

Using Efference Copy and a 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 biologically 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 through its upper body component controller. As a consequence, the controller drives the upper body component (UBC) to lean forwards/backwards as soon as an error occurs resulting in dynamical stable walking. We have evaluated the performance of the system on four different track configurations. Furthermore 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.

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
Max Planck Institute for Dynamics and Self-Organization,
Bunsenstr. 10, D-37073 Göttingen, Germany
E-mail: johannes.schroeder-schetelig@ds.mpg.de

Poramate Manoonpong · Florentin Wörgötter (✉)
Bernstein Center for Computational Neuroscience (BCCN), University
of Göttingen, Friedrich-Hund-Platz 1, D-37077 Göttingen, Germany
Phone: +49 551 39-10760; Fax: +49 551 39-7720
E-mail: poramate@physik3.gwdg.de, worgott@physik3.gwdg.de

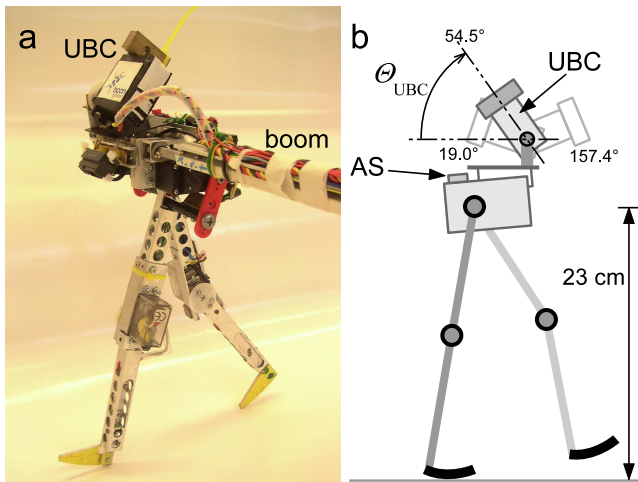


Fig. 1 a, b: The planar dynamic robot RunBot with its active upper body component (UBC) and the accelerometer sensor (AS). The UBC is drawn strongly in the zero position ($\theta_{\text{UBC}} \equiv 54.5^\circ$) and faintly in the minimum and maximum positions.

This work has been published in preliminary form on a Poster Presentation at Bernstein Symposium 2008, Munich, Germany, (Schröder-Schetelig et al. 2008).

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 (radio controlled) 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

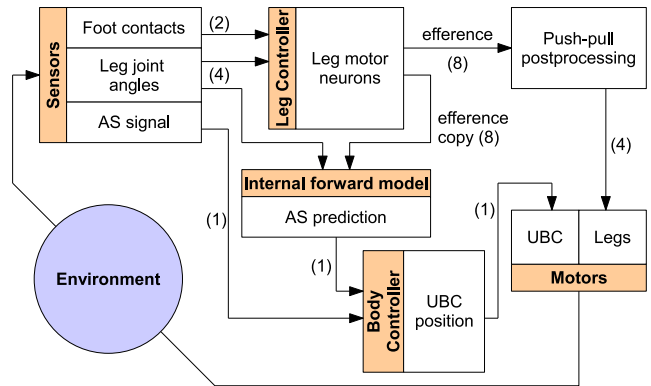


Fig. 2 Schematic diagram of leg and body control. Numbers in parentheses indicate the number of information channels going through the arrows.

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.

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 (through push-pull post-processing) and via the environment a closed loop is formed back to the sensors. There are eight motor neurons for just four leg servos. This is because the original neural design of RunBot is biologically inspired and resembles the principle of antagonistic muscle pairs (flexor / extensor). Muscles can only exert a pulling force, and therefore one muscle (the agonist) creates a specific movement while the other muscle (antagonist) is passively stretched back to its original posi-

tion. The push-pull postprocessing recombines the signals from two motor neurons to generate a single voltage signal for the servo.

