it also exhibits passive walking characteristics reflected by the fact that during one quarter of its step cycle all motor voltages remain zero [7].

2.2 Neural locomotion control

The neural locomotion control (see Fig. 1) consists of two main structures: the adaptive and reflexive neural control circuits. All neurons in the circuits are modeled as rate-coded neurons with the standard sigmoid transfer function. They are simulated on a Linux PC with an update frequency of 250 Hz.

2.2.1 Reflexive neural control

The reflexive neural control is based on several reflex mechanisms. It is composed of two submodules. One is for leg control and the other is for UBC control. Both leg and UBC controls are independent but they are indirectly coupled through the biomechanics of RunBot (see Figs. 2 and 3).

The leg control, simulated as mono-synaptic connections, contains motor neurons (N), which are linear and can send their signals unmodified to the motors (M) (see Figs. 3 and 4). There are several local sensor neurons (proprioceptor), which, by their conjoint reflex-like actions, trigger different gaits, e.g., slow and fast. These local sensor neurons can be classified into three loops: joint control (Local 1, see Fig. 3), intra-joint control (Local 2, see Fig. 3) and leg control (Local 3, see Fig. 3). Joint control arises from angle sensors S at each joint, which measure the joint angle and influence only their target motor neurons. Intra-joint control is achieved from sensors A, which measure the anterior extreme angle (AEA) at the hip and trigger an extensor reflex at the corresponding knee. Leg control comes from ground contact sensors G, which drive the motor neurons of all joints.

The UBC control represents a long-loop reflex, which is indirectly modulated by its AS through the adaptive neural control network (see Fig. 3). In general situations like when walking on flat terrain, the AS is inactive and the flexor body motor neuron $N_{B,F}$ is activated to lean the body backwards (see Fig. 2b) while the extensor motor neuron $N_{B,E}$ is inhibited. This situation is reverted

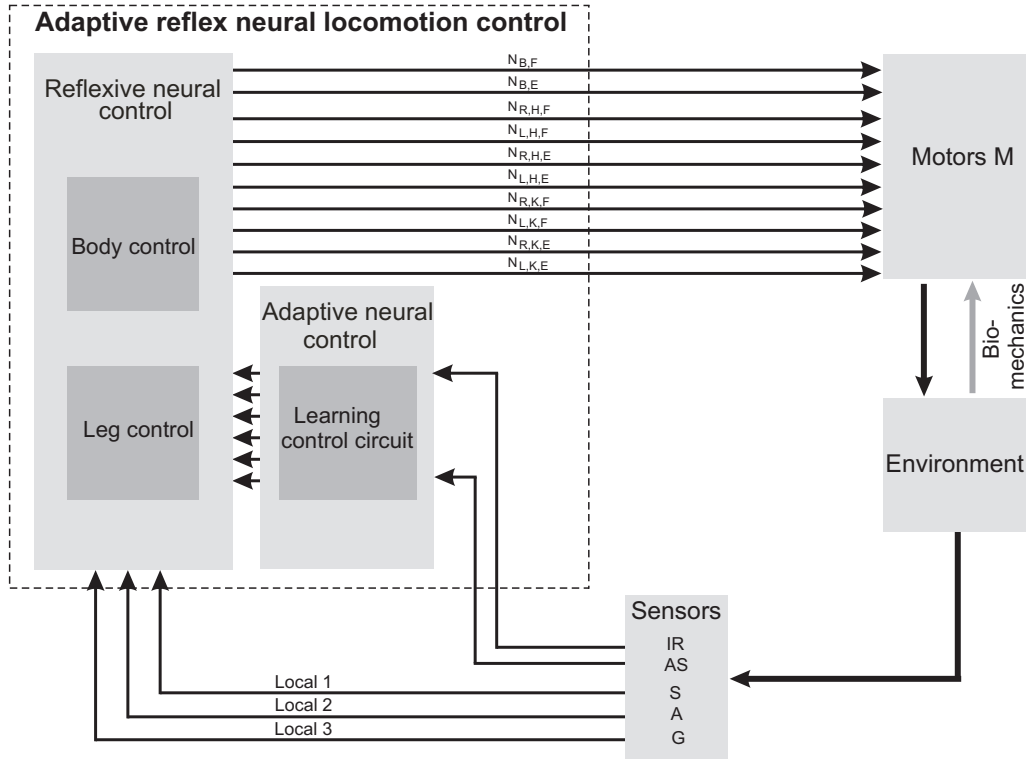

Fig. 3. Neural locomotion control (see text). Reflexive walking behavior arises from the interaction of three local sensorimotor loops (reflexive neural control) together with the passive properties (biomechanics). Additionally, adaptation is achieved by a learning mechanism. A gray arrow represents RunBot's physical embodiment eliciting its passive dynamic walking properties. IR, infrared sensor; AS, accelerometer sensor; S, joint angle sensor of hips and knees; A, stretch receptor for anterior extreme angle (AEA) of the hips; G, ground contact sensor; $N_{B,F}$, flexor body-motor signal; $N_{B,E}$, extensor body-motor signal; $N_{R,H,F}$, flexor leg-motor signal of the right hip; $N_{L,H,F}$, flexor leg-motor signal of the left hip; $N_{R,H,E}$, extensor leg-motor signal of the right hip; $N_{L,H,E}$, extensor leg-motor signal of the left hip; $N_{R,K,F}$, flexor leg-motor signal of the right knee; $N_{L,K,F}$, flexor leg-motor signal of the left knee; $N_{R,K,E}$, extensor leg-motor signal of the right knee; $N_{L,K,E}$, extensor leg-motor signal of the left knee. In general, indices are omitted below the last relevant level, e.g., N_R applies to flexor and extensor of the hip and knee of the right leg.

when a strong signal from the AS exists, which happens only when RunBot falls backwards, e.g., RunBot tries to walk up a ramp. This will trigger a leaning reflex of the UBC. More detailed descriptions of all neuron models together with the neural network structures and the discussion of their parameters can be found in [21].

2.2.2 Adaptive neural control

RunBot's task was to learn walking up a ramp and then continue again on a level floor. The learning goal is to avoid the leaning reflex and thereby learn

to also change gait parameters in an appropriate way to prevent RunBot from falling. We use adaptive neural control to change the leaning action of the UBC by learning and to also influence several other leg control parameters for gait adaptation. This is accomplished by using six learner neurons changing activation parameters of their target neurons (see Fig. 4). Our learning algorithm (described in details later) applies a correlation based differential Hebbian learning rule [27] where the modification of all those parameters will be controlled by two kinds of input signals: one is an early input (called predictive signal) and the other is a later input (called reflex signal). In general, we use the IR signal as a predictive signal while the AS signal serves as a reflex signal (see Fig. 4). At the beginning, the connections between the predictive signal and learner neurons converge with zero strengths (dashed arrows in Fig. 4). In this situation, parameters of the target neurons will be altered only by the reflex signal (solid arrows between the reflex signal and learner neurons in Fig. 4); i.e., the leaning reflex of the UBC together with the gait adaptation will be triggered by the AS signal as soon as RunBot falls. Hence, RunBot will begin to walk up the ramp with a wrong set of gait parameters and an inappropriate posture of the UBC. Thus, it will eventually fall leading to a signal at the AS, which will change RunBot's parameters but too late (when it already lies on the ground). Due to learning the modifiable synapses (ρ_1^1,...,ρ_1^6, dashed arrows in Fig. 4), which connect the predictive IR signal with the learner neurons ($\mathrm{L}_1, ..., \mathrm{L}_6$), will grow. Consequently, after 3-5 falls during the learning phase, gait adaptation together with posture control of the UBC will finally be driven by the predictive IR signal instead. Correspondingly, RunBot will adapt its gait together with leaning the UBC in time. The used learning algorithm has the property that learning will stop when the reflex signal is zero [27]; i.e., when RunBot does not fall anymore. On returning to flat terrain, the IR output will get small again and RunBot will change its locomotion and posture back to normal for walking on a level floor.

Learning algorithm: In general, each learner neuron $\mathrm{L_n}$ requires two input signals (u_0, u_1) with synaptic weights (ρ_0, ρ_1) (see solid frame in Fig. 4). Here, we use the AS and the IR signals as u_0 and u_1, respectively. Only ρ_1 (dashed arrows in Fig. 4) is allowed to change through plasticity while ρ_0 (solid arrows connecting the AS neuron with learner neurons in Fig. 4) is set to a positive value, i.e., $\rho_0 = 1.0$. The output activity v of $\mathrm{L_n}$ and the learning rule for the weight change ρ_1^n are given by:

$$v(L_n) = \rho_0^n u_0 + \rho_1^n u_1, \quad n = 1, \ldots, 6, \tag{1}$$

$$\frac{d\rho_1^n}{dt} = \mu_n u_1 \frac{du_0}{dt}, \quad n = 1, \ldots, 6; \tag{2}$$

where we here use only input signals and correlate them with each other [27]. μ_n is the learning rate. It is independently set for each learner neuron,

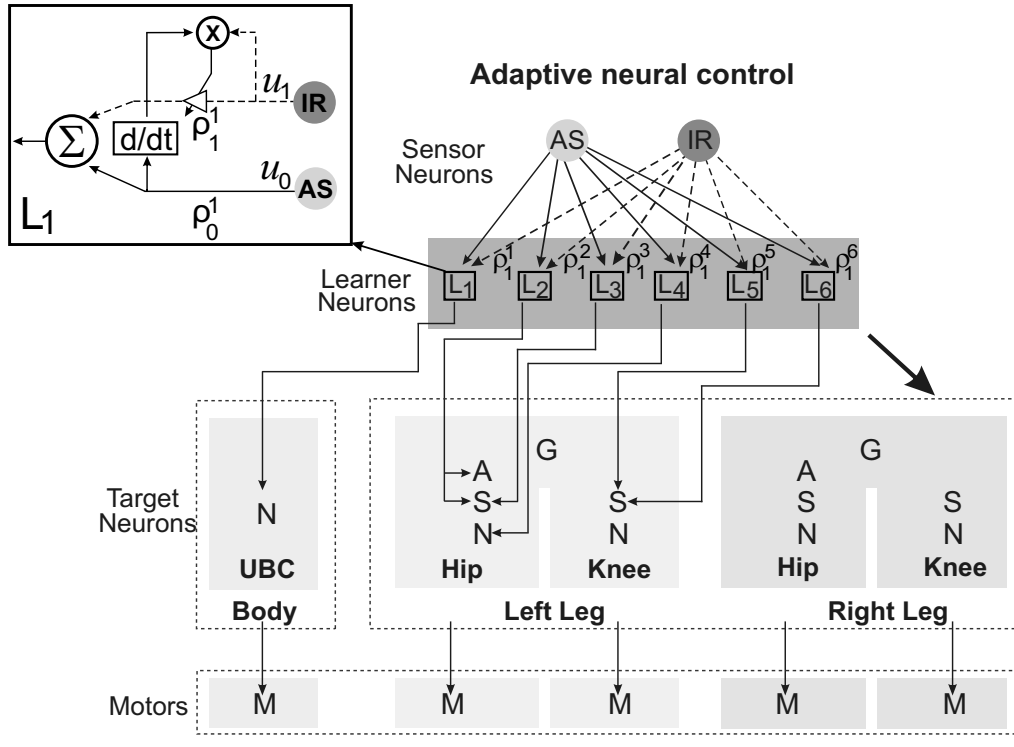

Fig. 4. Adaptive neural control where the neural learning mechanism is shown in the solid frame (top left, L_1, see text for details). Note that all learner neurons have the same learning mechanism. Connections between learner neurons and target neurons of the right leg, which are identical to those of the left leg, are not shown. ρ_1^1,...,ρ_1^6 are synaptic weights connecting the predictive IR signal with the learner neurons ($L_1, ..., L_6$).

which defines the desired equilibrium point and how fast the system can learn. In neurons with multiple inputs such a mechanism can be used to modify the synaptic strengths according to the order of the arriving inputs. As a consequence, the predictive input will get strengthened if the predictive signal u_1 is followed by the reflex input u_0, where the reflex drives the neuron into firing. This rule will lead to weight stabilization as soon as $u_0 = 0$ [27], hence, when the reflex has successfully been avoided. As a result, we obtain behavioral and synaptic stability at the same time without any additional weight-control mechanisms.

All in all, through the tight coupling of the biomechanics with the adaptive reflex neural locomotion control, RunBot can autonomously walk with a high speed (> 3.0 leg length/s), self-adapting to minor disturbances, and reacting in a robust way to abruptly induced gait changes. At the same time, it can learn walking on different terrains, requiring only few learning experiences. All these experimental results have been presented in [21].

