Training a network of electronic neurons for control of a mobile robot

International Journal of Bifurcation and Chaos, Vol. 26, No. 12 (2016) 1650196 (16 pages)

DOI: 10.1142/S0218127416501960

Training a Network of Electronic Neurons for Control of a Mobile Robot

T. G. M. Vromen
Department of Mechanical Engineering, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands
t.g.m.vromen@tue.nl

E. Steur
Institute for Complex Molecular Systems, Department of Mechanical Engineering, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands
e.steur@tue.nl

H. Nijmeijer
Department of Mechanical Engineering, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands
h.nijmeijer@tue.nl

Received October 29, 2015; Revised February 26, 2016

An adaptive training procedure is developed for a network of electronic neurons, which controls a mobile robot driving around in an unknown environment while avoiding obstacles. The neuronal network controls the angular velocity of the wheels of the robot based on the sensor readings. The nodes in the neuronal network controller are clusters of neurons rather than single neurons. The adaptive training procedure ensures that the input–output behavior of the clusters is identical, even though the constituting neurons are nonidentical and have, in isolation, nonidentical responses to the same input. In particular, we let the neurons interact via a diffusive coupling, and the proposed training procedure modifies the diffusion interaction weights such that the neurons behave synchronously with a predefined response. The working principle of the training procedure is experimentally validated and results of an experiment with a mobile robot that is completely autonomously driving in an unknown environment with obstacles are presented.

Keywords: Diffusive neuronal cell network; adaptive training procedure; practical synchronization; autonomous mobile robot control.

1. Introduction

Autonomous navigation of robots has a large number of application areas such as automatic driving, transporting objects in factory or office environments, and unmanned exploration of dangerous regions, see e.g. [Antonelo *et al.*, 2006; Zhang *et al.*, 1997; Guivant *et al.*, 2000]. Roughly speaking, autonomous navigation of mobile robots can be based on prior path planning or as a direct mapping of sensory input to actions [Antonelo *et al.*, 2008]. The methods based on prior path planning prove the existence of an optimal path, but require complete knowledge about the location, orientation and movements of the obstacles in the environment

[Floreano & Mondada, 1998]. Such detailed information is not available in many real world applications. Furthermore, these methods assume the environment not to change in time, whereas real world environments are prone to changes, which limits the applicability of prior path planning methods in such applications [Chatterjee & Matsuno, 2001].

On the contrary, neuro/biological networks are able to learn from previous experiences and search for an optimal solution in unknown situations. Such networks are thus a natural candidate to use for control of autonomous robots that have to operate in unknown real world environments. An additional advantage of such controllers over conventional digital processing using a microcontroller is the improved robustness with respect to possible faults; biologically-inspired controllers may work correctly if parts of the hardware are damaged, which is in contrast to a digital computer, where a small error may lead to catastrophic results [Wilamowski, 2003].

Multiple examples of neuro/biological networks used for control purposes can be found in literature. Before we discuss some examples, we make a clear distinction between what we call neural networks and neuronal networks. A neural network is understood as a network where the nodes are static input–output maps that are connected via weighted couplings, whereas the nodes in neuronal networks, like spiking neural networks, are dynamical systems. Some approaches using neur(on)al networks for control purposes are discussed here. For example, in [Manoonpong *et al.*, 2005] a four-legged walking machine is controlled by a neural network, which enables the robot to walk around and avoid obstacles. This modular controller consists of different smaller networks to carry out different tasks like, input processing and velocity control. Another neural controller is proposed in [Massa *et al.*, 2006] and is used to control a unicycle-type mobile robot, such that it can avoid obstacles and find a light source. The two subtasks (obstacle avoidance and localization of the light source) are solved by their own part of the network.

Examples of neuronal networks for control of mobile robots are found in, for example, [Pearson *et al.*, 2007; Floreano *et al.*, 2006; Wang *et al.*, 2008; Johnston *et al.*, 2010]. In particular, in [Pearson *et al.*, 2007] a controller has been designed to be employed on mobile robot vehicles using an FPGA approach, allowing bio-inspired neuronal processing models to be integrated directly into control environments. To reduce computational cost, relatively simple leaky-integrate-and-fire neurons are used. The neuronal network proposed in [Floreano *et al.*, 2006] uses integrate-and-fire neurons as its constituting units as well. The angular velocities of the motors are determined by the motor neurons, which receive input from excitatory and inhibitory neurons whose response depends on the signals received from sensor neurons. A similar spiking neural network for control of a mobile robot can also be found in [Wang *et al.*, 2008], which proposes a network with three different layers. In the input layer, the membrane potential of the sensor neurons is influenced by the sensor readings. Furthermore, the turning neurons and approximate neuron judge whether an opposite obstacle is too close. These neurons are coupled to the hidden neurons which determine to turn left or right. The hidden neurons are then coupled to the motor neurons in the output layer, whereas the sensor neurons are directly coupled to the motor neurons, which determine the angular velocities of the driving wheels of the mobile robot.

We present a network with coupled electronic neurons for control of a mobile robot with a network architecture that is inspired by the three-layer spiking neural network proposed in [Wang *et al.*, 2008]. Proper functioning of the network of electronic neurons, which we will refer to as the robot's electronic brain or simply electronic brain, requires the constituting nodes to produce identical responses to the same input. However, in any hardware implementation, the manufacturing tolerances (of components) and noise make the neurons behave nonidentically. We propose to use *clusters of neurons* instead of single neurons as nodes in the neuronal network controller and we train the clusters such that (1) the outputs of the neurons in a cluster are synchronized, and even more importantly, (2) the synchronized outputs of the clusters are (practically) indistinguishable when the same input is applied. More precisely, we propose a procedure to construct and train clusters by (re)defining the network structure (interaction weights) of a cluster. Thus we may say that our training procedure yields a robustly synchronized cluster. For a recent survey on (robust) synchronization and its applications we refer to [Tang *et al.*, 2014]. An additional advantage of this approach is that in the extreme case that

a neuron of the electronic brain fails, or, expressed dramatically, dies, a brief period of retraining (rehabilitation) makes the electronic brain function properly again.

In this paper the focus is on the training procedure of the clusters. We develop and illustrate this training procedure using clusters of neurons of the Hindmarsh–Rose type [Hindmarsh & Rose, 1984] and we verify our results of training using experiments. The experimental setup consists of an e-puck mobile robot [Mondada *et al.*, 2009], equipped with the neuronal controller presented in [Wang *et al.*, 2008] with electronic circuit realizations of the Hindmarsh–Rose neuron [Steur *et al.*, 2008; Neefs, 2009; Neefs *et al.*, 2010] as nodes of the neuronal network. We have chosen to use Hindmarsh–Rose model neurons as constituting units as electronic circuit realizations of this model are available to us. We remark that our results will not be limited to the use of this particular model neuron; previously obtained theoretical results on synchronization of neurons reported in [Steur *et al.*, 2009] imply that the training procedure will also work when the Hindmarsh–Rose model neurons are replaced by FitzHugh–Nagumo model neurons, Morris–Lecar model neurons or even Hodgkin–Huxley model neurons.

This paper is organized as follows. In Sec. 2, we introduce the mobile robot and the neuronal controller. Section 3 presents the electronic Hindmarsh–Rose neuron. Next, in Sec. 4, we introduce some theoretical results on (practical) synchronization of asymmetrically coupled, nonidentical Hindmarsh–Rose neurons. These theoretical results justify the training procedure that we describe next. The results of the experiments of the training procedure and the mobile robot controlled by the neuronal network are presented in Sec. 5. Finally, in Sec. 6 conclusions are drawn.