The body controller drives the UBC motor and indirectly influences the walking process through 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. 6d) 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 idea behind all this is not that the robot has a good internal model for walking on slopes with arbitrary inclination and therefore can detect external disturbances on the slopes. It is rather the aim that the robot has a good model for walking on level floor (normal situation) and can detect and compensate disturbances – which are the slopes. In other words, the robot knows how the movement of the walking “feels” on level floor and on a slope it tries to get the same “sensation”, which is governed by the walking speed. If it is getting too slow, then it leans forward, if it is getting too fast, it leans backward. This applies equally to walking on level floor as on a slope.

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 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.

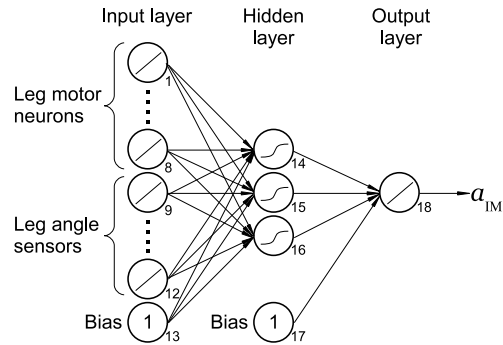


Fig. 3 Forward internal model. Three-layer feed-forward neural network with linear activation functions for input and output neurons and sigmoid activation functions in the hidden layer. The connection weights are trained by a backpropagation algorithm (see Sec. 3.1).

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’ ($x_0 \dots x_n$) and one output ‘axon’ $y(\mathbf{x})$. The weights ($\omega_0 \dots \omega_n$) 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 $g(x) = \tanh(x)$.

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. 6a). Using only these as inputs of the IM the output would also have had a very stair-like appearance and would not match

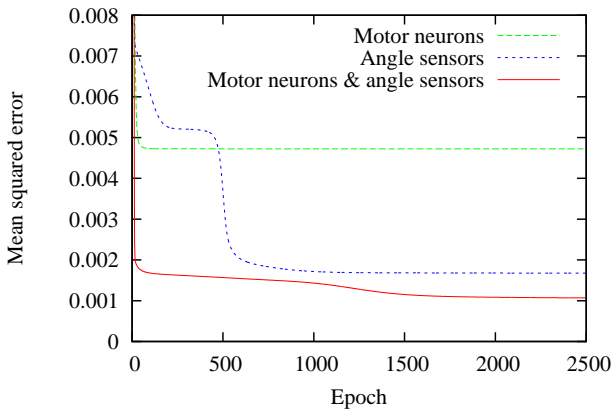


Fig. 4 Comparison of the learning curves for training the forward internal model with three different kinds of inputs: solely efference copies of the motor neurons, solely joint angle sensors and both together. The latter gives the best results.

the AS signal very well. Figure 4 shows typical learning curves for training the forward internal model based on different inputs. Using just efference copies of the motor neurons results in a network with an high mean squared error of about 0.0047. By using solely the joint angle sensors as inputs the training of the IM takes longer, but it finally comes to a much smaller error of 0.0017. Taking both motor neurons and angle sensors as inputs leads to a very quick learning of the network and after 1500 epochs the error even drops further to 0.0011. As a result, this shows that providing the sensory feedback and efference copies as inputs to the network gives the best performance of sensory prediction; i.e., smallest error. Thus, one should use methods that are able to deal with multiple inputs and here we feel that networks supercede conventional hand-designed control methods as networks can learn the balance of the different inputs. Therefore, we conclude that in the special case shown here a PID controller could probably still do the job, but that the here existing multi-input situations already would require careful design of such controller, where network learning will find the optimal solution without efforts. For our approach here 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” $\Theta_{\text{UBC}} \equiv 54.5^\circ$ (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. 5) 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

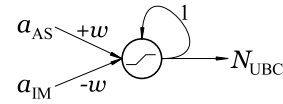


Fig. 5 Body controller. a_{AS} is the actual acceleration signal from the sensor and a_{IM} is the predicted acceleration signal from the internal model. The neuron computes the difference of both signals, weighted with the UBC control weight w , and integrates them over time via the recurrent connection. The activation function is linear, but hard limited to the range $[0, 4]$.