3 Experiment 1: Efference copy for external and self-generated sensory noise cancellation

As described above, RunBot uses IR (infrared eye) and AS (vestibular) information for posture and gait adaptation during walking up a painted slope. Due to the IR sensor characteristic, the sensor responds more strongly to the white color (see Fig. 2b). Thus, in our first experimental study on the application of the efference copy for locomotion control, the walking path of RunBot is modified by adding white spots on its black level tracks (compare Figs. 2b and 5a) in order to simulate disturbances to the IR sensor. As a consequence, the IR sensor gives unwanted periodic noise (see Fig. 5b, gray areas). In addition, RunBot's egomotion causes the AS to produce self-generated sensory events (see Fig. 5c, gray areas). These periodic perturbations will destabilize the activation parameters for the gait and lead to a wrong set of gait parameters as well as an inappropriate posture of the UBC. In other words, after a few learning experiences for walking up the slope, RunBot will perform upslope gait with leaning its UBC forwards during walking on level floors (location (1) or (3) shown in Fig. 5a). As a consequence, it will fall forwards before approaching a slope or after leaving it (see Sect. 3.2 for experimental results).

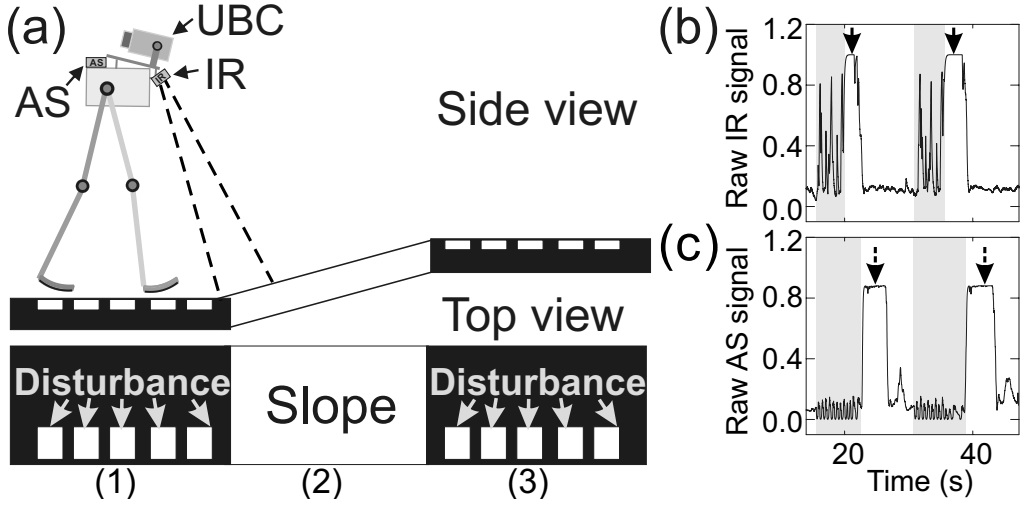

Fig. 5. (a) Walking path on which white spots are added at positions (1) and (3). They lead to a disturbance of the IR sensor. Note that the spots are empirically placed in the way that the IR sensor generates the unwanted noise every second step; i.e., periodic noise. Adding more spots (high density), the sensors will give continuous noise which makes the system impossible to discern between a slope detection signal and this continuous noise. (b, c) Raw sensor signals. Solid arrows in (b) depict the situation where RunBot detects a slope and dashed arrows in (c) where RunBot falls backwards. It falls over backwards, as it has not yet learned to react to its IR input with a change in gait.

To solve such problems, we need to filter the unwanted noise. By doing so, we copy the periodic motor signals, transform them into noise expectation

through so-called neural forward models (see Fig. 6). These expected sensory noise signals are fed into compensator units (see Fig. 6) to subtract the unwanted noise from the actual sensory feedback. Finally, we use neural post-processing units (see Fig. 6) to smooth and shape the compensated signals of the IR and AS sensors in order to obtain appropriate correlations for the learning mechanism. The details of this noise cancellation process is described in the following.

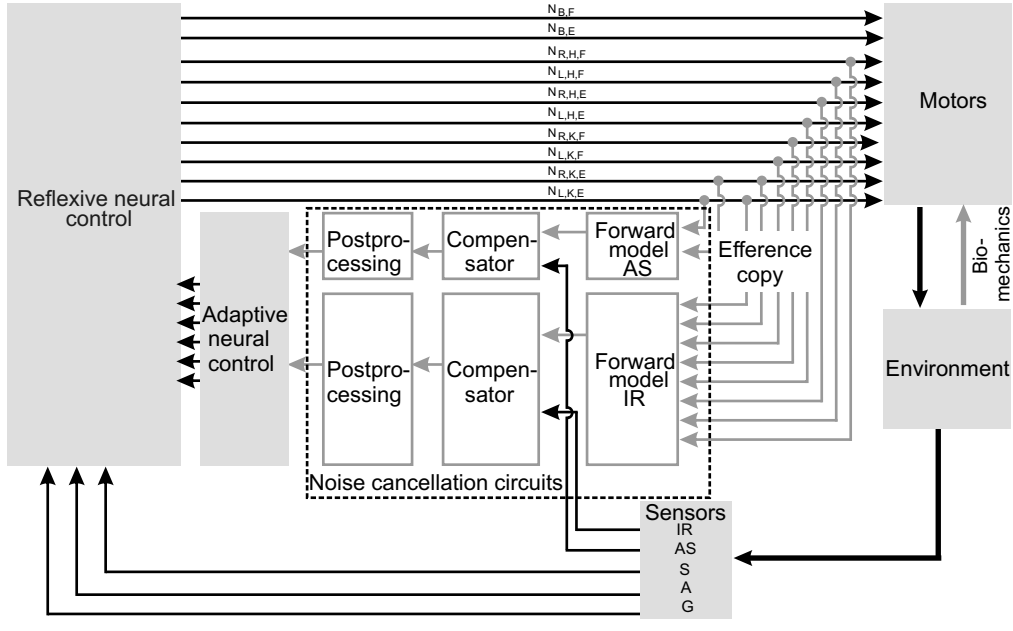

Fig. 6. Adaptive reflex neural locomotion control with external and self-generated sensory noise cancellation circuits. It applies an efference copy for eliminating sensory noise (compare Fig. 7).

3.1 Modeling noise cancellation circuits

To filter the unwanted noise of the IR signal, we copy all extensor and flexor motor signals (N_R, N_L, efference copy) of the leg joints (see Fig. 6). These motor signals are then transformed into a noise expectation through the neural forward model (forward model IR, see Figs. 6 and 7). This forward model is manually constructed as a series of 12 hysteresis elements (see Fig. 7). Apart from filtering the noise it shapes the motor signals to match to the noise of the IR signal for subtraction afterwards. We create a hysteresis by using a single neural unit with a "supercritical" self-connection ($w_{Self} > 4$) [26], modeled as a discrete-time, rate-coded neuron with activity that develops according to:

$$a_i(t+1) = \sum_{j=1}^{n} W_{ij}\sigma(a_j(t)) + \Theta_i \quad i = 1, \ldots, n \tag{3}$$

where n denotes the number of units, a_i their activities, Θ_i represents a fixed internal bias term together with a stationary input to neuron i, and W_{ij} the synaptic strength of the connection from neuron j to neuron i. The output of the neurons is given by the standard sigmoid $\sigma(a_i) = (1 + e^{-a_i})^{-1}$. Input units are linearly mapped onto the interval [0, 1].

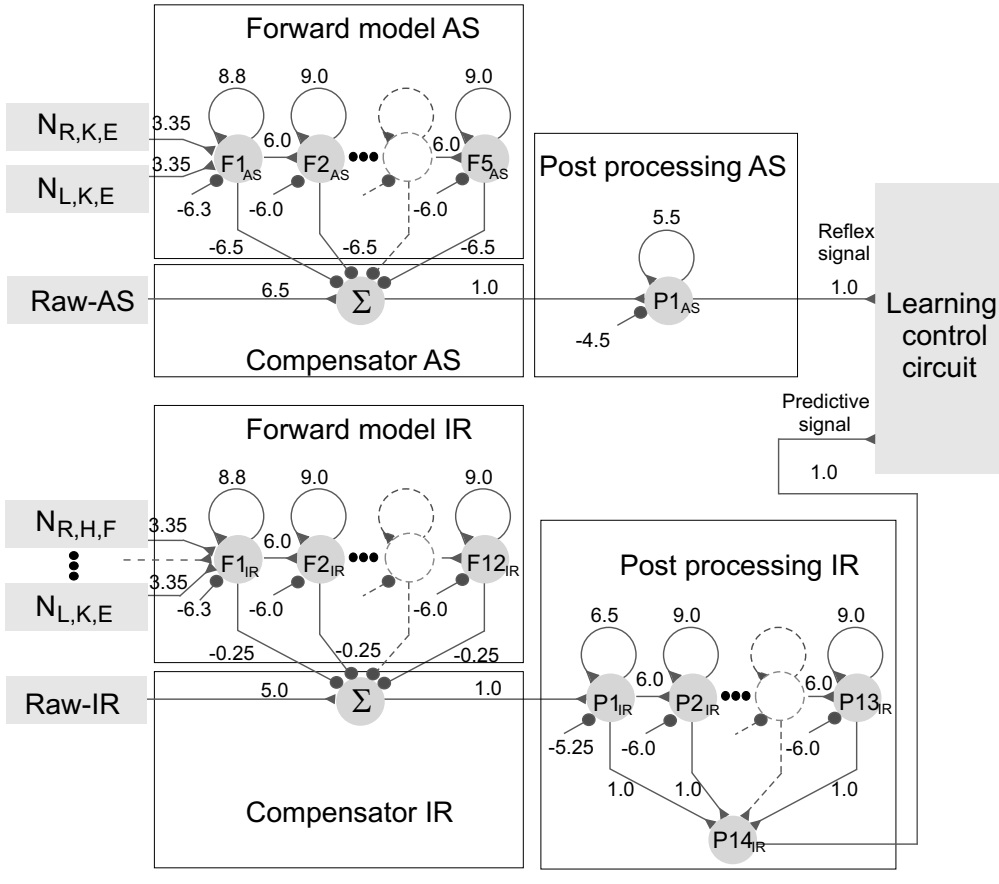

Fig. 7. Noise cancellation circuits of IR and AS signals. Each circuit is composed of three subunits: forward model, compensator, and postprocessing units. N_R and N_L indicate motor signals of leg joints (efference copy). Note that one can optimize these noise cancellation circuits, for instance by using an evolutionary algorithm [11], but for the purposes here, manual adjustment was sufficient.

The neural parameters of the forward model network were empirically adjusted as follows. First we combined all motor signals at the first recurrent neuron $F1_{IR}$ (see Fig. 7) and then we adjusted the combined motor signals such that they will cross forward and backward through the hysteresis domain [20], [26] for mainly filtering the noise of the motor signals. Hence, we set the synaptic weight, connecting between all motor signals and the recurrent neuron $F1_{IR}$, to a positive value, i.e., 3.35, to amplify the signals. Afterwards, we shifted the amplified signals by a negative bias term, i.e., -6.3. Consequently, the modified signals sweep over the input interval between -6.3 and -2.95. Finally, we tuned the self-connection weight of the neuron to derive a reasonable hysteresis

interval (see Fig. 8a) on the input space; i.e., 8.8. This hysteresis effect allows the output to show high (≈ 1.0) and low (≈ 0.0) activations at different points (see Fig. 8a). By utilizing this feature, the recurrent hysteresis neuron $\mathrm{F1_{IR}}$ acts as a low pass filter, which can eliminate unwanted motor noise (see Fig. 8).