2. Neuronal Controller for a Mobile Robot

2.1. *Mobile robot*

The mobile robot used in the experiments is the e-puck mobile robot [Mondada *et al.*, 2009], which is shown in Fig. 1. The position and orientation of the robot are defined in a two-dimensional Cartesian space (x, y) with origin O, see Fig. 2. The position of the robot in the Cartesian coordinate system is given by the following equations

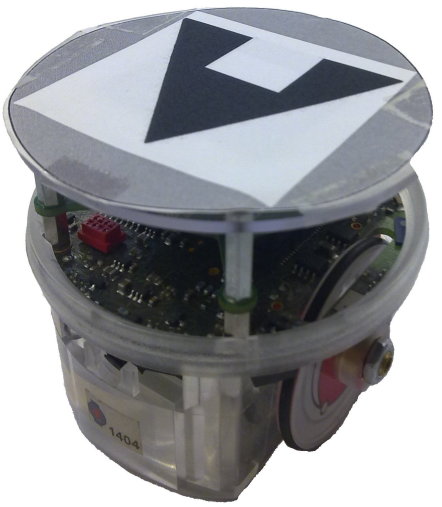

Fig. 1. The e-puck mobile robot.

$$\begin{aligned} \dot{x}(t) &= v(t)\cos(\theta(t)), \\ \dot{y}(t) &= v(t)\sin(\theta(t)), \\ \dot{\theta}(t) &= \omega(t), \end{aligned} \tag{1}$$

where $v(t)$ and $\omega(t)$ are the translational and rotational velocities of the robot, respectively. The forward velocity $v(t)$ and rotational velocity $\omega(t)$ of the robot satisfy the relations

$$\begin{aligned} v(t) &= \frac{r}{2}(\omega_r(t) + \omega_l(t)), \\ \omega(t) &= \frac{r}{2b}(\omega_r(t) - \omega_l(t)), \end{aligned} \tag{2}$$

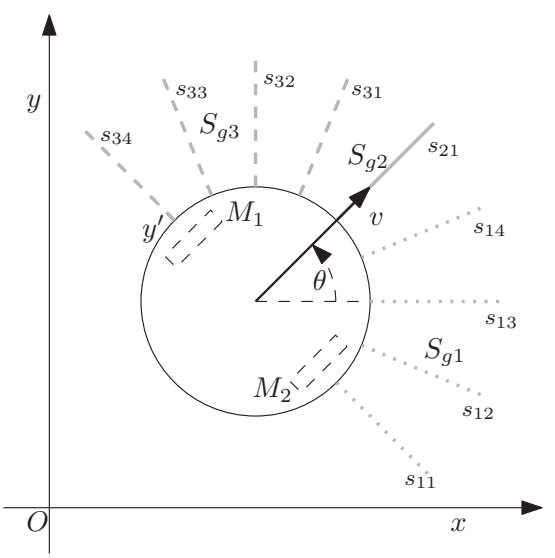

Fig. 2. Illustration of the mobile robot and the sensor layout. The dashed rectangles represent the wheels of the robot.

where ω_l and ω_r are the angular velocities of the left, and right wheel, respectively, r is the radius of the wheels and b is half the distance between the wheels. The angular velocities $\omega_r(t)$ and $\omega_l(t)$ can be changed independently, which allows the robot to move to every position in the (x, y) plane. Note, however that the robot cannot move sideways, i.e. perpendicular to the forward motion, because of a nonslip condition (nonholonomic constraint, cf. [Bloch, 2003]) on the wheels; an arbitrary position in the (x, y) plane is reached by the robot by moving back and forth and turning around its center.

The mobile robot is equipped with nine sensors for detection of obstacles and to compute the distance of the robot to obstacles, as shown in Fig. 2. The sensors are divided in three sensor groups S_{gj}, with $j = 1, 2, 3$, where sensors in groups 1 and 3 detect obstacles to the left and right of the robot respectively, and the sensor in S_{g2} finds obstacles in front of the robot. The distance to an object measured by sensor k in group j is denoted by s_{jk}. The minimal and maximal distances that can be detected by a sensor are denoted by $x_{\min}$ and $x_{\max}$, respectively. The distance to an object detected by sensor group j is given by

$$d_{gj} = \min_k(s_{jk}). \tag{3}$$

These distances d_{gj} are available to the controller. The control inputs of the mobile robot are the angular velocities ω_l and ω_r of the two driving wheels.

2.2. *Neuronal controller*

The task of the robot is relatively simple; it has to maneuver in an (unknown) environment and avoid collisions with all objects in the environment. In addition, when the object is right in front of the robot, it has to come to a standstill. We remark that we used this application as a proof of principle, therefore we assumed that the angular velocities of the wheels can only be positive, i.e. $\omega_l, \omega_r \geq 0$. If one would allow the wheels to have negative angular velocities the robot would be able to turn on its axis to turn around instead of making a complete stop when an object is right in front of the robot.

The mobile robot will achieve these tasks using a neuronal controller with network structure as shown in Fig. 3, which is based on the neuronal controller presented in [Wang *et al.*, 2008]. The nodes in the input layer, which we call sensor neurons, receive external inputs I_{gj}, $j = 1, 2, 3$. For a constant input I_{gj} the sensor neurons fire periodically

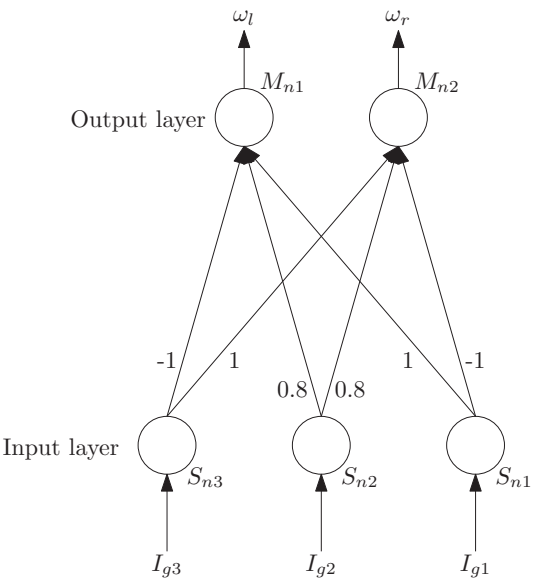

Fig. 3. Network structure of the neuronal controller.

and the larger I_{gj}, the higher the firing rate. The inputs I_{gj} are chosen to be (exponentially) proportional to the distances d_{gj} measured by the sensor group, according to the (saturated) function

$$I_{gj} = \begin{cases} I_{\min} & \text{if } d_{gj} \leq x_{\min} \\ I_{\min}\left(\dfrac{I_{\max}}{I_{\min}}\right)^{\frac{d_{gj}-x_{\min}}{x_{\max}-x_{\min}}} & \text{if } x_{\min} < d_{gj} < x_{\max} \\ I_{\max} & \text{if } d_{gj} \geq x_{\max} \end{cases} \tag{4}$$

with $I_{\min}$ and $I_{\max}$ the minimum and maximum input voltages, respectively.