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

$$N_{\text{UBC}}^t = \begin{cases} 4 & \text{for } \tilde{N}_{\text{UBC}}^t \geq 4 \\ \tilde{N}_{\text{UBC}}^t & \text{for } 0 < \tilde{N}_{\text{UBC}}^t < 4 \\ 0 & \text{for } \tilde{N}_{\text{UBC}}^t \leq 0 \end{cases}$$

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

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:

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

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.

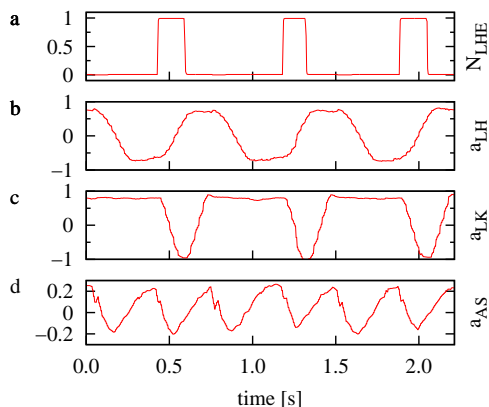


Fig. 6 Typical recordings for walking on a level floor with the UBC in its zero position. **a:** left hip extensor motor neuron (N_{LHE}). **b:** left hip angle sensor neuron (a_{LH}). **c:** left knee angle sensor neuron (a_{LK}). **d:** accelerometer sensor neuron (a_{AS}).

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 (Nissen 2003). For training we recorded data from ten runs of Runbot walking on a level floor. The UBC was positioned in its zero position $\Theta_{UBC} \equiv 54.5^\circ$ (corresponding to $N_{UBC} \equiv 1.0$), where it stayed all the time during recording.

Fig. 6 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 to predict the accelerometer data of the next time step using a standard backpropagation algorithm, where the weights are updated after each training pattern. 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 randomly initialized weights, until the network showed a good over-all prediction, e.g. the prediction had a symmetrical shape for left and right steps. Note that the UBC position is not included into the learning because it is fixed at 54 degrees and would only lead to a bias term. If we had used several training sets on level floor with different UBC positions for training, then the difference between the predicted and actual acceleration signal would always be almost zero on level floor, regardless of the UBC position. This means that the UBC would be driven by small random fluctuations to

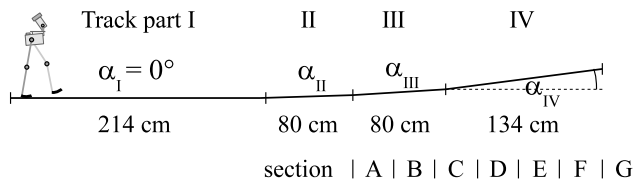


Fig. 7 Track layout. The slope of the track parts II, III and IV can be adjusted via the angles α_{II} , α_{III} and α_{IV} . Parts III and IV are divided in six sections A to F (each 35.5 cm long), while G stands for the end of the track.

Table 1 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} [$^\circ$]	α_{III} [$^\circ$]	α_{IV} [$^\circ$]
#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

any possible position, instead of the desired behavior. As a consequence, on the slope the situation is expected to become even worse.

The connection weights of the resulting network are given in Table 2 in the appendix.

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. 7). The first part (I) is a level floor ($\alpha_I = 0^\circ$). The parts II to IV have angles α_{II} , α_{III} and α_{IV} which are given in Table 1 for different track configurations.