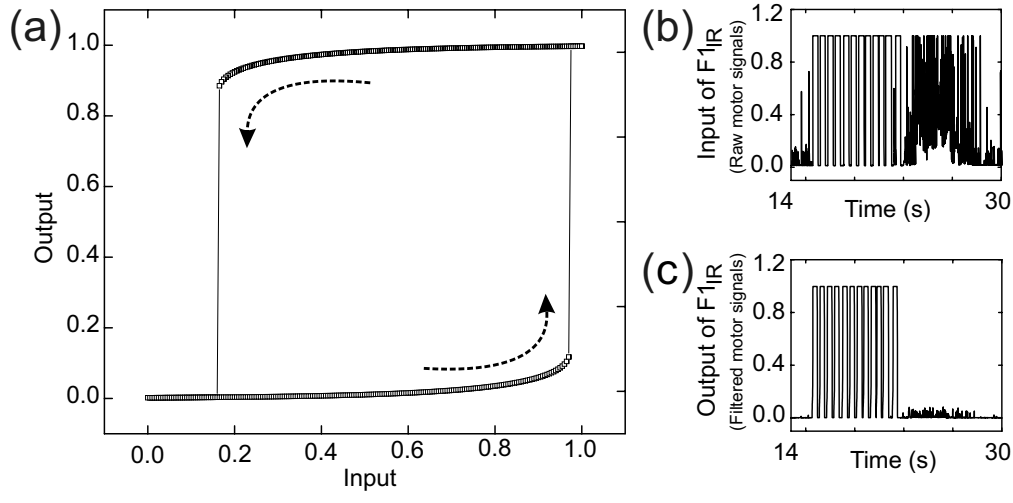

Fig. 8. (a) Hysteresis effect between the input and output of the recurrent neurons $\mathrm{F1_{IR}}$ and $\mathrm{F1_{AS}}$ (see Fig. 7). The input varies between 0.0 and 1.0 while the output shows high activation when the input increases to values above 0.97. On the other hand, it will show low activation when the input decreases below 0.165. Utilizing this hysteresis property, high frequency motor noise is eliminated. In other words, these recurrent neurons $\mathrm{F1_{IR}}$ and $\mathrm{F1_{AS}}$ act as a low pass filter. (b) Raw motor signals. (c) Filtered motor signals after passing through the recurrent neuron $\mathrm{F1_{IR}}$. Note that the raw and filtered motor signals at the recurrent neuron $\mathrm{F1_{AS}}$, having similar patterns to those of (b) and (c), are not shown.

After that, the output of the recurrent neuron $\mathrm{F1_{IR}}$ is provided to a series of single recurrent neurons $\mathrm{F2_{IR}}, ..., \mathrm{F12_{IR}}$ (see Fig. 7). The structure of each single recurrent neuron was configured in the same manner as the recurrent neuron $\mathrm{F1_{IR}}$ but the neural parameters were set differently. We chose them in the way that they provide the hysteresis effect (see Fig. 9a) that will shape the filtered motor signals to match the periodic noise of the IR signal. As a result, the connection weight between neurons, the bias term, and the self-connection weight are set as 6.0, -6.0, and 9.0, respectively. Eventually, the output of each recurrent neuron ($\mathrm{F1_{IR}}, ..., \mathrm{F12_{IR}}$) is transmitted to subtract the unwanted noise of the actual IR sensory feedback at a compensator unit (compensator-IR, see Fig. 7) through a connection weight set to -0.25. The compensator unit is simply modeled as a standard additive neuron with a linear transfer function. Subsequently, the compensator output is postprocessed at another series of recurrent neurons. All neural parameters of this postprocessing unit (postprocessing-IR, see Fig. 7) are set to similar values as $\mathrm{F2_{IR}}, ..., \mathrm{F12_{IR}}$, described above, except the first unit ($\mathrm{P1_{IR}}$). Its neural parameters are given as: connection weight from the compensator unit IR to this

first neurons = 1.0, bias term = −5.25, and the self-connection weight = 6.5. This postprocessing unit will smooth the signal at the recurrent neuron $P1_{IR}$ and through the remaining recurrent neurons $P2_{IR}, ..., P13_{IR}$ it will derive the appropriate correlation with the reflex signal for our learning mechanism. The final output of each postprocessing neuron is then summed up at the neuron $P14_{IR}$ before applying to the learning circuit as a predictive signal.

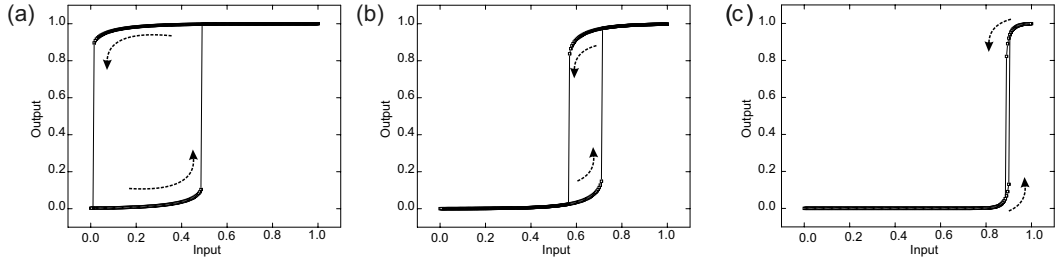

Fig. 9. Hysteresis diagrams of different recurrent neural parameters. All hystereses have an input which varies between 0.0 and 1.0 while their output shows low and high activations at different points. (a) Hysteresis loop of the recurrent neurons $F2_{IR}, ..., F12_{IR}$, $F2_{AS}, ..., F5_{AS}$, and $P2_{IR}, ..., P13_{IR}$ (see Fig. 7). The output shows high activation when the input increases to values above 0.49 while it will show low activation when the input decreases below 0.015. We use this hysteresis property for prolonging the activation time to obtain appropriate correlation of the signals, i.e., matching between the unwanted sensory noise and motor signals as well as correlating between the reflex (noise free AS) and predictive (noise free IR) signals. (b) Hysteresis loop of the recurrent neuron $P1_{IR}$ (see Fig. 7). The output shows high activation when the input increases to values above 0.71. On the other hand, it will show low activation when the input decreases below 0.57. This hysteresis effect is applied to smooth the output of the compensator-IR. (c) Small hysteresis loop of the recurrent neuron $P1_{AS}$ (see Fig. 7). Its output gives high activation when the input increases to values above 0.9 and it will show low activation when the input decreases below 0.88. This hysteresis effect is for smoothing the output of the compensator-AS.

So far we have discussed the filtering process of the IR signal. To cancel self-generated noise at the AS signal, we use the same technique as above but here only the extensor knee-motor signals of the left ($N_{L,K,E}$) and right ($N_{R,K,E}$) legs are copied. Then they are transmitted to the neural forward model (forward model AS, see Figs. 6 and 7). Here the forward model AS consists of five recurrent neurons $F1_{AS}, ..., F5_{AS}$ (see Fig. 7). They are configured similar to the ones of the forward model IR. As a consequence, they lead to the same hysteresis phenomena (see Figs. 8a and 9a) which are used to filter motor noise and also transform the motor signals into the expected sensory noise signal in order to subtract the unwanted noise from the actual AS sensory feedback (self-generated sensation). The subtraction is done in the compensator unit (compensator-AS) where the output of each recurrent neuron is amplified and sent to this compensator-AS by means of a connection weight of −6.5. Note that the compensator-AS is modeled similar to the compensator-

IR. Finally, the compensator output is shaped at the recurrent neuron $P1_{AS}$ (postprocessing-AS, see Fig. 7) before applying to the learning circuit as a reflex signal. The neural parameters of this postprocessing unit are set as: connection weight from the compensator-AS to its postprocessing neuron = 1.0, bias term = -4.5, and self-connection weight = 5.5. With these neural parameters, this postprocessing unit shows an appropriate hysteresis loop (see Fig. 9c) for refining the signal and providing proper correlation with the predictive signal for our learning mechanism.

3.2 Results

Figure 10 shows experimental results where the noise cancellation circuits (see Fig. 6) described above are employed. As a consequence, the external and self-generated sensory noises [2] are eliminated. Thus RunBot can successfully learn to walk up an eight-degree painted slope after 3-5 falls and stably adapts its gait for walking on different terrains, i.e., level floors versus up the slope. For this demonstration, we refer the reader to the video clip at
http://www.nld.ds.mpg.de/∼poramate/RAS/EfferenceCopy.mpg.

On the other hand, when the noise cancellation circuits were not applied to the control, the sensory noises destabilize the activation parameters for the gait after a few learning experiences because the synaptic connections (ρ_1^1,...,ρ_1^6, dashed arrows in Fig. 4) between the IR signal and the sites of movement control get strengthened. Once these connections are established, RunBot will react to its IR input with a gait change as soon as it gives a high activation value (either detecting the white slope or the white spots on the floors). In addition, the weights also show small glitches arising from the self-generated noise of the AS. Such glitches lead to a weak correlation with the IR signal and to minor weight changes (see Fig. 11). As a result, these perturbations make RunBot change its gait and UBC posture and make it fall forwards during walking (see Fig. 11). For this demonstration, we refer the reader to the video clip at
http://www.nld.ds.mpg.de/∼poramate/RAS/EfferenceCopy.mpg.

[2] Recall that the external noise occurs from responding of the IR sensor to white spots placed on the level floors (see Fig. 5a) while the self-generated noise comes from the AS due to RunBot's ego motion.

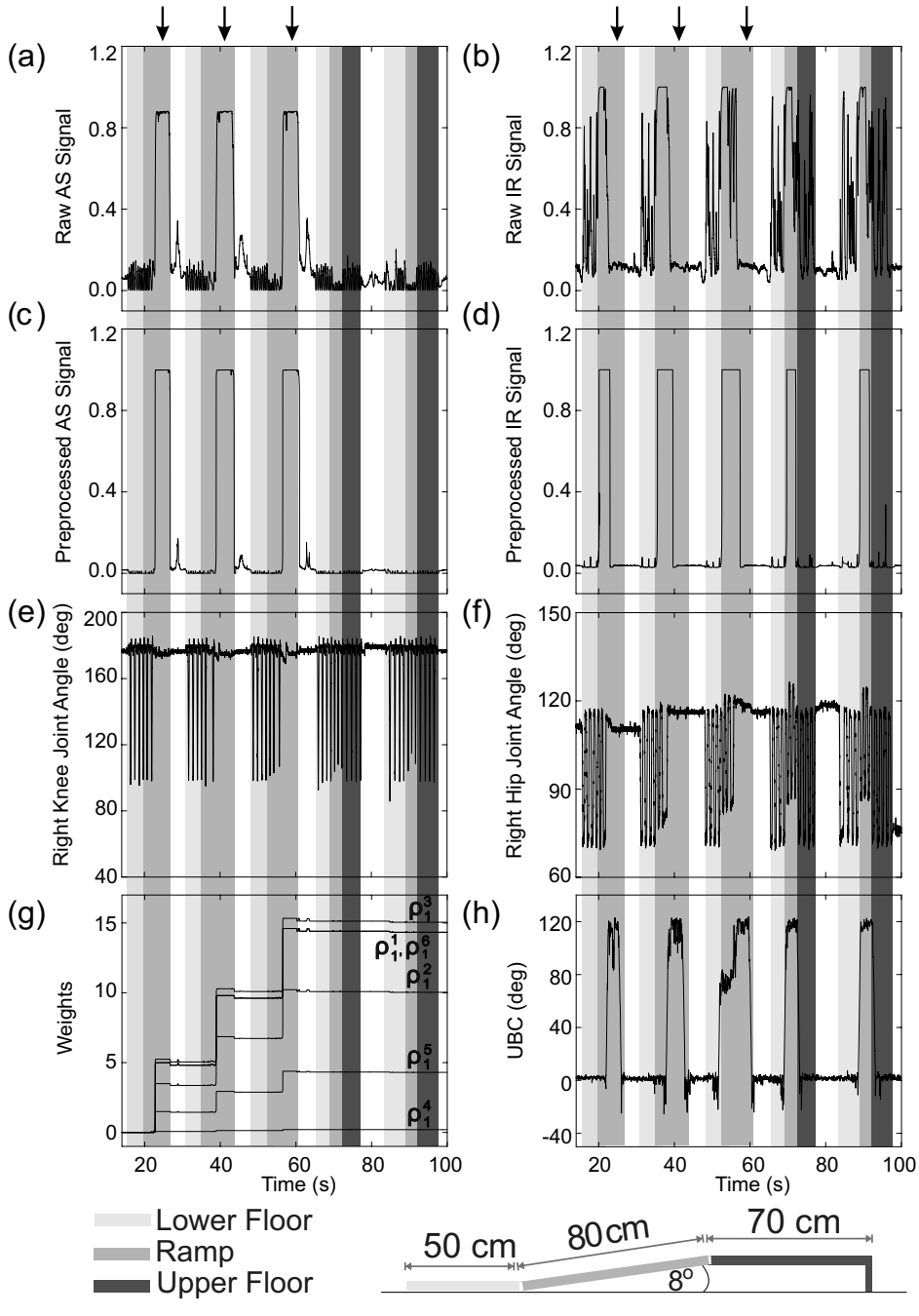

Fig. 10. Real-time data of an adaptive walking experiment when the noise cancellation circuits were applied to the control. (a), (b) Raw sensor signals. (c), (d) Corrected sensor signals showing a clear improvement. (e), (f) Right knee and hip joint angles for all situations. (g) Growing synaptic strengths during the learning phase. (h) Posture of the UBC where 0 degree means learning backwards while 120 degrees means leaning forwards. The data was recorded while RunBot was initially walking from a lower floor (light gray areas) to an upper floor (dark gray areas) through a ramp (gray areas). Arrows depict the situation where RunBot falls backwards and white areas where RunBot was manually returned to the initial position. Note that in this walking experiment we set the learning rate of each learner neuron (Eq. 2) as $\mu_1 = 10.0$, $\mu_2 = 7.0$, $\mu_3 = 10.5$, $\mu_4 = 0.14$, $\mu_5 = 3.0$, $\mu_6 = 10.0$.