The sensor neurons couple to the neurons in the output layer, which we refer to as motor neurons, as the outputs of these neurons directly affect the angular velocities of the wheels of the robot. In particular, the firing rate of motor neuron M_{n1} (M_{n2}) is proportional to the angular velocity of the left (right) wheel. Motor neuron M_{n1} receives an excitatory input from sensor neurons S_{n1} and S_{n2}, and an inhibitory input from sensor neuron S_{n3}. Similarly, motor neuron M_{n2} receives an excitatory input from sensor neurons S_{n3} and S_{n2}, and an inhibitory input from sensor neuron S_{n1}. The weights of the inhibitory and excitatory inputs from S_{n3} and S_{n1} are identical whereas the weight of the excitatory input from S_{n2} is a little less. In more detail, the activating potential $U_i(t)$, $i = 1, 2$, of the

motor neurons is defined by the relation

$$U_i(t) = \sum_{j=1}^{3} \sum_{N_j^f} W_{Sn}(i,j)\psi\Big(t - t_j^f\Big). \tag{5}$$

Here t_j^f is the spike emitting time of the jth sensor neuron in a predefined time window and N_j^f denotes the number of spikes of that neuron in that time window. The coupling gain matrix W_{Sn} from the sensor neurons to the motor neurons which is, in accordance to Fig. 3, is given by

$$W_{Sn} = \begin{pmatrix} 1 & 0.8 & -1 \\ -1 & 0.8 & 1 \end{pmatrix}. \tag{6}$$

The function $\psi(\cdot)$ is given by

$$\psi(s) = \frac{s}{\tau_s} \exp\left(-\frac{s}{\tau_s}\right), \tag{7}$$

with τ_s a time constant. This function ψ introduces "exponential forgetting" such that recent spikes result in a larger increase of the activating potential than previous spikes in the time window. The generation of a spike for the motor neuron can now be determined by the membrane potential depending on the activating potential. The membrane potential $V_i(t)$ of the ith motor neuron is given by

$$V_i(t) = \begin{cases} 2, & \text{if } U_i(t) \geq V_M, \\ U_i(t), & \text{if } U_i(t) \geq V_{\text{rest}} \text{ and } t - t_i^f \geq \delta_t, \\ V_{\text{rest}}, & \text{otherwise}, \end{cases} \tag{8}$$

where V_{rest} is the resting potential, t_i^f is the spike emitting time of motor neuron i and δ_t is the refractory period. A spike is emitted if $V_i = 2$ and the number of spikes n_i in a predefined time window is proportional to the angular velocities of the driving wheels, hence the forward and angular velocities of the robot. Note that the number of spikes in the time window is limited by the refractory period, i.e. the minimum time between two successive spikes. The angular velocities of the left and right wheels are given by $\omega_l = n_1 p$ and $\omega_r = n_2 p$ respectively, with p a constant determining the angular velocity per outputting pulse in radians per second. The values of the parameters as well as the lengths of the time-windows will be provided in Sec. 5.

It follows that this particular neuronal controller has the desired features:

(1) if an obstacle is detected on the left, or if an obstacle on the left is closer to the robot than an obstacle on the right, then the input to motor neuron M_{n1} is larger than the input to M_{n2} and the robot makes a turn to the right;
(2) the robot will move forward at almost maximal speed in case that there are no obstacles detected by any sensor neuron or in case both S_{n1} and S_{n3} detect an obstacle at the same distance, e.g. when the robot moves straight through a channel;
(3) if the robot is moving perpendicular to a wall, then it comes to a standstill.

3. Electronic Hindmarsh–Rose Neuron

As mentioned in the introduction, the nodes in our neuronal controller consists of clusters of Hindmarsh–Rose (HR) model neurons. The Hindmarsh–Rose model neuron is described by the following system of coupled nonlinear differential equations

$$\begin{aligned} \dot{y} &= -y^3 + 3y + 5z_1 - z_2 - 8 + I, \\ \dot{z}_1 &= -y^2 - 2y - z_1, \\ \dot{z}_2 &= 0.005(4y + 4.472 - z_2), \end{aligned} \tag{9}$$

where $\dot{} := \frac{\mathrm{d}}{\mathrm{d}t^*}$, $t^* = 1000t$ with t the time in seconds, y denotes the membrane potential of a neuron, which also serves as the natural output of the neuron, z_1, z_2 are internal variables and I is the input. The time scaling factor $\frac{t^*}{t} = 1000$ is in accordance with the electronic realization of the HR model as

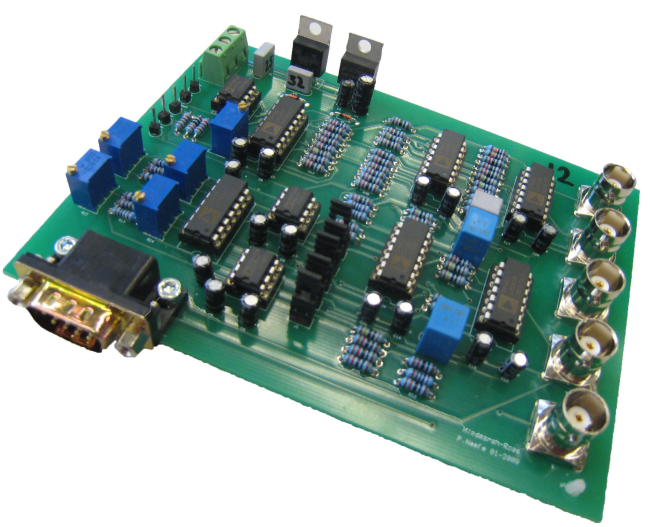

Fig. 4. Electronic HR model neuron.

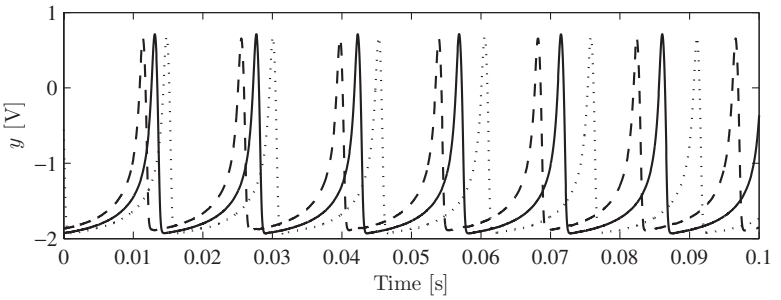

Fig. 5. Response of three different electronic HR-cells, each driven by the input $I = 4.5$ [V].

presented in Fig. 4. However, from the analysis point of view any positive time-scaling factor may be chosen. (Of course, one may need to determine different parameter values for the neuronal controller.) Depending on the value of the synaptic current I, the HR model is able to produce several dynamical spiking patterns like (chaotic) bursting and tonic spiking, cf. [Hindmarsh & Rose, 1984]. As for our purpose we rely on the existence of a well-defined firing rate,[1] hence we let the HR neurons operate in the tonic spiking regime, which is the case for synaptic inputs in the range $4 < I < 12$.

An electronic circuit board realization of the HR model neuron, shown in Fig. 4, is available for experiments. This electronic HR model neuron consists of three integrating circuits, which integrate the three states of the HR model (9) and two multiplier circuits that generate the squared and cubic terms of the y-state. For a detailed description of the electronic neurons, the reader is referred to [Neefs, 2009; Neefs *et al.*, 2010].

The responses of the electronic cells are very similar to the signals obtained by numerical integration of the system equations. This means that the shape of the spikes, timing and range of the signals qualitatively agree. Total similarity cannot be expected since slight imperfections in the realization and measurement noise are present. Due to the small differences in the off-the-shelf components of the electronic circuits, the measured responses do not only differ from the numerical simulations, but the electronic cells also show mutual differences. Measured responses for three neurons with equal input I are shown in Fig. 5. It can be seen that the time between two consecutive spikes differs for all three neurons. This becomes more evident from Fig. 6, which shows the difference in period time, i.e. the time between consecutive spikes, of these three electronic neurons as function of the synaptic input. It is shown in [Neefs, 2009] that the non-identical behavior of the electronic HR neurons is explained, to a very large extent, by the mismatches in the constant parameters. Thus a careful selection of components or even some tuning of the circuits might decrease these problems, but one cannot expect that the circuits will become identical.