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. 7). 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. 8. 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

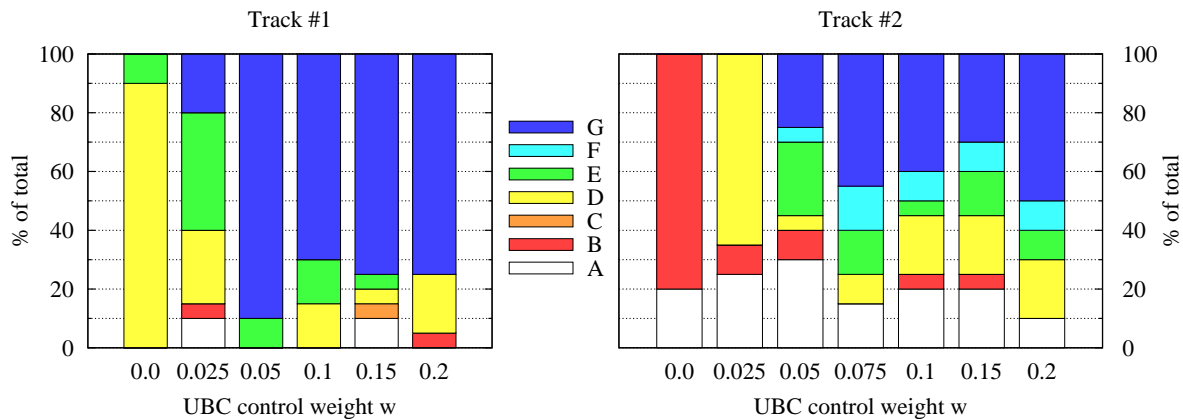


Fig. 8 Stacked histograms of the results of Experiment 1 on Track #1 and Track #2. The labels A to F stand for the section in which RunBot falls backwards (compare Fig. 7). G means that RunBot reached the end of the track.

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 an equilibrium position near the zero position.

First we discuss the results for track #1 shown in the left histogram in Fig. 8. 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. 8. 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 an even slightly better success rate than $w = 0.075$, but this is not caused by a good performance of the control system, but rather because of the self-stabilizing properties of RunBot (Manoonpong et al. 2007). In fact the control system behaved badly with the strong weight ($w = 0.2$) and we observed that sometimes, regard-

less of the actual slope, and even already on the level floor, the UBC went directly to the front and stayed there, because it cannot go further. It seems that the forward internal model was driven into a range that it was not trained for. So although the IM prediction actually was not good in this situation, RunBot was able to easily reach the end of the track, because the front most UBC position is optimal for walking upslope and is still acceptable for walking on level floor with a faster speed because of the self-stabilizing properties. This position, however, is not appropriate for walking down slopes (not shown here but see (Manoonpong et al. 2006) for experiments) where it will definitely lead to falling forward since the center of mass moves too far out from the supporting foot area. Here we consider the front most UBC position as inappropriate, because we want RunBot to walk with an upright UBC position on level floor and lean the UBC only if necessary as in natural human walking. For smaller weights this behavior was not observed. Note that one can observe that the quality of the walking behavior against the weight is a kind of inverted bell curve in both walking tracks according to the success rate in section A (not falling). This is because RunBot is not a trajectory-controlled robot and its dynamical stability is also derived from the moving speed of the UBC according to the strength of the UBC weight.

3.2.2 Adaptive walking example

Fig. 9 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. 9a shows the outputs of the accelerometer sensor neuron and the internal model, while Fig. 9b 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. 9c). The first 4.6 seconds Runbot was walking on a level