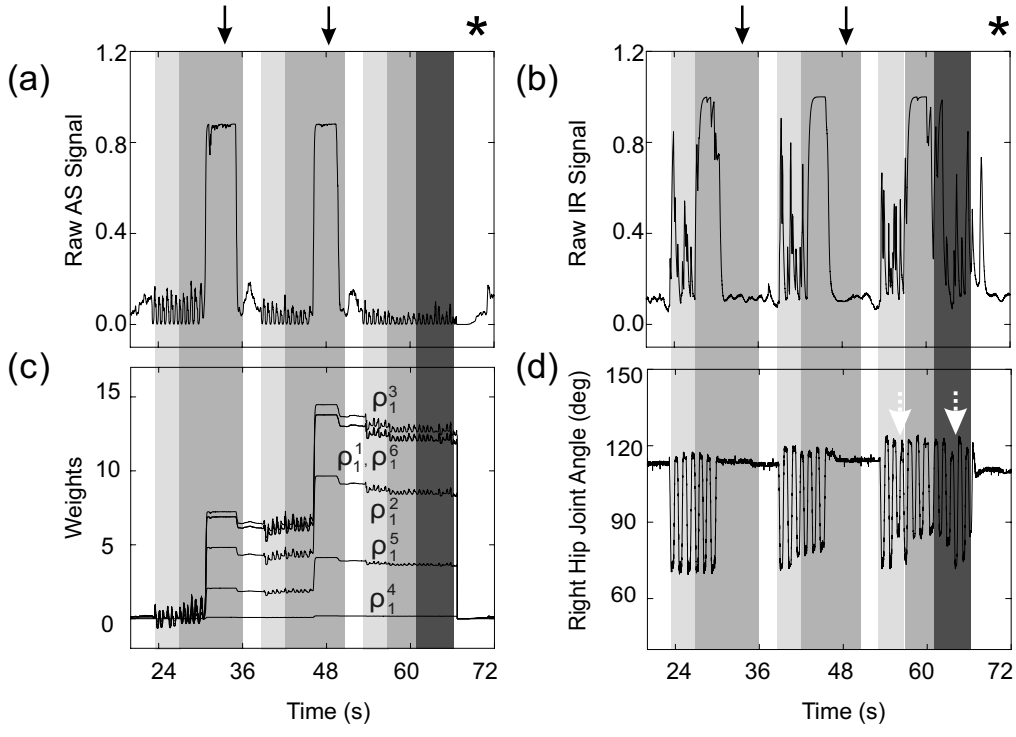

Fig. 11. Real-time data of an adaptive walking experiment when the noise cancellation circuits were not applied to the control. (a), (b) Raw sensor signals. (c) Growing synaptic strengths during the learning phase showing small glitches arising from the self-generated noise of the AS. (d) Right hip joint angle for all situations. The data was recorded while RunBot was initially walking from a lower floor (light gray areas) to an upper floor (dark gray areas) through a ramp (gray areas). Arrows depict the situation where RunBot falls backwards and white areas where RunBot was manually returned to the initial position. Dashed white arrows indicate the situation where RunBot's gait was disturbed. $*$ means the situation at which RunBot falls forwards. Note that here RunBot performs a few steps before falling forwards while walking on the upper floor. In this walking experiment we set the learning rate of each learner neuron (Eq. 2) as $\mu_1 = 10.0$, $\mu_2 = 7.0$, $\mu_3 = 10.5$, $\mu_4 = 0.14$, $\mu_5 = 3.0$, $\mu_6 = 10.0$.

4 Experiment 2: Efference copy for slope detection

Due to gravitation, which excerts a different effect during walking up a slope, RunBot's forwards motion will be resisted. Consequently, the gait period of its walking cycle will be enlarged. This can be measured at the motor signals (see Fig. 12) because they are basically derived from the proprioceptive feedback, i.e., foot contact and joint angle sensors.

According to this effect, the experimental study here will show the use of only the motor signals for discerning an unpainted slope (see Fig. 12). This can replace the use of the IR signal, where the slope needs to be painted

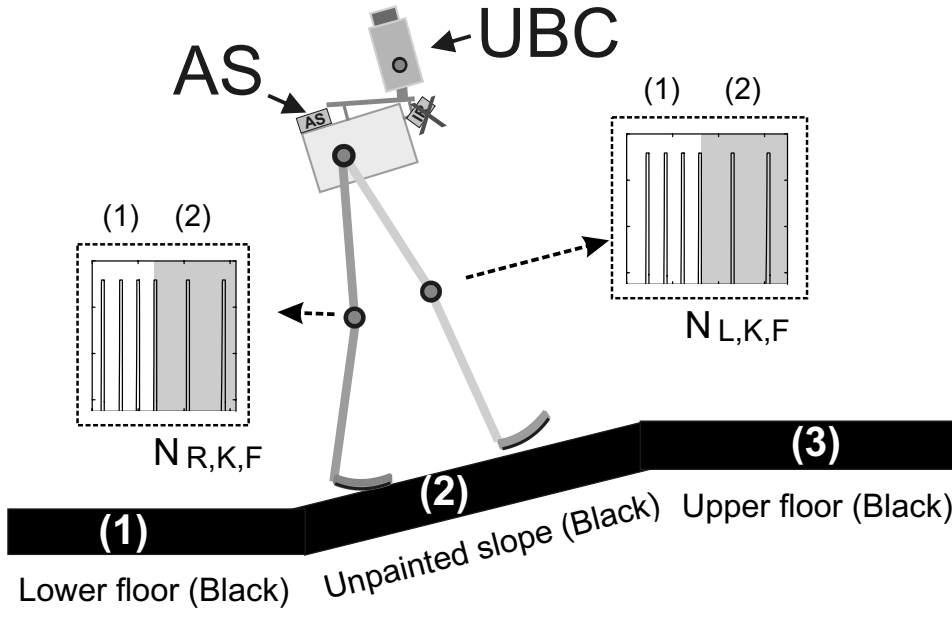

Fig. 12. Observed flexor-knee motor signals of the left ($N_{L,K,F}$) and right ($N_{R,K,F}$) legs during walking from a lower floor (1) to an eight-degree unpainted slope (2). They show that the gait period of RunBot's walking cycle is increased (gray areas in windows) while walking on the slope. In this situation, RunBot walks a few steps on the slope before it falls over backwards, as it has not yet learned to detect the slope and react to it with a change in gait and its UBC posture. Note that for this experiment we set its UBC posture in a more upright position (75 degrees) in order to allow it to walk a few steps on the slope before falling backwards.

as shown in the previous experiment. However, before applying the motor signals as a predictive signal to our learning mechanism (see Fig. 4), they need to be transformed into a signal (called slope detection signal) which can appropriately correlate with the reflex signal (AS). Hence a so-celled slope detection circuit is developed and employed for this purpose (see Fig. 13) as described in the following.

4.1 Modeling a slope detection circuit

To obtain the slope detection signal derived from the efference copy, we use here the knee flexor-motor signals of the right $N_{R,K,F}$ and left $N_{L,K,F}$ legs and feed them into the motor signal transformation circuit (see Fig. 14), which was empirically constructed. It consists of 15 neurons $M1, ..., M15$ where the neural parameters of $M1, ..., M14$ together with their function are similar to those of the neural forward model IR. That is the first recurrent neuron filters the motor noise while the rest shape the motor signals by prolonging their activation time. Eventually the output of each recurrent neuron is combined at the neuron M15. Consequently, the pulse shaped motor signals will become

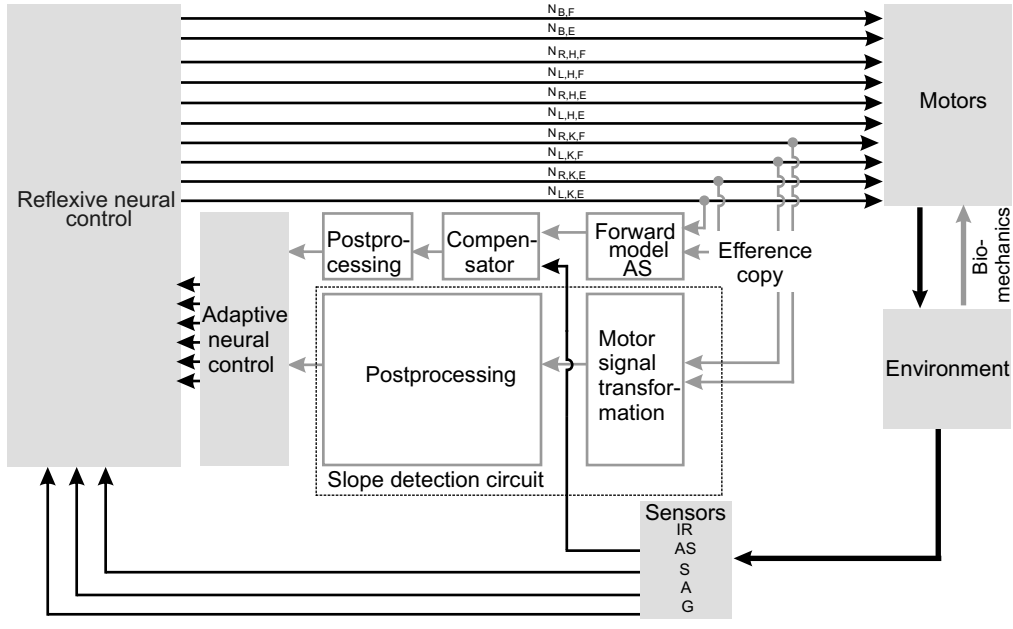

Fig. 13. Adaptive reflex neural locomotion control with noise cancellation and slope detection circuits. This control enables RunBot to detect an unpainted slope through the change of its own motor signals rather than through the IR signal. As a result, it can successfully learn to walk up the slope by utilizing only AS and motor information.

continuous and smooth. In other words, they show continuous high activation (≈ 1) during walking on level floors rather than spikes. But they will show a drop to about zero of their continuous activation while walking on the slope due to increasing the gait period of its walking cycle or they will become completely deactivated if RunBot falls backwards on the slope. Such a drop and/or deactivation enables the system to recognize the slope. However we need to convert this into a positive value for correlation with the AS signal in our learning mechanism. Thus, a postprocessing unit is used to invert the signal via neuron $P1_M$. As a result, the drop and deactivation will turn into a positive value (≈ 1) while the continuous high activation will become zero. Subsequently, the recurrent neuron $P2_M$ enlarges the response time of the inverted drop and deactivation signals. The resulting signal will be sent to the learning control circuit, which allows RunBot to learn to walk up the unpainted slope. Note that all neurons are modeled as additive neurons with a standard sigmoid transfer function (Eq. 3) except the neuron $P2_M$ which has a linear transfer function with its output restricted to the interval [0, 1].

4.2 Results

Figure 15 shows experimental results where the motor signals together with the slope detection circuit described above is employed (see Fig. 13) instead

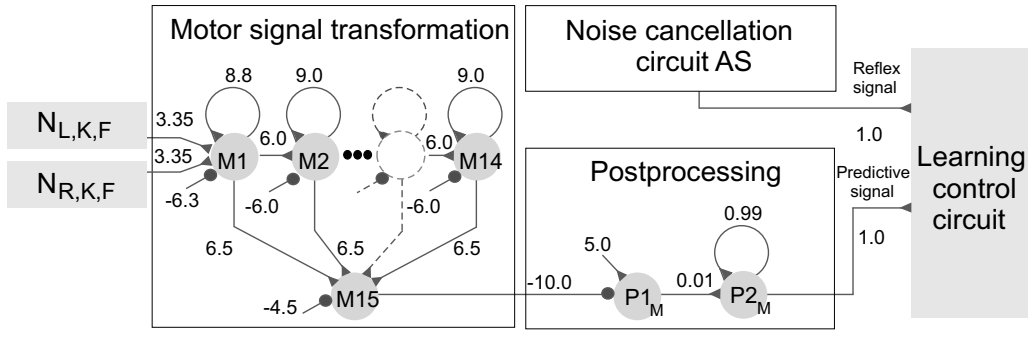

Fig. 14. Slope detection circuit. It consists of two subunits: motor transformation and postprocessing units. In the postprocessing unit the neuron $P1_M$ performs as a signal inverter while the recurrent neuron $P2_M$ with a linear transfer function acts as a low pass filter. $N_{L,K,F}$ and $N_{R,K,F}$ indicate motor signals of the left and right knee flexors, respectively. Note that one can optimize this slope detection circuit, for instance by using an evolutionary algorithm [11], but for the purposes here, manual adjustment was sufficient.

of using the IR signal for discerning a slope. As a consequence, RunBot can successfully learn to walk up an eight-degree unpainted slope after 2-5 falls. After that it can stably adapt its gait and UBC for walking on different terrains, i.e., level floors versus up the slope. For this demonstration, we refer the reader to the video clip at
http://www.nld.ds.mpg.de/∼poramate/RAS/EfferenceCopy.mpg.