To illustrate the problems that can arise if non-identical neurons are used in the neuronal network, a simple experiment is performed. In this experiment we use single electronic neurons as nodes in our neuronal controller. Recall that the input I_{gj} for

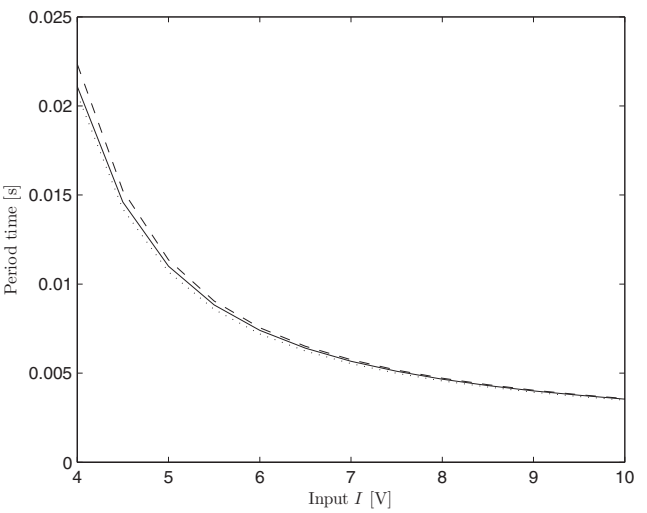

Fig. 6. Period time of three electronic HR-cells as function of the neuron input I.

[1]The firing rate of a HR neuron is determined by the number of spikes it produces in a predefined time window; the more spikes appearing in the time window, the larger the firing rate. Here a spike is defined by the rapid rise and fall of the neuron's membrane potential, and the spike emitting time is defined as the time at which, during a spike, the membrane potential exceeds the firing threshold the first time.

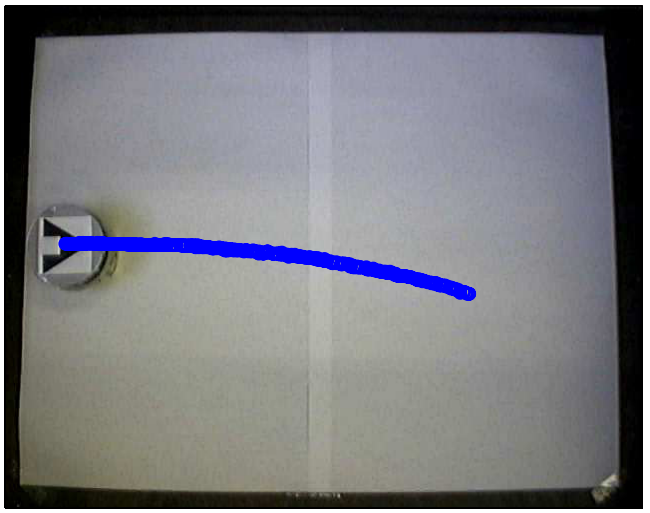

Fig. 7. Experimental result, the robot does not detect any obstacles and should drive in a straight line, but due to differences between the sensor neurons it has a preferred direction.

each electronic neuron S_{nj}, $j = 1, 2, 3$, is monotonically increasing with the measured distance (d_{gj}) between the robot and an obstacle. If, for example, the left and right sensors both measure an equal distance to an obstacle, the input for sensor neurons 1 and 3 is the same and the robot should drive in a straight line. However, due to (small) differences in the neurons, the spiking frequency of different neurons will not be equal for this equal input and therefore the robot will make a turn rather than moving straight ahead. An experiment result showing this behavior is shown in Fig. 7. The differences between neurons will be present in every practical problem, e.g. also in nature, two cells with the same function will respond slightly differently. We propose to use a small network (or cluster) of electronic neurons instead of single neurons as a solution to the problem discussed above, therewith improving the robustness of the neuronal controller. A training procedure, presented in the next section, will ensure that the responses of the clusters of neurons will be identical when stimulated by the same input I_{gj}.

4. Training of Clusters of HR Neurons

Recall that the training procedure should ensure that, for every input I supplied to all neurons,

(1) the outputs of the electronic neurons within a cluster need all to be practically synchronized, i.e. the difference between the output of all neurons in a cluster should be small;

(2) the firing rate of the practically synchronized neurons within a cluster should be practically identical among the clusters.

Because the dynamics of the electronic HR neurons cannot be changed (because that would require replacing some of the components of the circuit), we couple the neurons and modify the interaction weights to achieve the desired behavior. Before we introduce our procedure for tuning of the interaction weights we have to introduce some notation and give a result on practical synchronization. It will become clear later in this section that it suffices to consider the practical synchronization problem of only two coupled neurons. Implementation of the training procedure and quantification of the practical synchronization error in the output ϵ are presented in Sec. 5.

We let the ith electronic HR neuron be described by the equations

$$\begin{cases} \dot{y}_i = -\alpha_{i,1} y_i^3 + \alpha_{i,2} y_i + \alpha_{i,3} z_{i,1} \\ \qquad - \alpha_{i,4} z_{i,2} - \alpha_{i,5} + \alpha_{i,6} I + \alpha_{i,7} u_i, \\ \dot{z}_{i,1} = -\alpha_{i,8} y_i^2 - \alpha_{i,9} y_i - \alpha_{i,10} z_{i,1}, \\ \dot{z}_{i,2} = \alpha_{i,11} y_i + \alpha_{i,12} - \alpha_{i,13} z_{i,2}, \end{cases} \tag{10}$$

with positive parameters $\alpha_{i,j}$ that deviate from the nominal parameters $\overline{\alpha}_j$ by an amount $\delta_{i,j}$, $j = 1, \ldots, 13$. Here the nominal parameters are

$$\begin{aligned} &\overline{\alpha}_1 = 1, \quad \overline{\alpha}_2 = 3, \quad \overline{\alpha}_3 = 5, \quad \overline{\alpha}_4 = 1, \\ &\overline{\alpha}_5 = 8, \quad \overline{\alpha}_6 = 1, \quad \overline{\alpha}_7 = 1, \quad \overline{\alpha}_8 = 1, \\ &\overline{\alpha}_9 = 2, \quad \overline{\alpha}_{10} = 1, \quad \overline{\alpha}_{11} = 0.02, \\ &\overline{\alpha}_{12} = 0.005 \cdot 4.472, \quad \overline{\alpha}_{13} = 0.005. \end{aligned}$$

In the Appendix it is shown that two of these non-identical HR neurons, which interact via

$$u_1 = \gamma\sigma(y_2 - y_1), \tag{11}$$

$$u_2 = \gamma(1 - \sigma)(y_1 - y_2), \tag{12}$$

with coupling strength $\gamma > 0$ and any parameter $\sigma \in (0, 1)$, practically output-synchronize when γ is sufficiently large. That is, for γ sufficiently large there is a $\epsilon = \epsilon(\gamma)$ sufficiently small, for each $\epsilon^* > \epsilon$ there exists a $T = T(\epsilon^*) > 0$ such that $\|y_1(t) - y_2(t)\| < \epsilon^*$ for all $t \geq t_0 + T$. In addition, $\epsilon(\gamma)$ is a monotonically decreasing function of γ with $\epsilon \to 0$ as $\gamma \to \infty$.

4.1. *Training algorithm for neuronal clusters*

To explain the training procedure, consider a network of two coupled HR neurons coupled as given in (11) and (12). As shown in the previous section we know that the two coupled nonidentical HR neurons practically output-synchronize for sufficiently strong coupling. Let us first explain the role of parameter σ. Consider the network of two coupled HR neurons as schematically shown in Fig. 8. For $\sigma = \frac{1}{2}$ the neurons are symmetrically coupled, that is neuron 1 influences neuron 2 just as much as neuron 2 influences neuron 1. But as $\sigma \to 1$, neuron 2 couples to neuron 1 while neuron 1 does not couple to neuron 2 ($u_2 \approx 0$). In other words, for $\sigma \to 1$ the coupling configuration is of the master-slave type where neuron 1 is enslaved to neuron 2. That implies that for sufficiently strong coupling the output of neuron 1 practically synchronizes to the output of neuron 2, hence the period time of neuron 1 will adapt to the period time of neuron 2. Of course, if $\sigma \to 0$ and the coupling is sufficiently strong, neuron 2 is enslaved to neuron 1 and the period time of the practically synchronized neurons will equal the period time of neuron 1. A simulation result that illustrates this concept is shown in Fig. 9.