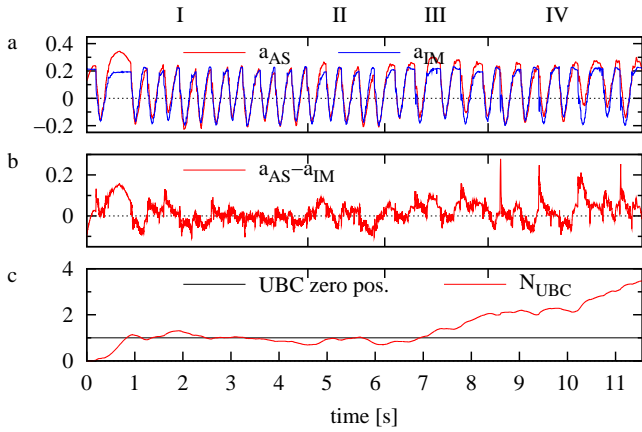


Fig. 9 Recordings of a walk on track #1 with control weight $w = 0.05$. RunBot leaned his UBC forward and successfully reached the end of the track without falling. **a:** Outputs of AS and IM. **b:** Resulting difference of actual and predicted acceleration signal. **c:** Output of UBC motor neuron.

floor and one can see how the difference of the actual and the predicted acceleration signal (prediction error) drives the UBC position to an equilibrium position slightly below the zero position, where it was during training (dashed line in Fig. 9c). 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 equilibrium 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 (Online Resource 1) shows some walks of RunBot on track #3. 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, where the last two parts III and IV had equal slopes $\alpha_{III} = \alpha_{IV}$. Again we recorded several runs of RunBot like in Experiment 1. For track #4 ($\alpha_{III} = \alpha_{IV} = 2.6^\circ$) we recorded $n = 18$ successful runs with weight $w = 0.1$, for track #5 ($\alpha_{III} = \alpha_{IV} = 1.3^\circ$) we got $n = 19$ runs with $w = 0.05$. The

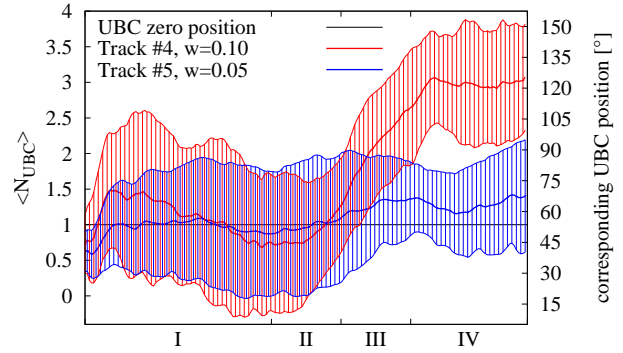


Fig. 10 Converging of the mean UBC position to new equilibrium positions for tracks with different slopes. The shaded areas around the mean curves give the standard deviations.

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. 10 the mean UBC position $\langle N_{UBC} \rangle = \frac{1}{n} \sum_{i=1}^n N_{UBC}^i$ is shown for both tracks.¹ The shaded areas give the standard deviations of the curves:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (N_{UBC}^i - \langle N_{UBC} \rangle)^2}$$

In track part I the UBC position rises from the initial position and relaxes to the equilibrium position for flat terrain as before. On track #4 the UBC position is overshooting the zero position on the first steps because of the stronger weight w , which leads to more pronounced oscillations around the equilibrium position. 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 $N_{UBC} = 4.0$. 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

¹ The index i in N_{UBC}^i 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. Furthermore, we have showed that the length of the slope will not disturb the stability of the system (compare Fig. 9 and Fig. 10). It is important to note that our concern here is also to show that the learned network (forward internal model) is general such that it enables RunBot to adaptively walk on different terrains without changing the network parameters and structure. However, only the UBC weight is required to change according to a terrain condition leading to simplification when an online learning mechanism is later applied (Porr and Wörgötter 2003).