5 Discussion and conclusions

Here, we concisely discuss and conclude some remaining issues following the presented experiments while most of the relevant discussion points have been described alongside the experimental sections above. In this study, we have addressed the exploitation of an efference copy together with internal models for dynamic locomotion control in terms of sensory processing and terrain determination. The first experiment has shown the relationship between afference (sensory information) and efference (motor command). That is a copy of the efference after modification through neural forward models is used to subtract external and self-generated sensory noise (sensory processing) in order to obtain perceptual stability for correctly guiding locomotion.

In the second experiment, we have demonstrated that efference copy signals derived from a reflexive mechanism [21] are capable of determining terrain condition changes, e.g., level floor versus up a slope. Due to gravitation, which excerts a different effect during walking up a slope, RunBot's forwards motion will be resisted. In other words, the gait period of its walking cycle will be enlarged which can be measured at the motor signals.

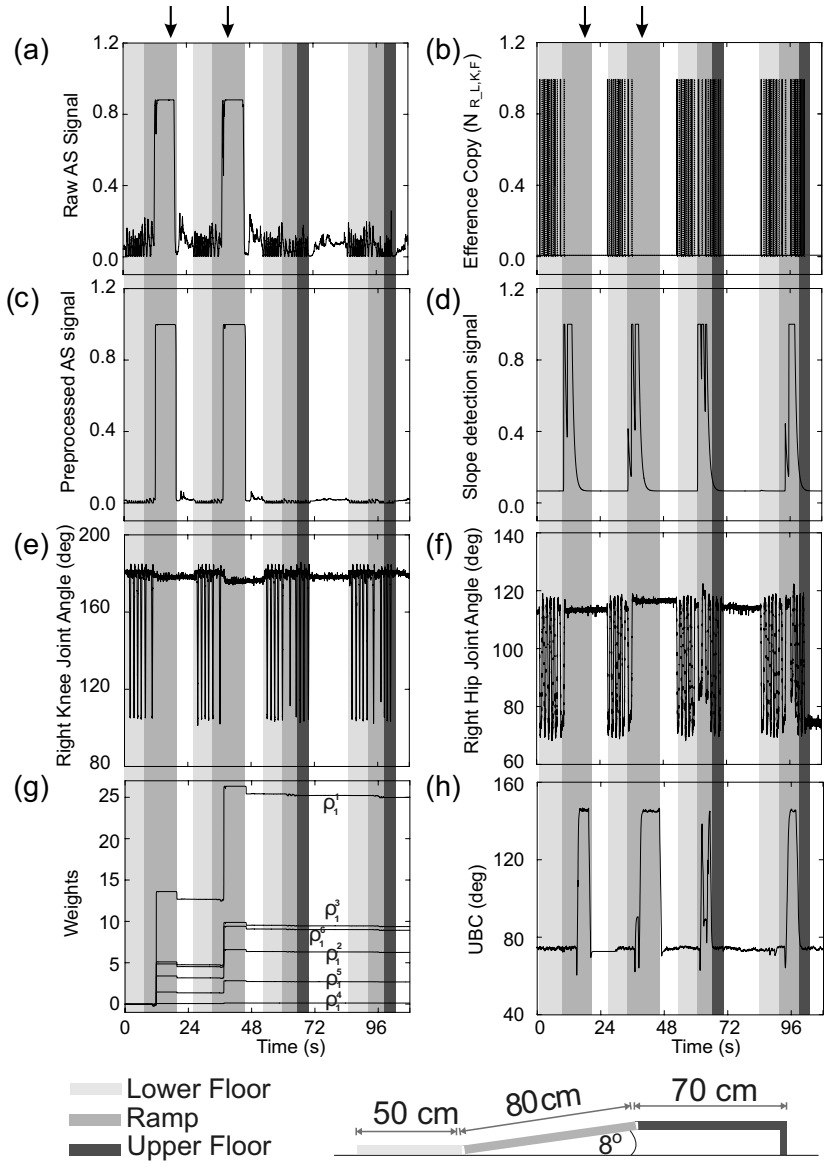

Fig. 15. Real-time data of an adaptive walking experiment where RunBot detects an unpainted slope through its motor signals. (a) Raw AS signal. (b) Knee flexor-motor signals of the right $N_{R,K,F}$ (solid line) and left $N_{L,K,F}$ (dashed line) legs used for slope detection. (c) Preprocessed AS signal (reflex signal). (d) Slope detection signal (predictive signal). (e), (f) Right knee and hip joint angles for all situations. (g) Growing synaptic strengths during the learning phase. (h) Posture of the UBC. It leans 75 degrees as an initial upright position, while 145 degrees means leaning forwards. In this experiment, RunBot can successfully walk up the slope after two learning experiences. The data was recorded while RunBot was initially walking from a lower floor (light gray areas) to an upper floor (dark gray areas) through a ramp (gray areas). Arrows depict the situation where RunBot falls backwards and white areas where RunBot was manually returned to the initial position. Note that in this walking experiment we set the learning rate of each learner neuron (Eq. 2) as $\mu_1 = 14.0$, $\mu_2 = 3.5$, $\mu_3 = 5.25$, $\mu_4 = 0.07$, $\mu_5 = 1.5$, $\mu_6 = 5.0$.

The employed dynamic locomotion controller of RunBot in these experimental studies was modeled as artificial neural networks using discrete-time dynamics. The networks consist of three main components or modules: Reflexive network for basic walking, adaptive network for learning capability, and internal model building network [3] for transforming efference copy signals into desired sensory signals. All in all this leads to a certain higher complexity of the control structure. The three modules, however, can be understood one by one, which makes network design analyzable. In addition to this, we have shown that hysteresis effects of a single recurrent neuron could be utilized for designing forward model, motor signal transformation, and postprocessing units (internal model building units). Although most units consist of several single recurrent neurons in series (see Figs. 7 and 14), they give understandable and analyzable network characteristics which can be applied to other applications [10] requiring high and low output activations at different points on the input space (hysteresis effects, see Figs. 8 and 9). If desired, all manually adjusted parameters of these units can be optimized by an evolutionary algorithm [11] or entire units can be constructed as feedforward neural networks by an error back-propagation technique.

To a certain extent the experimental studies pursued here sharpen our understanding of how the efference copy influences the dynamic locomotion control. They also emphasize how biological findings (efference copy and internal models) can be beneficially used in robotic systems. To date, efference copy and internal model concepts have been applied to a number of robot control problems in different approaches. For example, Namiki et al. (2003) [25] presented a hierarchical parallel control architecture for high-speed visual servoing (arm motion control system with visual perception). The architecture is based on an interaction model between efferent and afferent signals in a motor control network used for a parameter adaptation mechanism. As a consequence, it allows the robot to perform high-speed tracking, grasping, handling, and collision avoidance tasks. Russo et al. (2005) [29] simulated phonotaxis (auditory orientation towards sound sources) and an optomotor reflex (a visual capability compensating for external disturbances to maintain a straight trajectory) on a robot. The smooth integration of auditory and visual stimuli is achieved via a forward model. It takes acoustically driven motor command signals (efference copy) and tries to predict the reafferent visual signal such that the optomotor reflex is inhibited during phonotaxis behavior. In the domain of walking robot control, Lewis and Bekey (2002) [19] used innate internal models to transform an efference copy from a central pattern generator (CPG) unit into the sensory expectation. This expected sensory information is compared with the actual sensory feedback and an adaptive rule then modifies the CPG to coordinate the limbs of a quadruped robot. Duerr et al. (2003) [5]

[3] Here we call the noise cancellation circuits (see Fig. 7) and the slope detection circuit (see Fig. 14) as the internal model building network.

purposed a neural three-joint leg control mechanism for a hexapod robot for leg searching movement. In addition, they also provided a generalized form of their mechanism where the internal model and the efference copy are applied for central pattern control. Compared to many of these approaches, here we focus on showing the usefulness of the efference copy and internal model in dynamic locomotion control which, to the best of our knowledge, has not been investigated so far.

Although our approach cannot be directly related to how biological systems solve similar tasks, there is ample evidence suggesting that biological systems use efference copy and internal model mechanisms to maintain stable perception as well as to perform fast, robust, and adaptive behavior [3], [4], [31], [33]. For example, as described in [9], flying insects can discern self-generated sensation (i.e., rotation of the visual field caused by tracking a target) from external sensation due to changes in the external world (i.e., visual rotation due to air disturbances). This could be done by using motor outputs transformed into the expected visual inputs to suppress the self-generated sensation. In male grasshoppers, an auditory interneuron activity (G-neuron) is inhibited during stridulation (making a shrill sound by rubbing hind legs and wings) [32]. This is because proprioceptive feedback and efference copy signals of the hind legs act together to switch-off the interneuron response during stridulation. In crickets, interneurons sensitive to movement of the antennae give less activation during active motion by the cricket itself [6]. Another classic example is that moving our eyes causes the image on the retina to move, but we obtain stable image perception because the image movement is predictable from the eye movement command [23] (but see [2]). Furthermore, Cullen and Roy (2004) [28] showed that in primates vestibular signals arising from self-generated head motions are inhibited by an internal model mechanism for perceptual stability and accurate behavior control.

Finally, we would like to discuss our locomotion control network (see Figs. 6 and 13) with respect to the general concept of internal models [17], [34]. Internal models or internal loops, systems that imitate the behavior of a biological process, have appeared as an important theoretical concept in motor control. They are generally divided into two main categories [17], [34]: Inverse internal model and forward internal model. In addition, the combination of inverse and forward internal models is called integration of multiple internal models. The inverse internal model is a system that transforms a desired trajectory/state information into a motor command for generating movements. Such a model can be described as a controller. By contrast, the forward internal model is a system that predicts the next state (state estimation) and/or sensory consequences (expected sensory feedback) from the current state and motor command (efference copy). In other words, it can be viewed as a predictor. The integration of multiple internal models proposes that multiple pairs of inverse and forward internal models are tightly coupled as functional units.

Additionally, from neurophysiological and biological studies, it is known that in movement control forward and inverse models involve the dynamics of the motor system changing under different conditions. Thus it has been suggested that the internal models must be adaptable and learnable by, e.g., supervised learning [13], feedback-error-learning [16], direct inverse modeling [18], and auto-imitatively adapting inverse modeling [14], [15].

Compared to the different types of the internal models described above, the reflexive neural network of our robot can be implicitly understood as the inverse model that calculates motor commands from sensory inputs rather than desired trajectories. Note that the RunBot system does not use any trajectory control, instead only a pure sensor-driven mechanism is employed [21]. On the other hand, the internal model building network is comparable to the forward internal model that estimates sensory feedback (see experiment 1) and walking state (see experiment 2) from motor commands. This internal network was designed as a non-adaptable network where a motor learning mechanism is executed separately in the adaptive neural network [4], which results in locomotor adaptation; i.e., adaptive walking on different terrains.

Acknowledgments

This research was supported by the PACO-PLUS project as well as by BMBF (Federal Ministry of Education and Research), BCCN (Bernstein Center for Computational Neuroscience)–Göttingen W3.

References

[1] T. Bem, J. M. Cabelguen, Ö. Ekeberg, S. Grillner, From swimming to walking: a single basic network for two different behaviors, Biological Cybernetics 88 (2003) 79–90.

[2] B. Bridgeman, Efference copy and its limitations, Computers in Biology and Medicine 37 (7) (2007) 924–929.

[3] J. M. Camhi, A. Levy, Organization of a complex movement: fixed and variable components of the cockroach escape behaviour, Journal of Comparative Physiology A 163 (1988) 317–328.

[4] K. E. Cullen, Sensory signals during active versus passive movement, Curr. Opin. Neurobiol. 14 (6) (2004) 698–706.

[4] Recall that the learning algorithm applies a correlation based differential Hebbian learning rule.

[5] V. Duerr, A. Krause, J. Schmitz, H. Cruse, Neuroethological concepts and their transfer to walking machines, International Journal of Robotics Research 22 (2003) 151–167.