These results imply that the parameter σ is an interpolation parameter that allows the period time of the two practically output-synchronized HR neurons to be anywhere between the period times of the uncoupled neurons. This interpolation is shown in Fig. 10, which indeed indicates that the period time of the cluster of two neurons can be anywhere between the period times of the two uncoupled coupled neurons.

The key idea of the training procedure is to choose a reference neuron and train the clusters of neurons by varying the coupling strength γ and σ in such a way that for every input I the period time of the cluster of practically output-synchronized neurons matches the period time of the reference neuron. However, in case we consider a cluster consisting of only two neurons, it is not unlikely that the

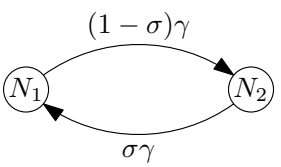

Fig. 8. Two neurons coupled with asymmetrical coupling.

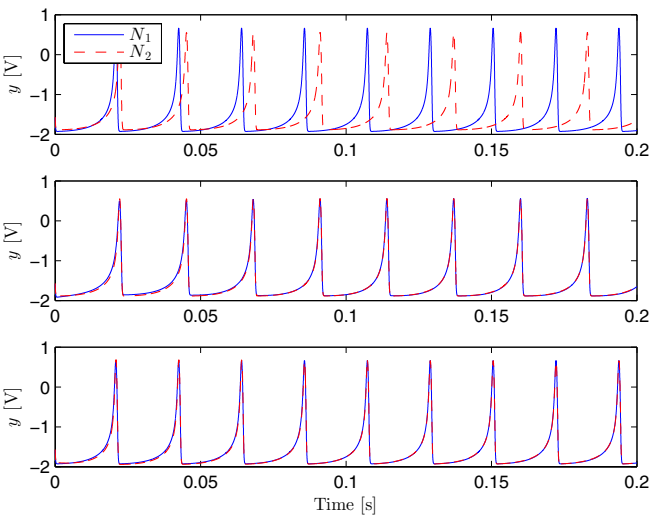

Fig. 9. Top plot: Simulation result of a slow neuron and a fast neuron, bottom plots: master-slave configurations of these neurons.

period times of both neurons are larger/lower than that of the reference neuron; increasing the number of neurons per cluster increases the probability that the period time of the reference neuron is in between the period time of the uncoupled neurons in that cluster such that, by defining appropriate coupling between the neurons, the period time of the reference can be matched.

The following example with a cluster consisting of three neurons illustrates the training procedure we propose:

- Pick two neurons in the cluster and couple them according to (11) and (12);

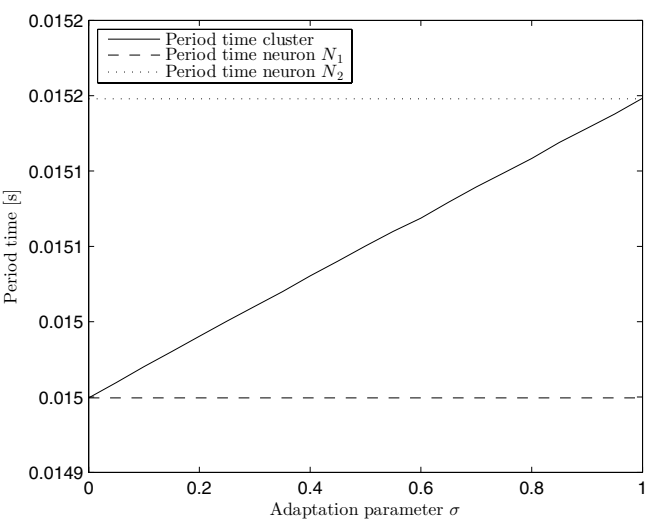

Fig. 10. Numerically determined period time of the cluster as function of the adaptation parameter σ.

- Fix $\sigma = \sigma_1 = \frac{1}{2}$ and increase $\gamma = \gamma_1$ until the two coupled neurons practically output-synchronize with a given bound ϵ;
- Change the value of σ_1 in such a way that the period time of the cluster of practically output-synchronized neurons is as close as possible to the period time of the reference neuron;
- Fix σ_1 and γ_1 and couple a third neuron to the two coupled neurons according to

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = -\Gamma_3(\sigma_1, \sigma_2, \gamma_1, \gamma_2) \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

with matrix

$$\Gamma_3(\sigma_1, \sigma_2, \gamma_1, \gamma_2) = \begin{pmatrix} \gamma_1\Gamma_2(\sigma_1) + \sigma_2\gamma_2\mathbf{I} & -\gamma_2\sigma_2 \\ & -\gamma_2\sigma_2 \\ -\gamma_2(1-\sigma_2) - \gamma_2(1-\sigma_2) & 2\gamma_2(1-\sigma_2) \end{pmatrix};$$

- Fix $\sigma_2 = \frac{1}{2}$ and increase γ_2 until the three coupled HR neurons practically output-synchronize within the bound ϵ;
- Change the value of σ_2 in such a way that the period time of the cluster of three practically output-synchronized neurons is as close as possible to the period time of the reference neuron.

An illustration of adding the third neuron to the cluster of two practically output-synchronized neurons is given in Fig. 11. Of course this procedure can be repeated to create a cluster with $p > 3$ practically synchronized neurons whose period time is as close to the period time of the reference neuron as possible; our training procedure is summarized in the flowchart in Fig. 12.

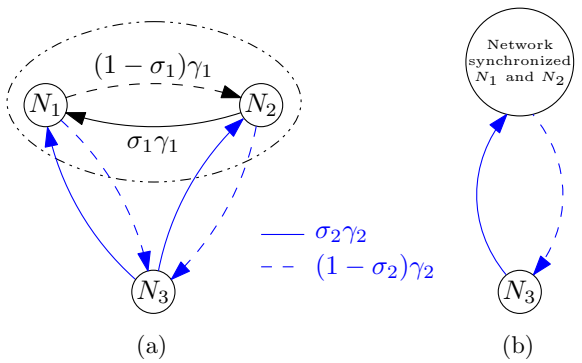

Fig. 11. Cluster of three neurons: (a) Output-synchronized cluster of two neurons asymmetrically coupled with neuron N_3 and (b) illustrating the same structure as for two neurons.

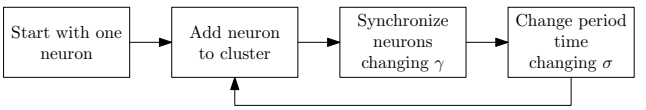

Fig. 12. Flowchart of the steps in the training algorithm.

Formally: starting with a cluster of two neurons, the coupling between the neurons is modified such that they become practically output-synchronized. Then the practically output-synchronized cluster of two neurons can be considered as a single neuron whose period time is compared with the period time of the reference neuron. The next step is to change the coupling weights between the output-synchronized neurons within the cluster such that the period time of the cluster, T_{cluster}, becomes equal to T_{ref}, the period time of the reference neuron. Once the training procedure for the cluster of two neurons is finished, that is, (14) is fulfilled, the coupling weights of that cluster are fixed such that the cluster can now be considered as a single neuron. Then another neuron is added and the procedure is repeated.

The training procedure starts with the choice of the reference neuron. As mentioned before, the period time of the trained clusters needs to be equal to the period time of this reference neuron whenever the input I is the same. As discussed above, the proposed training procedure produces a period time of the cluster that is an interpolate of the period times of the individual neurons within that cluster, hence the period time of the reference neuron has to be somewhere in between the period times of the uncoupled neurons. We choose the reference neuron to have the period time as close as possible to the mean of the period times of the uncoupled neurons.