Although some of us have previously investigated the use of efference copies and neural control for adaptive walking of RunBot (Manoonpong and Wörgötter 2009), the results here are different as follows: 1) The two experiments conducted in the previous work were embedded in the context of the adaptive neural control and learning mechanism described in (Manoonpong et al. 2007), whereas in our current work the infrared eye (IR), the UBC reflex behaviour and the adaptive neural control and learning mechanism are omitted. Instead we implement a new type of UBC controller that solely relies on the principles of efference copy and internal forward model. 2) The first experiment described in (Manoonpong and Wörgötter 2009) used efference copies and internal forward models to eliminate external and self-generated periodic noise from the IR and AS sensors to enhance learning performance. This is in contrast to our current work, where we do not aim to eliminate the external error, but instead use it to drive the UBC controller. 3) In the second experiment of the previous work a slope detecting circuit was constructed based on efference copies to replace and simulate the IR signal, as required by the learning mechanism. No online comparison of the actual and predicted IR signal was performed. 4) The internal forward model here is developed based on a simple feed-forward network, which is trained offline with a standard back-propagation of error algorithm, while in the previous work it was manually constructed based on dynamical properties of recurrent neural networks.

To a certain extent our experimental study pursued here sharpens our understanding of how the efference copy can be exploited for the dynamic locomotion control in particular walking on different terrains. It also emphasizes how biological findings (efference copy and internal models) can be beneficially used in robotic systems. Up to date, efference copy and internal model concepts have been applied to a number of robot control problems in different ways. For example 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 phonotaxis reflex, while still reacting to external stimuli. Namiki et al. (2003) 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. In the domain of legged locomotion control, 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. Lewis and Simó (2001) used motor data phase, motor signals (efference copies), and other sensory signals including visual information to enable the bipedal robot to be aware of unexpected features in the environment as well as to the sensory consequences (sensory prediction) of its own movement. As a consequence the robot can learn to expect a smooth surface in front of it when trained on a smooth surface, and without being explicitly told about smooth surfaces. Note that the robot, however, due to its hip joints fixed attachment to a boom, is indeed not a dynamic biped. 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.

In general most bipedal robots use the target ZMP (Zero Moment Point) (Vukobratovic et al. 1990) control algorithm for locomotion in particular on different and uneven terrains, which requires precise modeling and actuation with high control gains (Kim et al. 2005; Huang et al. 2008). However, there are also other interesting approaches for bipedal walking on different terrains. For example, Iida et al. (2006) proposed a dual adaptation loop model for locomotion control on up/down slopes of a simulated biped walking robot. The first adaptation loop is based on the phase entrainment ability of pattern generators. The other is for the feedforward elicitation of sensorimotor constraints; that is kinematic parameters constrain limbs trajectories (e.g. length of stride) according to the environmental state). Miyakoshi (2006) proposed memory based control where a robot can walk on a

known slope and a rolling slope. Ogino et al. (2008) developed a walking controller that enables a robot to walk on rough terrain by changing the compliance of the joints without sensing the state of the surface of the ground. Iida and Tedrake (2009) presented a minimalistic control architecture with a minimum sensory feedback of a compass gait model for dynamic bipedal walking on the different inclinations of slope.

In contrast to all these locomotion control mechanisms, our controller here is purely based on neural control and it does not employ any trajectory control for locomotion generation. Instead only a pure sensor-driven mechanism is employed. To obtain adaptive walking on different terrains we use efference copies and a neural forward internal model for sensory sequence prediction. Although the developed forward model is quite simple, it is general as described above. Nevertheless 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 offline. 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 and different terrain conditions require a slightly different UBC weight for effective walking. However, based on this experimental study one can use this setup with some preset weight values and employ an online learning method (Porr and Wörgötter 2003), such that the robot can learn to select the appropriate weight by itself governed by the momentarily existing terrain condition. This would require an additional sensor for determining terrain condition, e.g. a slope angle detection sensor and goes beyond the scope of the current study.