[6] M. Gebhart, H. W. Honnegger, Physiological characterisation of antennal mechanosensory descending interneurons in an insect (Gryllus bimaculatus, Gryllus campestris) brain, Journal of Experimental Biology 204 (2001) 2265–2275.

[7] T. Geng, B. Porr, F. Woergoetter, Fast biped walking with a sensor-driven neuronal controller and real-time online learning, The International Journal of Robotics Research 25 (3) (2006) 243–259.

[8] R. Held, Exposure history as a factor in maintaining stabilitity of perception and coordination, Journal of Nervous and Mental Disease 132 (1961) 26–32.

[9] E. v. Holst, H. Mittelstaedt, Das Reafferenzprinzip, Naturwissenschaften 37 (1950) 464–476.

[10] M. Huelse, S. Wischmann, P. Manoonpong, A. Twickel, F. Pasemann, Dynamical systems in the sensorimotor loop: On the interrelation between internal and external mechanims of evolved robot behavior, in: M. Lungarella, R. Pfeifer (eds.), Proceedings of 50 Years of Artificial Intelligence, vol. 4850, Springer-Verlag, 2007.

[11] M. Huelse, S. Wischmann, F. Pasemann, Structure and function of evolved neuro-controllers for autonomous robots, Connection Science 16 (4) (2004) 249–266.

[12] A. J. Ijspeert, A. Crespi, D. Ryczko, J. M. Cabelguen, From swimming to walking with a salamander robot driven by a spinal cord model, Science 315 (5817) (2007) 1416–1420.

[13] L. Jordan, Supervised learning and systems with excess degrees of freedom, Technical Report COINS 88/27 (1998) 1–41.

[14] K. T. Kalveram, The inverse problem in cognitive, perceptual and proprioceptive control of sensorimotor behaviour: Towards a biologically plausible model of the control of aiming movements, International Journal of Sport and Exercise Psychology 2 (2004) 255–273.

[15] K. T. Kalveram, A. Seyfarth, Learning the inverse model of the dynamics of a robot leg by auto-imitation, Springer Berlin Heidelberg, 2007.

[16] M. Kawato, Feedback-error-learning neural network for supervised motor learning, Advanced neural computers (1990) 365–372.

[17] M. Kawato, Internal models for motor control and trajectory planning, Curr. Opin. Neurobiol. 9 (1999) 718–727.

[18] M. Kuperstein, Neural model of adaptive hand-eye coordination for single postures, Advanced neural computers 239 (1998) 1308–1311.

[19] M. A. Lewis, G. A. Bekey, Gait adaptation in a quadruped robot, Autonomous Robots 12 (3) (2002) 301–312.

[20] P. Manoonpong, Neural preprocessing and control of reactive walking machines: Towards versatile artificial perception-action systems, Cognitive Technologies, Springer, 2007.

[21] P. Manoonpong, T. Geng, T. Kulvicius, B. Porr, F. Woergoetter, Adaptive, fast walking in a biped robot under neuronal control and learning, PLoS Computational Biology 3 (7) (2007) e134.

[22] P. Manoonpong, T. Geng, F. Woergoetter, Exploring the dynamic walking range of the biped robot "Runbot" with an active upper-body component, in: Proceedings of the Sixth IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006), 2006.

[23] L. Matin, Eye movements and perceived visual direction, Springer, New York, 1972.

[24] J.-A. Meyer, S. Doncieux, D. Filliat, A. Guillot, Evolutionary approaches to neural control of rolling, walking, swimming and flying animats or robots, Physica-Verlag GmbH, Heidelberg, Germany, 2003.

[25] A. Namiki, K. Hashimoto, M. Ishikawa, A hierarchical control architecture for high-speed visual servoing, International Journal of Robotics Research 22 (10–11) (2003) 873–888.

[26] F. Pasemann, Dynamics of a single model neuron, International Journal of Bifurcation and Chaos 2 (1993) 271–278.

[27] B. Porr, F. Woergoetter, Strongly improved stability and faster convergence of temporal sequence learning by using input correlations only, Neural Computation 18 (6) (2006) 1380–1412.

[28] J. E. Roy, K. E. Cullen, Dissociating self-generated from passively applied head motion: neural mechanisms in the vestibular nuclei, J. Neuroscience 24 (9) (2004) 2102–2111.

[29] P. Russo, B. Webb, R. Reeve, P. Arena, L. Patane, A cricket-inspired neural network for feedforward compensation and multisensory integration, in: Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, 2005.

[30] R. Sperry, Neural basis of the spontaneous optokinetic response produced by vision inversion, Journal of Comparative and Physiological Psychology 43 (1950) 482–489.

[31] B. Webb, Neural mechanisms for prediction: Do insects have forward models?, Trends in Neurosciences 27 (2004) 278–282.

[32] H. Wolf, O. von Helversen, 'Switching off' of an auditory interneuron during stridulation in the acridid grasshopper Chorthippus biguttulus L., Journal of Comparative Physiology A 158 (1986) 861–871.

[33] D. M. Wolpert, Z. Ghahramani, Computational principles of movement neuroscience, Nature Neuroscience 3 (2000) 1212–1217.

[34] D. M. Wolpert, Z. Ghahramani, M. I. Jordan, An internal model for sensorimotor integration, Science 269 (5232) (1995) 1880–1882.

[35] D. M. Wolpert, M. Kawato, Multiple paired forward and inverse models for motor control, Neural Networks 11 (1998) 1317–1329.

Manuscript compiled as PDF
Click here to download Manuscript: manuscript_compiled.pdf
Click here to view linked References

Autonomous Robots, SI Robot Learning manuscript No.
(will be inserted by the editor)

Using Efference Copy and Forward Internal Model for Adaptive Biped Walking

Johannes Schröder-Schetelig · Poramate Manoonpong · Florentin Wörgötter

Received: date / Accepted: date

Abstract To behave properly in an unknown environment, animals or robots must distinguish external from self-generated stimuli on their sensors. The biological inspired concepts of efference copy and internal model have been successfully applied to a number of robot control problems. Here we present an application of this for our dynamic walking robot RunBot. We use efference copies of the motor commands with a simple forward internal model to predict the expected self-generated acceleration during walking. The difference to the actually measured acceleration is then used to stabilize the walking on terrains with changing slopes.

Keywords Efference copy · forward internal model · neural network · biped robot · dynamic walking · walking machine

1 Introduction

In the early 1950s, it was proposed that in the central nervous system (CNS) motor commands are copied to predict the expected sensation (v. Holst and Mittelstaedt 1950). A motor signal going from the CNS to the periphery is called an *efference* and a signal from the peripheral sensors to the CNS is called an *afference*. An *efference copy*, which is an internal reference signal, can be used to distinguish *reafference* (sensory signals resulting from an animal's own actions) from *exafference* (sensory signals arising from external stimuli).

Later, Held (1961) indicated that efference copies and the reafference cannot be directly compared due to the different dimensionality between motor commands and sensory feedback. Therefore, he proposed a neural mechanism that transforms an efference copy signal into an expected sensory input to compare to the actually incoming sensory signal. This neural transformation mechanism is known as a *forward internal model* (Kawato 1999). The second large class of internal models is called *inverse internal models*. An inverse internal model takes a desired trajectory and transforms it into an appropriate motor command for generating the movement.

Based on these biological findings, we apply the principles of efference copy and forward internal model to our biped walking robot RunBot (Manoonpong et al. 2007) to cleanse the signal from an accelerometer sensor off the self-generated noise from the walking movement (reafference). The remaining exafference signal is then used to stabilize the walking on terrains with different slopes. This way RunBot is able to adapt to terrain changes 'blindly', i.e. without the use of the infrared sensor, which was necessary for slope detection previously (Manoonpong et al. 2007).

Johannes Schröder-Schetelig · Poramate Manoonpong · Florentin Wörgötter (✉)
Bernstein Center for Computational Neuroscience (BCCN), University of Göttingen, Bunsenstraße 10, D-37073 Göttingen, Germany
E-mail: j.schroeder-schetelig@bccn-goettingen.de, poramate@bccn-goettingen.de, worgott@bccn-goettingen.de
Tel.: +49-(0)551-5176-528; Fax: +49-(0)551-5176-449

2 Materials and methods

2.1 Mechanical Setup of RunBot

Following we give a short description of RunBot's mechanical setup. For details see (Manoonpong et al. 2007). RunBot is a planar biped walking robot, 23 cm tall from foot to hip joint axis (see Fig. 1). It is held sagittally by a boom of 1 m length, so that it cannot fall sideways, while the freely-rotating joint of the boom influences the walking dynamics in no way other than that RunBot is constrained on a circular path.

Its legs have four actuated joints: left hip, right hip, left knee and right knee. Each joint is driven by a modified RC

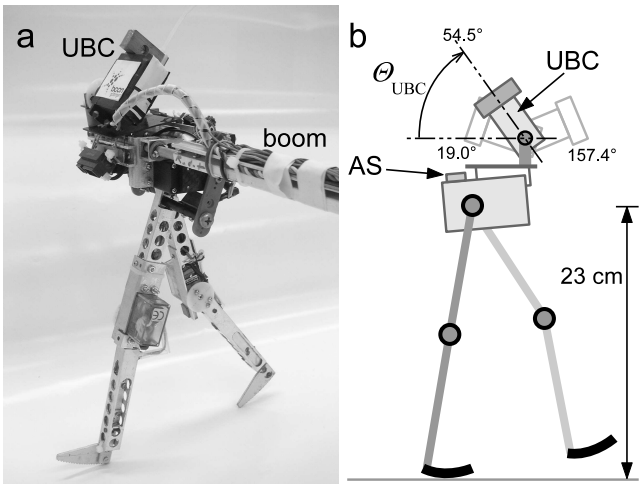

Fig. 1 a, b: The planar dynamic robot RunBot with its active upper body component (UBC) and the accelerometer sensor (AS). The UBC is drawn strongly in the zero position ($\Theta_{UBC} \equiv 54.5°$) and faintly in the minimum and maximum positions.

servo motor where the built-in pulse width modulation (PWM) control circuit is disconnected while its built-in potentiometer is used to measure the joint angles. A mechanical stopper is implemented on each knee joint to prevent it from going into hyperextension, similar to the function of human kneecaps. Approximately seventy percent of the robot's weight is concentrated on its trunk and the parts of the trunk are assembled in a way that its center of mass is located forward of the hip axis. RunBot's design also relies on the principles of passive walkers (Collins et al. 2005).

RunBot has no actuated ankle joints resulting in very light feet being efficient for fast walking. Each foot is equipped with a switch sensor to detect ground contact events. The mechanical design of RunBot has some special features, e.g. small curved feet and a properly positioned center of mass that allow the robot to perform passive walking during some stage of its step cycles. Hip and knee joints are driven by output signals of the leg controller (running on a Linux PC) through a DA/AD converter board (USB-DUX).

To extend its walking capabilities for walking on different terrains, e.g. level floor versus up or down a ramp, one servo motor with a fixed mass, called the upper body component (UBC), is implemented on top. The UBC has a total weight of 98 g (including servo). The position of the UBC is controlled by the body controller. It leans back in its "zero position" (see Fig. 1b) for walking on a level floor, while it is necessary to lean forward when RunBot walks up a ramp. The body controller relies on an accelerometer sensor (AS) serving as a vestibular organ. The AS is installed on top of the right hip joint and measures the acceleration in the direction of walking. In our set-up, the AS signal is fed to the USB-DUX for digitalization providing it to the body controller afterwards.

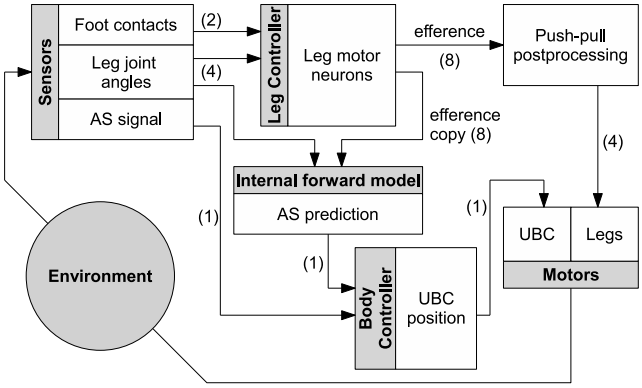

Fig. 2 Schematic diagram of leg and body control. Numbers in parentheses indicate the number of information channels going through the arrows.

2.2 Control structure

Fig. 2 schematically shows the structure of RunBot's leg and body control. For the generation of the walking movements the leg controller gets input from the feet's ground contact sensors and the legs' hip and knee joint angle sensors. Its motor neurons drive the leg motors (trough push-pull postprocessing) and via the environment a closed loop is formed back to the sensors.