The next step in the training procedure is to determine the coupling strength between two neurons, or an output-synchronized cluster with multiple neurons and the neuron to be added, for which there is practical output-synchronization. Therefore we start at $t = t_0$ with a symmetric network ($\sigma = \frac{1}{2}$) and initial coupling strength $\gamma = 0$, which is increased with an increment $\Delta\gamma$ that depends on the maximal output-synchronization error on a nonempty interval $[t_1, t_2]$,

$$\Delta\gamma = \begin{cases} 0 & \text{if } \sup_{\tau\in[t_1,t_2]} |y_i(\tau) - y_j(\tau)| < \epsilon \\ \dfrac{\alpha}{m-1} \sup_{\tau\in[t_1,t_2]} |y_i(\tau) - y_j(\tau)| & \text{otherwise} \end{cases} \quad (13)$$

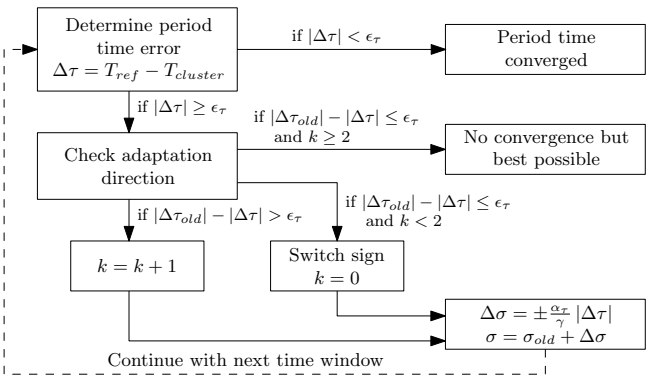

Fig. 13. Flowchart of the procedure to change the period time of the cluster.

where $\alpha > 0$ is a parameter that controls the maximum of $\Delta\gamma$ and $m \geq 2$ denotes the current number of neurons in the cluster. The interval $[t_1, t_2]$ is chosen in such a way that all transients have died out. We set $\gamma = \gamma + \Delta\gamma$ and repeat the process until the neurons practically output-synchronize. Recall that we can always get practical output-synchronization for γ sufficiently large.

When the neurons in the cluster are practically output-synchronized the period time adaptation can start. The period times of the reference neuron T_{ref} and the cluster of neurons T_{cluster} are determined by counting the number of spikes in a predefined time-window. Then we check if the difference in period time of the cluster and the period time of the reference neuron, $\Delta\tau = T_{\text{cluster}} - T_{\text{ref}}$, is sufficiently small, that is

$$|\Delta\tau| < \epsilon_\tau, \tag{14}$$

with $\epsilon_\tau > 0$ a sufficiently small, predefined bound. If this is not the case, the period time of the cluster is adapted by changing the adaptation parameter σ. The adaptation is obtained according to a stepwise change $\Delta\sigma$ of the adaptation parameter, given by

$$\Delta\sigma = \pm\frac{\alpha_\tau}{\gamma}|\Delta\tau| \tag{15}$$

with $\alpha_\tau > 0$ and $\Delta\tau = T_{\text{ref}} - T_{\text{cluster}}$ the difference in period time between the reference neuron and the cluster. Since it is not known beforehand which neuron is "fast" and which neuron is "slow", the correct sign of $\Delta\sigma$ is initially undetermined. Therefore we start with a positive sign, and we change sign if the period time of the cluster starts to deviate more from that of the reference neuron. If the period time is converged, i.e. $|\Delta\tau| < \epsilon_\tau$, a new neuron can be added to the cluster according to Fig. 12. However, it is possible that the sign of adaptation direction is correct but the reference period cannot be reached because both the cluster and the added neuron are "faster" or "slower" than the reference neuron. In that case, if after k steps the period time of the cluster is not converged within ϵ_τ bound of the reference neuron, the σ adaptation procedure is stopped and new neuron is added to the cluster. The adaptation procedure of σ is summarized in the flowchart in Fig. 13.

5. Results of Experiments

5.1. *Experimental setup*

The experimental setup, shown in Fig. 14, consists of a personal computer, a data acquisition device, a coupling interface and the electronic neurons which are placed in three different stacks representing the clusters in the input layer of the neuronal controller. Each stack contains an input port for the external input I and a power connection. For each neuron the y-state can directly be measured and the coupling signal u_i can be supplied. The neurons interact via the coupling interface in which any desirable coupling structure can be specified. More information about the coupling interface can be found in [Neefs *et al.*, 2010]. The PC and data acquisition device (DAQ) are used to log, display and process the measured data. Moreover, the neuron inputs and coupling strengths are sent to the electronic neurons and the coupling interface, respectively. For this setup, a National Instruments$^{\text{TM}}$ PCIe-6363 multifunction data acquisition device is used, which consists of a PCI Express card and two connector blocks that can measure up to 32 analog signals and has four analog outputs. The signals are measured sequentially, but due to the high sample rate compared to the timescales of the system, the effect

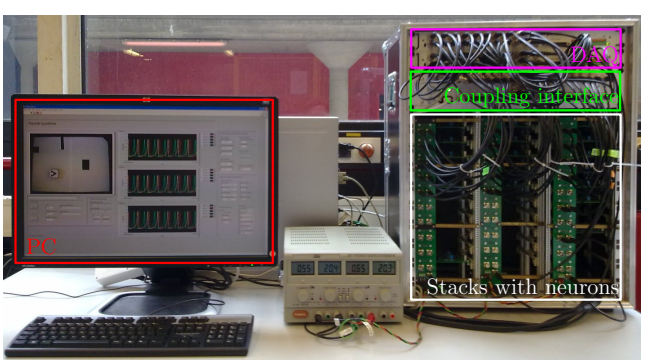

Fig. 14. Photograph of the experimental setup.

of sequential sampling is negligible. The program to send and receive data from the neurons, as well as the training algorithm are written in Labview^{TM}. Furthermore, a VGA-camera is used to make images of the mobile robot and the operating environment. These images are analyzed using image processing software, to detect the robot and to determine its position and orientation in the environment. Several detection lines, simulating the sensors, are defined. These detection lines are used to detect an obstacle in the specified directions based on the camera image. In addition, the distance to a possible obstacle can be measured. The robot is controlled by the angular velocity of the two driving wheels, the current velocities, determined by the neuronal controller, are sent to the mobile robot via a bluetooth connection.

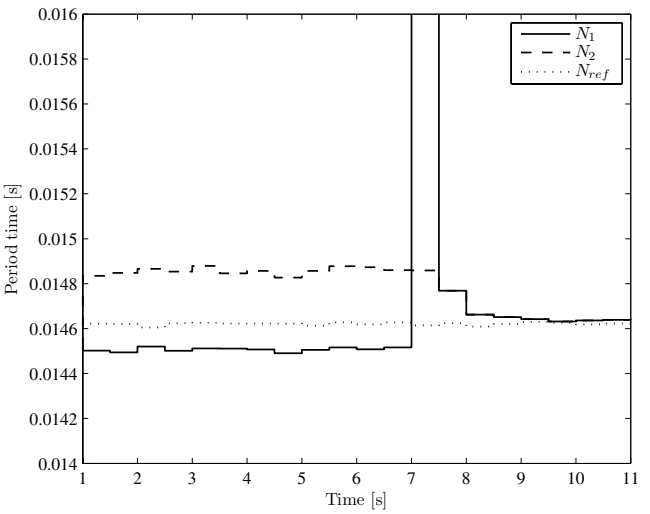

Fig. 15. Experimental result of the adaptation mechanism for a cluster of two neurons.

5.2. *Training procedure results*

As mentioned before, we need to specify some time windows to compute spike rates, output-synchronization errors and period time errors. The length of these time windows is chosen to be 500 ms. For the training procedure, the first 100 ms of each time window are used to send and receive the (new) coupling gains from the PC to the coupling interface. The next period (275 ms) within the time window is used to adapt to the new settings and only the last part (125 ms) is used for measurements. These measurements include the determination of the synchronization error for the synchronization procedure and the error in the period time for the period time adaptation. The parameter values for the training algorithm used during the experiments are presented in Table 1.

An experimental result of a cluster with two electronic HR neurons is shown in Fig. 15. The uncertainty of the period time can already be recognized for the uncoupled neurons at the start of the experiment (the first 7 sec the neurons are uncoupled). One would expect the neurons to have a fixed period time in this interval, but as can be seen there are small deviations in the period time, caused by the uncertainties present in the setup.