Acknowledgements 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

- Collins, S., Ruina, A., Tedrake, R., Wisse, M. (2005). Efficient Bipedal Robots Based on Passive-Dynamic Walkers. *Science* 307, 1082–1085.
- Dürr, V., Krause, A., Schmitz, J., Cruse, H. (2003). Neuroethological Concepts and their Transfer to Walking Machines. *International Journal of Robotics Research* 22, 151–167.
- Held, R. (1961). Exposure history as a factor in maintaining stability of perception and coordination. *Journal of Nervous and Mental Disease* 132, 26–32.
- von Holst, E., Mittelstaedt, H. (1950). Das Reafferenzprinzip. *Die Naturwissenschaften* 37, 464–476.
- Huang, W., Chew, C.-M., Zheng, Y., Hong, G.-S. (2008). Pattern generation for bipedal walking on slopes and stairs. In: *8th IEEE-RAS International Conference on Humanoids*, 205–210.
- Iida, S., Kondo, T., Ito, K. (2006). An Environmental Adaptation Mechanism for a Biped Walking Robot Control Based on Elicitation of Sensorimotor Constraints. *SAB 2006, LNAI 4095*, 174–184.
- Iida, F., Tedrake, R. (2009). Minimalistic control of a compass gait robot in rough terrain. *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA)*, IEEE/RAS, 1985–1990.
- Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology* 9, 718–727.
- Kim, D., Seo, S.-J., Park, G.-T. (2005). Zero-moment point trajectory modelling of a biped walking robot using an adaptive neuro-fuzzy system. *IEEE Proceedings - Control Theory and Applications* 152(4), 411–426.
- Lewis, M. A., Simó, L. S. (2001). Certain Principles of Biomorphic Robots. *Autonomous Robots, Special Issue on Biomorphic Robots* 11(3), 221–226.
- Lewis, M. A., Bekey, G. A. (2002). Gait Adaptation in a Quadruped Robot. *Autonomous Robots* 12(3), 301–312.
- Manoonpong, P., Geng, T., Wörgötter, F. (2006). Exploring the dynamic walking range of the biped robot “Runbot” with an active upper-body component. *Proceedings of the Sixth IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006)*, 418–424.
- Manoonpong, P., Geng, T., Kulvicius, T., Porr, B., Wörgötter, F. (2007). Adaptive, Fast Walking in a Biped Robot under Neuronal Control and Learning. *PLoS Computational Biology* 3(7), e134.
- Manoonpong, P., Wörgötter, F. (2009). Efference Copies in Neural Control of Dynamic Biped Walking. *Robotics and Autonomous Systems*, Elsevier Science, 57(11), 1140–1153.
- Miyakoshi, S. (2006). Bipedal Walking with a Memory-based Motion Controller. *Journal of the Robotics Society of Japan*, 24(5), 623–631 (in Japanese).
- Namiki, A., Hashimoto, K., Ishikawa, M. (2003). A hierarchical control architecture for high-speed visual servoing. *International Journal of Robotics Research* 22(10-11), 873–888.
- Nissen, S. (2003). Implementation of a Fast Artificial Neural Network Library (fann). Report: *Department of Computer Science, University of Copenhagen (DIKU)*. Software available at <http://www.sourceforge.net/projects/fann>
- Ogino, M., Toyama, H., Fuke, S., Mayer, N. M., Watanabe, A., Asada, M. (2008). Compliance Control for Biped Walking on Rough Terrain. *RoboCup 2007, LNAI 5001*, 556–563.
- Porr, B., Wörgötter, F. (2003). Isotropic sequence order learning in a closed loop behavioural system. *Roy Soc Phil Trans Mathematical Physical Engineer Sci* 361, 2225–2244.
- Russo, P., Webb, B., Reeve, R., Arena, P., Patané, L. (2005). 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*, Sevilla, Spain, 227–232.
- Schröder-Schetelig, J., Manoonpong, P., Wörgötter, F. (2008). Using efference copy and neural control for adaptive walking on different terrains. *Frontiers in Computational Neuroscience. Conference Abstract: Bernstein Symposium 2008*. doi: 10.3389/conf.neuro.10.2008.01.115
- Vukobratovic, M., Borovac, B., Surla, D., Stokic, D. (1990). *Biped Locomotion-Dynamics, Stability, Control and Application*. Springer-Verlag.

A Appendix

Table 2 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 (compare Fig. 3).

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	