The body controller drives the UBC motor and indirectly influences the walking process trough the environment. It is necessary to lean the UBC forward in order to walk up a slope. For slope detection the body controller only relies on the accelerometer sensor and has no input from a long range sensor like an infrared eye. The AS signal is dominated by the acceleration arising from RunBot's ego-motion (see Fig. 5d) and cannot directly be used for slope detection. To distinguish reafferent signals (arising from ego-motion) from exafferent signals (arising from external influences like slope changes) the body controller additionally receives input from the forward internal model (IM).

The role of the IM is to predict the expected acceleration (of the next time step) that is caused by RunBot's own motor commands (of the present time step). To do so the IM receives an efference copy of the motor commands and additionally has access to the hip and knee joint angles that define the momentary posture.

The leg controller, the forward internal model and the body controller are described in detail in the following sections.

2.2.1 Leg controller

The leg controller is a reflexive neural network with a hierarchical design. It is unchanged, inherited from the original work of RunBot (Manoonpong et al. 2007) and not subject to this study. The reflexive locomotion generation works as follows: When one foot touches the ground the hip extensor

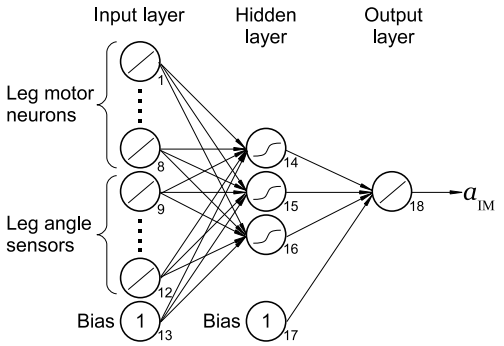

Fig. 3 Forward internal model. Three-layer feed-forward neural network with linear activation functions for input and output neurons and sigmoid activation functions in the hidden layer. The connection weights are trained by a backpropagation algorithm (see Sec. 3.1).

and knee flexor of the other leg (swing leg) are triggered, as well as the hip flexor and knee extensor of the stance leg. When the hip stretch receptor of the swing leg is activated, the extensor of the knee joint in this leg is triggered. Finally the foot of the swing leg touches the ground and the swing leg and the stance leg swap their roles thereafter. The network is designed with flexor and extensor neurons for each hip and knee motor.

Further details of the leg controller are not necessary for this study, but can be found in (Manoonpong et al. 2007). For the reader it is sufficient to know, that there exists a leg controller and that we have access to the generated motor commands, upon which we can build the internal model.

During walking on different terrains RunBot's walking patterns remain unchanged (i.e. the weights of the leg controller's neural network are constant) while adaptation is done only through active UBC control.

2.2.2 Forward internal model

We designed the forward internal model (Fig. 3) as a very simple three-layer (including input layer) feed-forward neural network. It has 12 input neurons, three hidden neurons and one output neuron. Input and hidden layer have one additional bias neuron each. The output of every single artificial neuron is defined by

$$y(\mathbf{x}) = g\left(\sum_{i=0}^{n} \omega_i x_i\right). \quad (1)$$

The neuron has n input 'dendrites' $(x_0 \ldots x_n)$ and one output 'axon' $y(\mathbf{x})$. The weights $(\omega_0 \ldots \omega_n)$ determine, how much the inputs are transmitted, and the activation function g does a transformation of the output. The bias neurons are special, they receive no input and emit a constant output of 1.0. The inputs of the IM are given by efference copies of the eight leg motor neurons (range [0, 1]) and the actual posture of the legs via the joint angle sensors (range [-1, 1]). The activation function of the input and output neurons is linear, while the hidden layer neurons have a symmetrical sigmoid activation function $g(x) = \tanh(x)$.

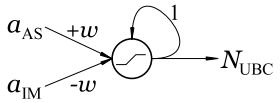

Fig. 4 Body controller. a_{AS} is the actual acceleration signal from the sensor and a_{IM} is the predicted acceleration signal from the internal model. The neuron computes the difference of both signals, weighted with the UBC control weight w, and integrates them over time via the recurrent connection. The activation function is linear, but hard limited to the range [0, 4].

The internal model not only relies on efference copies from the leg motor neurons, because the outputs of all leg motor neurons are rectangular shaped (compare Fig. 5a). Using only these as inputs of the IM the output would also have had a very stair-like appearance and would not match the AS signal very well. So we additionally used the leg joint angle sensors as inputs, and with this the prediction becomes good (see Fig. 8). For this just three hidden neurons were sufficient.

The IM was trained with data obtained during RunBot walking on a level floor, where the UBC was positioned in its "zero position" $\Theta_{UBC} \equiv 54.5^\circ$ (compare Fig. 1). The output of the IM serves as a reference signal for the body controller.

2.2.3 Body controller

The body controller (Fig. 4) drives the motor of the UBC. It consists of just one motor neuron which gets input from the accelerometer sensor and from the internal model. To obtain the exafference acceleration signal, it simply computes the difference of the two signals weighted with the factor w, which is set to a fixed value during experiment. These prediction error values are proportional to the (de-)acceleration caused by the slope of the track and the UBC posture, i.e. they are mostly positive, when RunBot is deaccelerated by the slope, and mostly negative, when RunBot is getting too fast compared to the reference signal of the internal model. The prediction error values are then integrated over time by means of the neuron's recurrent connection having synaptic strength of 1.0. This causes the UBC to move forward (backward), as long as the prediction error is positive (negative). When the prediction error vanishes, the UBC has reached a new equilibrium position. As a consequence such mechanism enables RunBot to stably continue walking on an altered terrain.

The activation function of the neuron is piecewise linear, so that the output of the neuron is clamped to the range [0, 4], which linearly corresponds to a setting of the UBC position in the range 19.0° to 157.4° given by its physical

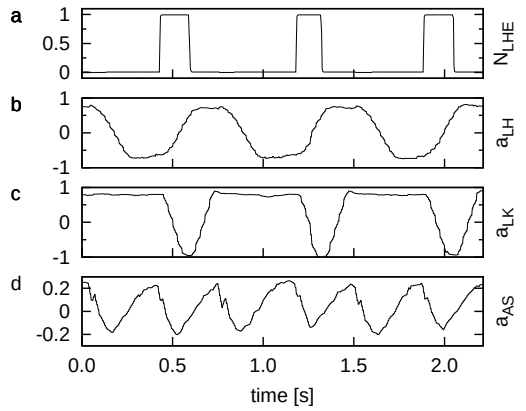

Fig. 5 Typical recordings for walking on a level floor with the UBC in its zero position. **a:** left hip extensor motor neuron (N_{LHE}). **b:** left hip angle sensor neuron (a_{LH}). **c:** left knee angle sensor neuron (a_{LK}). **d:** accelerometer sensor neuron (a_{AS}).

limits (compare Fig.1). The output N^t_{UBC} of the UBC motor neuron at time-step t is calculated according to:

$$N^t_{\mathrm{UBC}} = \begin{cases} 4 & \text{for } \tilde{N}^t_{\mathrm{UBC}} >= 4 \\ \tilde{N}^t_{\mathrm{UBC}} & \text{for } 0 < \tilde{N}^t_{\mathrm{UBC}} < 4 \\ 0 & \text{for } \tilde{N}^t_{\mathrm{UBC}} <= 0 \end{cases}$$

$$\text{where} \quad \tilde{N}^t_{\mathrm{UBC}} = w \cdot (a^t_{\mathrm{AS}} - a^t_{\mathrm{IM}}) + N^{t-1}_{\mathrm{UBC}}. \tag{2}$$

a_{AS} and a_{IM} are the output signals of AS and IM respectively and w is the UBC control weight.

The output a_{AS} of the accelerometer sensor neuron is modeled according to:

$$a_{\mathrm{AS}} = \left(1 + e^{\alpha_{\mathrm{AS}}(\theta_{\mathrm{AS}} - C_{\mathrm{AS}} V_{\mathrm{AS}})}\right)^{-1} \tag{3}$$

where V_{AS} is the output voltage signal from the accelerometer sensor. θ_{AS} and α_{AS} are the threshold and a positive constant which are set to 4.0 and 2.0, respectively. C_{AS} is a positive amplification of the input signal set to 6.0.

3 Experiments and Results

3.1 Training of the forward internal model

The network of the forward internal model was implemented using the Fast Artificial Neural Network Library (FANN, version 1.2.1). For training we recorded data from ten runs of Runbot walking on a level floor. The UBC was positioned in its zero position $\Theta_{\mathrm{UBC}} \equiv 54.5^\circ$ (corresponding to $N_{\mathrm{UBC}} \equiv 1.0$), where it stayed all the time during recording.

Fig. 5 shows typical outputs of some sensor and motor neurons during walking on a level floor. The training was done off-line, after all irrelevant data (manual return of RunBot to the start position and the transient phase) had been removed from the recorded files. The remaining training data then was shuffled randomly to avoid local minima during training and to get an over-all good prediction. First the net was initialized with random weights in the range [0.01, 0.05] and then trained incrementally to predict the accelerometer data of the next time step using a standard backpropagation algorithm. It was trained for approximately 2000 epochs up to a mean squared error of 0.00127 (one epoch = every data point used once for training). Because this error value is just a mean, we repeated the training several times with new random initialized weights, until the network showed a good over-all prediction, e.g. the prediction had a symmetrical shape for left and right steps. The connection weights of the resulting network are given in Table 1 (compare Fig. 3).

Table 1 Connection weight matrix of the internal model. The column index gives the originating neuron and the row index the target neuron. The symbol — means, that there is no connection between the neurons.

N	1	2	3	4	5	6
14	-0.454	-4.000	0.136	0.463	-0.139	0.102
15	-1.732	4.235	-0.995	-0.110	0.402	0.134
16	0.253	-2.581	-0.830	-0.211	-5.286	-0.136

N	7	8	9	10	11	12
14	0.204	0.401	0.058	-0.685	1.050	-2.177
15	-0.017	-0.640	-2.831	0.528	1.143	3.361
16	-4.282	1.618	2.454	-2.766	1.598	-0.250

N	13	14	15	16	17
14	1.276	—	—	—	—
15	-3.887	—	—	—	—
16	1.619	—	—	—	—
18	—	0.390	0.363	0.347	0.245

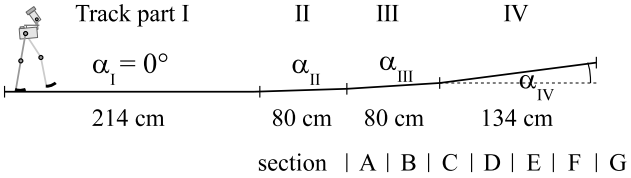

Fig. 6 Track layout. The slope of the track parts II, III and IV can be adjusted via the angles α_{II}, α_{III} and α_{IV}. Parts III and IV are divided in six sections A to F (each 35.5 cm long), while G stands for the end of the track.

3.2 Walking experiments

Walking experiments were performed on a circular track, which consists of four parts (I, ..., IV), whose lengths are 214 cm, 80 cm, 80 cm and 134 cm (see Fig. 6). The first part (I) is a level floor ($\alpha_{\mathrm{I}} = 0^\circ$). The parts II to IV have angles α_{II}, α_{III} and α_{IV} which are given in Table 2 for different track configurations.

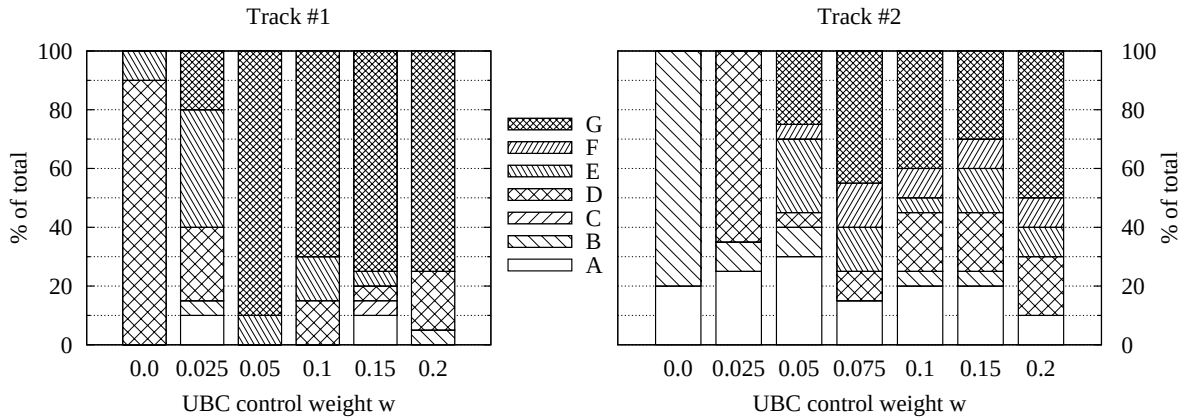

Fig. 7 Stacked histograms of the results of Experiment 1 on Track #1 and Track #2. The labels A to F stand for the section in which RunBot falls backwards (compare Fig. 6). G means that RunBot reached the end of the track.