Table 1. Parameter values for the training algorithm and motor neuron parameters.

ϵ	0.2 [V]	ϵ_τ	$7e^{-6}$ [s]
α	0.3125	α_τ	2500

The adaptation is started at $t = 7$ sec with the synchronization part. During the synchronization procedure the coupling strength between the neurons is increased, which means that they influence each other, but are not synchronized yet. Due to the extra input term for the coupling signal it is possible that the neurons act in a different dynamical mode and therefore do not have a well-defined period time. This explains the large peak in the period time of neuron 1 during the synchronization procedure. At $t = 8$ sec, the synchronization procedure is completed and the cluster can be interpreted as a single neuron with one period time. The period time of the cluster is already relatively close to the reference period. The period time adaptation procedure is started and the period time of the cluster converges within the predefined bound of the reference period.

In this example, the period time of the cluster of two neurons can successfully converge to the period time of the reference neuron. However, when the period time of the reference neuron is not between the period times of the neurons in the cluster, the reference period cannot be reached and at least one extra neuron is needed to make a cluster with a period time equal to the reference period. This motivates the assumption that the cluster should have as many neurons as possible. But from experiments with the electronic neurons it turned out that if we have a cluster with more than seven neurons, the difference in period time between the cluster and the reference neurons starts to increase again, even outside the specified bound. This is a

result of the all-to-all coupling structure, with the increase of the number of neurons in a cluster, the total coupling gain increases. On the experimental setup, this means that besides the synchronization error measurement noise is also amplified. Thus for a large cluster, the presence of noise influences the synchronized state and the uncertainty in the period time increases. Therefore, the adaptation mechanism also reacts on fluctuations mainly caused by noise and it is unable to converge within the bounds. We found experimentally that the optimal number of neurons in a cluster is five, hence we use clusters with five neurons in the remainder of this work.

5.3. *Robot experiments*

In this subsection we will present results of experiments of the robot controlled by the neuronal network. We have replaced the sensor neurons S_{nj} of the neuronal controller discussed in Sec. 2.2 by the trained clusters of electronic Hindmarsh–Rose neurons, while keeping the motor neurons as described in Sec. 2.2. The parameters of the motor neurons used during the experiments are given in Table 2.

In Fig. 7 it was shown that the robot was unable to drive in a straight line if single, nonidentical neurons are used as sensor neurons. To show that the training procedure is able to solve this problem, the experiment is repeated, but now using trained clusters of neurons as sensor neurons. Because the clusters are trained by the procedure presented in the previous section, the sensor neurons now have an equal response for equal input voltages. Figure 16 shows the result of the repeated experiment with trained clusters as sensor neurons, from which we see that the robot is now able to drive in a straight line.

Driving in a straight line is of course not the goal of the neuronal controller. The experiment shown in Fig. 17 shows the mobile robot driving around in an unknown environment with different obstacles. The trajectory of the robot shows that it is able to drive around and avoid the obstacles. A movie of an experiment showing the mobile robot driving around, including the response of the three clusters of neurons in the sensor layer, can be found online: http://www.youtube.com/watch?v=sGg7JWWYsqY.

Table 2. Motor neurons parameters.

p	10	V_{rest}	0
τ_s	0.014	V_M	1.5
δ_t	0.02		

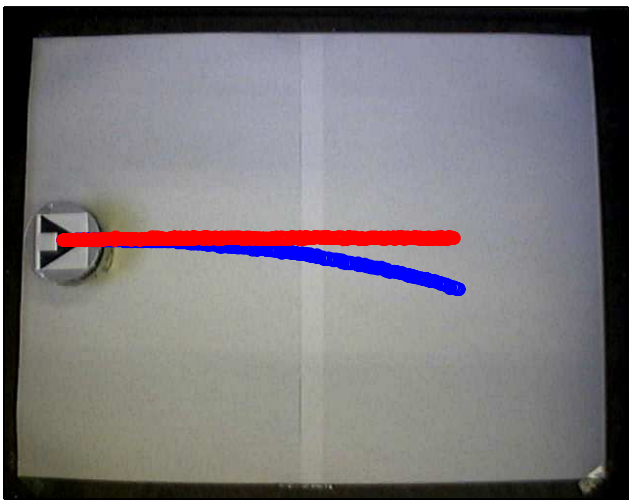

Fig. 16. Comparison between two experiments, the first using three nonidentical neurons (blue line) and the other with three trained clusters each containing five nonidentical neurons (red line).

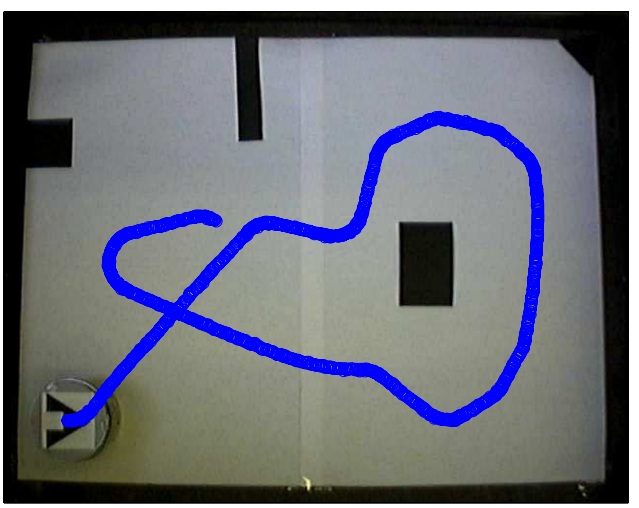

Fig. 17. Experimental result of the robot driving around, the initial robot position is shown and the black areas indicate obstacles.

6. Conclusions

Due to its ability to learn from previous experiences neuro/biological networks are a natural candidate to use for control of autonomous robots that have to operate in unknown real world environments. In addition, these networks have an improved robustness with respect to possible faults compared

to using a microcontroller. On the other hand, robustness of these types of networks is still an important issue in the area of synchronization. In this work we have presented a procedure to construct and train clusters of neurons. It allows to (re)define the network structure (interaction weights) of a cluster, in such a way that all neurons in a cluster are synchronized and the input–output properties of all clusters are practically identical. This practically identical input–output response is essential for proper functioning of the neuronal network controller. An additional advantage is that in case a few neurons of the brain (i.e. in a cluster) fail, a brief period of retraining (rehabilitation) ensures that the electronic brain is working properly again. The working principle of the proposed training procedure is demonstrated by controlling a real mobile robot using a neuronal controller using electronic neurons.

A cluster is a small network of synchronized neurons that can be interpreted as an average neuron. This means that the neurons in the cluster are synchronized by a sufficiently large coupling strength and the weighting of this coupling is adapted to change the period time of the cluster. Using this approach a cluster can obtain a period time in between the period times of the individual neurons in the cluster. Because our strategy requires all neurons in a cluster to respond as identical as possible to the same input, the neurons in the cluster need to be practically synchronized. Theoretical results on practical synchronization of asymmetrically coupled nonidentical neurons are presented.

The electronic neurons are electronic circuit realizations of Hindmarsh–Rose neural oscillators, which do not show identical behavior due to parametric mismatches. Experimental results demonstrate that the effects of mutual differences between the single cells are canceled out by the trained clusters of neurons. Moreover, we have shown that if we combine the neuronal controller and the training procedure we are able to make an electronic brain to control a mobile robot. The experiments confirm that training makes the neuronal network controller function properly.

References

Antonelo, E., Baerveldt, A.-J., Rögnvaldsson, T. & Figueiredo, M. [2006] "Modular neural network and classical reinforcement learning for autonomous robot navigation: Inhibiting undesirable behaviors," *Proc. Int. Joint Conf. Neural Networks*, pp. 498–505.