Table 2 Different configurations of the tracks. α_{II}, α_{III} and α_{IV} are the angles of track parts II, III and IV.

Track	α_{II} [°]	α_{III} [°]	α_{IV} [°]
#1	0.6	1.9	3.7
#2	0.8	2.6	4.7
#3	0.9	2.6	4.7
#4	0.9	2.6	2.6
#5	0.9	1.3	1.3

3.2.1 Experiment 1: Body control performance

This experiment was performed on tracks #1 and #2 in order to see how efficient the trained body controller is with respect to the weight w. We set up the parts of the tracks to have gradually increasing angles up to 3.7° for track #1 and 4.7° for track #2. The angles have to increase gradually, because this type of body control only is reactive, and large and sudden changes in the slope of the track would cause RunBot to fall. To see how good control performs, we divided the last two parts of the track into six sections of length 35.5 cm each, labeled A to F (Fig. 6). For each value of the weight w we performed 20 runs and looked in which section RunBot falls. The results are shown in the stacked histograms in Fig. 7. A section value of G means that RunBot did not fall and instead successfully reached the end of the track. RunBot was placed manually at the beginning of the track with the UBC approximately in its zero position. So part I of the track had the purpose to let RunBot enter its regular walking process and to allow relaxation of the UBC to the zero position.

First we discuss the results for track #1 shown in the left histogram in Fig. 7. The first stack shows the results for $w = 0$, i.e. no body control, where the UBC stayed in its zero position. Here we can see that without body control RunBot always falls backwards at a certain point of the slope, which is in 90% of the cases section D, and in 10% section E. This is because the slope decreases the velocity of RunBot until it falls. For $w = 0.025$ we see that in 20% of the cases RunBot successfully reaches the end of the track (section G) and that the amount for section D has decreased to 20%. We also see that in some cases RunBot only got to section A and B. This is because the activation of the body control ($w \neq 0$) also introduces a certain degree of variability, and in some cases the UBC position might be below the zero position when RunBot is reaching the slope, causing it to fall earlier. Best results were obtained for $w = 0.05$ with 90% success. Larger values of w led to lower success rates (60%-65%) with higher instability.

The results for track #2 are shown in the right histogram in Fig. 7. Here the angles are larger than on track #1, so we expect that we have to use larger values for w to get similar results. For $w = 0$ we see again that RunBot is falling, this time a little bit earlier at section B (80%). With $w = 0.025$ RunBot gets up to section D but still falls in all trials. Results are getting better for $w = 0.05$ and are best for $w = 0.075$ with 45% success. For $w = 0.1$ and $w = 0.15$ performance drops again. $w = 0.2$ shows good results comparable to $w = 0.075$. This is due to the effect that with this strong weight, even already on the level floor, sometimes the UBC went directly to the front and stayed there, because it cannot go further. Although the IM prediction actually was *not* good in this situation, RunBot was able to easily reach the end of the track, because of its self-stabilizing properties (Manoonpong et al. 2007).

3.2.2 Adaptive walking example

Fig. 8 presents the results of a walk on track #1 with control weight $w = 0.05$ taken from Experiment 1. RunBot leaned its UBC forward and successfully reached the end of the track without falling. Fig. 8a shows the outputs of the ac-

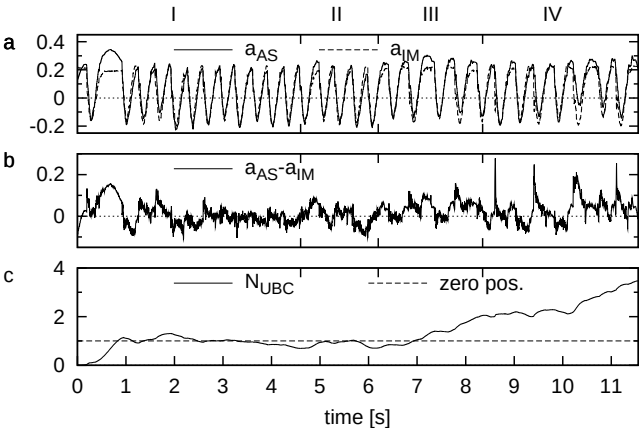

Fig. 8 Recordings of a walk on track #1 with control weight $w = 0.05$. RunBot leaned his UBC forward and successfully reached the end of the track without falling. **a:** Outputs of AS and IM. **b:** Resulting difference of actual and predicted acceleration signal. **c:** Output of UBC motor neuron.

celerometer sensor neuron and the internal model, while Fig. 8b magnifies the difference of these actual and predicted AS signals (prediction error). A positive/negative error drives the output of the UBC motor neuron up/down (compare Fig. 8c). The first 4.6 seconds Runbot was walking on a level floor and one can see how the difference of the actual and the predicted acceleration signal (prediction error) drives the UBC position to the zero position, where is was during training (dashed line in Fig. 8c). Because the AS signal now nearly resembles the reference signal of the IM, the prediction error is becoming small and the UBC oscillates around the zero position. As RunBot reaches the slope the prediction error is getting positive most of time and consequently the UBC position increases. But because the slope of the track still is getting steeper, this goes on till the end of the track is reached. If the slope had continued with a fixed angle, the UBC position would have converged to a certain value, as can be seen from the following Experiment 2.

The supplementary Video 1 shows some walks of RunBot on track #3.[1] First it is shown that with deactivated body controller ($w = 0$) RunBot falls backwards at a certain point of the track, when the slope is getting too steep. Then the controller is activated ($w = 0.1$) and RunBot is able to reach the top end of the track. Also note that here the initial positions of the UBC are just roughly set to the zero position.

3.2.3 Experiment 2: UBC equilibrium position for different slopes

With this experiment we wanted to check if the UBC position converges to a specific value for a track with a certain slope. For this we used track configurations #4 and #5,

[1] The supplementary video can be downloaded at: `http://www.nld.ds.mpg.de/~poramate/AutonomousRobots/Video1.mpg`

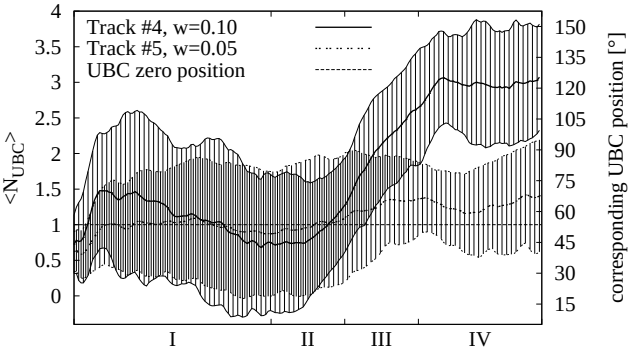

Fig. 9 Converging of the mean UBC position to new equilibrium positions for tracks with different slopes. The shaded areas around the mean curves give the standard deviations.

where the last two parts III and IV had equal slopes $\alpha_{\mathrm{III}} = \alpha_{\mathrm{IV}}$. Again we recorded several runs of RunBot like in Experiment 1. For track #4 ($\alpha_{\mathrm{III}} = \alpha_{\mathrm{IV}} = 2.6^\circ$) we recorded $n = 18$ successful runs with weight $w = 0.1$, for track #5 ($\alpha_{\mathrm{III}} = \alpha_{\mathrm{IV}} = 1.3^\circ$) we got $n = 19$ runs with $w = 0.05$. The positions of the UBC differ from run to run and are quite sensitive to the initial values, but on average a clear tendency is observable. In Fig. 9 the mean UBC position $\langle N_{\mathrm{UBC}} \rangle = \frac{1}{n}\sum_{i=1}^{n} N^i_{\mathrm{UBC}}$ is shown for both tracks.[2] The shaded areas give the standard deviations $\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(N^i_{\mathrm{UBC}} - \langle N_{\mathrm{UBC}} \rangle)^2}$ of the curves. In track part I the UBC position rises from the initial position and oscillates around the zero position as before. In part II it begins to rise together with the track slope. This continues in part III, and in part IV the UBC finally stabilizes and oscillates around the new equilibrium positions, which are approximately 122° for track #4 and 63° for track #5. As expected the equilibrium position takes larger values for steeper slopes. For track #4 the UBC position for a few times reached its upper limit, where the output of the body controller neuron was clamped to $N_{\mathrm{UBC}} = 4.0$. Nevertheless this is a real new equilibrium position and not just an artifact of the clamping.

4 Discussion and Outlook

We have demonstrated the use of biologically inspired principles of signal processing in a walking robot. Based on efference copies of motor commands it was possible to predict the afferent signals of an accelerometer sensor using a simple forward internal model. This acceleration signal prediction was subtracted from the actual acceleration signal to obtain an exafference, which was successfully used to stabilize the walking on terrains with changing slopes. The dependence of the body control performance on the UBC weight

[2] The index i in N^i_{UBC} here denotes the index of the run and not the time-step as in Eq. (2).

w was studied. Finally we verified, that the UBC position settles to new equilibrium positions for different slopes.

The approach is simple yet quite powerful, but there could be several ways for improvement: i) The IM is designed as a feed-forward network with access only to the actual sensory data, but it might perform better if it had access also to the history or if it were designed as a recurrent network. ii) The training of the IM was done off-line. It would be useful if it could be trained during walking. iii) The body controller does only control the posture of the UBC. If also the weights of the leg controller neurons responsible for step length had been adapted, it should be possible for RunBot to walk up much steeper slopes as shown in (Manoonpong et al. 2007). iv) The UBC weight w now has to be adjusted by hand. It would be preferable, that it adapts automatically.

Finally, we would like to discuss our approach with respect to other applications of efference copies and internal models in robotic systems. For example Lewis and Bekey (2002) presented a model for a quadruped robot, that – like a newborn foal – can learn to walk several minutes after inception. They used an efference copy from a central pattern generator (CPG) that was transformed into the sensory expectation via innate internal models. This information is compared to the actual sensory feedback and an adaptive rule tunes the CPG to coordinate the limbs. Dürr et al. (2003) proposed a neural control mechanism for three-joint legs of a hexapod robot for leg searching movement. They also present a generalized form of the mechanism, where the internal model and the efference copy are applied for central pattern control. Russo et al. (2005) simulated a robot with phonotaxis (auditory orientation towards sound sources) and optomotor reflex (visual capability allowing to maintain a straight trajectory against disturbances). The motor commands driven by the phonotaxis reflex (efference copies) are transferred to the expected reafferent visual signal via a forward model. This way it is possible to smoothly integrate the visual and auditory stimuli, filtering out the optical disturbances caused by the phonotactic reflex, while still reacting to external stimuli. Compared to such approaches our study to a certain extent shows how efference copy and forward models can be applied in dynamic locomotion control, which, to the best of our knowledge, has not been investigated so far.

References

S. Collins, A. Ruina, R. Tedrake, M. Wisse, *Efficient Bipedal Robots Based on Passive-Dynamic Walkers*, Science 307, 1082 (2005).

V. Dürr, A. Krause, J. Schmitz and H. Cruse, *Neuroethological Concepts and their Transfer to Walking Machines*, International Journal of Robotics Research 22, 151-167 (2003).

FANN: Fast Artificial Neural Network Library, available at: `http://www.sourceforge.net/projects/fann`

R. Held, *Exposure history as a factor in maintaining stabilitity of perception and coordination*, Journal of Nervous and Mental Disease 132, 26-32 (1961).

E. v. Holst and H. Mittelstaedt, *Das Reafferenzprinzip*, Die Naturwissenschaften 37, 464-476 (1950).

M. Kawato, *Internal models for motor control and trajectory planning*, Current Opinion in Neurobiology 9, 718-727 (1999).

M. A. Lewis, G. A. Bekey, *Gait Adaptation in a Quadruped Robot*, Autonomous Robots 12(3), 301-312 (2002).

P. Manoonpong, T. Geng, T. Kulvicius, B. Porr and F. Wörgötter, *Adaptive, Fast Walking in a Biped Robot under Neuronal Control and Learning*, PLoS Computational Biology 3(7), e134 (2007).

P. Russo, B. Webb, R. Reeve, P. Arena, L. Patan, *A cricket-inspired Neural Network For FeedForward Compensation and Multisensory Integration*, Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference (2005).