Antonelo, E., Schrauwen, B. & Stroobandt, D. [2008] "Event detection and localization for small mobile robots using reservoir computing," *Neural Netw.* **21**, 862–871.

Bloch, A. [2003] *Nonholonomic Mechanics and Control* (Springer-Verlag, NY).

Burton, T. A. [1985] *Stability and Periodic Solutions of Ordinary and Functional Differential Equations* (Academic Press, New York and London).

Chatterjee, R. & Matsuno, F. [2001] "Use of single side reflex for autonomous navigation of mobile robots in unknown environments," *Robot. Auton. Syst.* **35**, 77–96.

Floreano, D. & Mondada, F. [1998] "Evolutionary neurocontrollers for autonomous mobile robots," *Neural Netw.* **11**, 1461–1478.

Floreano, D., Epars, Y., Zuffery, J.-C. & Mattiussi, C. [2006] "Evolution of spiking neural circuits in autonomous mobile robots," *Int. J. Intell. Syst.* **21**, 1005–1024.

Guivant, J., Nebot, E. & Baiker, S. [2000] "Autonomous navigation and map building using laser range sensors in outdoor applications," *J. Robot. Syst.* **17**, 565–583.

Hindmarsh, J. L. & Rose, R. M. [1984] "A model of neuronal bursting using three coupled first order differential equations," *Proc. Roy. Soc. London Ser. B, Biol. Sci.* **221**, 87–102.

Johnston, S. P., Prasad, G., Maguire, L. & McGinnity, T. M. [2010] "An FPGA hardware/software co-design towards evolvable spiking neural networks for robotics application," *Int. J. Neural Syst.* **20**, 447–461.

Khalil, H. [2002] *Nonlinear Systems*, 3rd edition (Prentice-Hall).

Manoonpong, P., Pasemann, F. & Fischer, J. [2005] "Modular neural control for a reactive behavior of walking machines," *Proc. IEEE Int. Symp. Computational Intelligence in Robotics and Automation*, pp. 403–408.

Massa, G. L. O., Vinuesa, H. & Lanzarini, L. [2006] "Modular creation of neuronal networks for autonomous robot control," *Proc. IEEE Latin American Robotics Symp.*, pp. 66–73.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zuffery, J.-C., Floreano, D. & Martinoli, A. [2009] "The e-puck, a robot designed for education in engineering," *Proc. Conf. Autonomous Robot Systems and Competitions*, pp. 59–65.

Neefs, P. [2009] "Experimental synchronisation of Hindmarsh–Rose neurons in complex networks,"

Master's thesis (DCT 2009.107), Eindhoven University of Technology, the Netherlands.
Neefs, P., Steur, E. & Nijmeijer, H. [2010] "Network complexitiy and synchronous behavior — An experimental approach," *Int. J. Neural Syst.* **3**, 233–247.
Pearson, M. J., Pipe, A. G., Mitchinson, B., Gurney, K., Melhuish, C. & Nibouche, M. [2007] "Implementing spiking neural networks for real-time signal-processing and control applications: A model-validated FPGA approach," *IEEE Trans. Neural Netw.* **18**, 1472–1487.
Steur, E., Kodde, R. & Nijmeijer, H. [2008] "Synchronization of diffusively coupled electronic Hindmarsh–Rose oscillators," *Proc. Euromech Nonlinear Dynamics Conf.*
Steur, E., Tyukin, I. & Nijmeijer, H. [2009] "Semi-passivity and synchronization of diffusively coupled neuronal oscillators," *Physica D* **238**, 2119–2128.
Tang, Y., Qian, F., Gao, H. & Kurths, J. [2014] "Synchronization in complex networks and its application — A survey of recent advances and challenges," *Ann. Rev. Contr.* **38**, 184–198.
Wang, X., Hou, Z.-G., Zou, A., Tan, M. & Cheng, L. [2008] "A behavior controller based on spiking neural networks for mobile robots," *Neurocomputing* **71**, 655–666.
Wilamowski, B. [2003] "Neural network architectures and learning," *Proc. Int. Conf. Industrial Technology*, pp. TU1–TU12.
Zhang, M., Peng, S. & Meng, Q. [1997] "Neural network and fuzzy logic techniques based collision avoidance for a mobile robot," *Robotica* **15**, 627–632.

Appendix A
Practical Synchronization of Two Nonidentical HR Neurons

A.1. *Bounded solutions*

We start with finding an ultimate bound on the solutions of the coupled HR neurons. It is convenient to denote

$$x_i = \begin{pmatrix} y_i \\ z_{i,1} \\ z_{i,2} \end{pmatrix}.$$

As shown in [Steur *et al.*, 2009] the HR neuron with nominal parameters is strictly semi-passive with a quadratic storage function. That is, there exists a positive definite storage function $S : \mathbb{R}^3 \to [0, \infty)$,

$$s_1\|x_i\|^2 \leq S(x_i) \leq s_2\|x_i\|^2,$$

that is such that

$$\dot{S}(x_i) = y_i u_i - H(x_i),$$

where continuous function $H : \mathbb{R}^3 \to \mathbb{R}$ is positive for all $\|x_i\| \geq R$ for some $R > 0$. In particular, for the HR neuron the function H satisfies

$$H(x_i) > h\|x_i\|^2 - M$$

for some positive constants h and M such that $R^2 = \frac{M}{h}$. It is straightforward to show that each HR neuron (10) is also strictly semi-passive with a quadratic positive definite storage function S_i and a quadratic function H_i.

Let $s_1\|x_i\|^2 \leq S_i(x_i) \leq s_2\|x_i\|^2$ and $H_i(x_i) > h\|x_i\|^2 - M$ for $i = 1, 2$ and consider the function

$$V(x) = (1-\sigma)S_1(x_1) + \sigma S_2(x_2),$$
$$x = (x_1^\top \quad x_2^\top)^\top.$$

Denoting the norm

$$\|x\|_\sigma = \sqrt{(1-\sigma)\|x_1\|^2 + \sigma\|x_2\|^2},$$

we find

$$s_1\|x\|_\sigma^2 \leq V(x) \leq s_2\|x\|_\sigma^2.$$

In addition we have

$$\begin{aligned}\dot{V}(x) &= (1-\sigma)y_1u_1 + \sigma y_2u_2 - (1-\sigma)H_1(x_1) \\ &\quad - \sigma H_2(x_2) \\ &\leq -h\|x\|_\sigma^2 + M\end{aligned}$$

because

$$\begin{aligned}&(1-\sigma)y_1u_1 + \sigma y_2u_2 \\ &\quad = -\gamma\sigma(1-\sigma)(y_1^2 - 2y_1y_2 + y_2^2) \leq 0.\end{aligned}$$

Then by Theorem 4.1.16 of [Burton, 1985] the solutions of the two coupled nonidentical HR neurons are uniformly bounded and uniformly ultimately bounded with bound B, which is defined by the identity $s_1B^2 = s_2(R+1)^2$. Of course, this uniform (ultimate) boundedness of solutions is with respect to the norm $\|\cdot\|_\sigma$.

A.2. *Practical synchronization*

The results on uniform (ultimate) boundedness of solutions allow us to consider the practical synchronization problem as a practical stabilization problem of the origin of the error-system $x_1 - x_2$. Without loss of generality we assume $\sigma \in (0, \frac{1}{2}]$ such

that $1-\sigma \geq \frac{1}{2} \geq \sigma$. Then by the uniform ultimate boundedness we may assume

$$(1-\sigma)\|x_1\|^2 \leq \|x\|_\sigma \leq B^2 \Rightarrow \|x_1\| \leq \sqrt{2}B.$$

Denote

$$\tilde{y} = y_1 - y_2, \quad \tilde{z}_1 = z_{1,1} - z_{2,1}, \quad \tilde{z}_2 = z_{2,1} - z_{2,2}$$

and consider the following identities:

$$y_1^2 - y_2^2 = \tilde{y}(y_1 + y_2),$$
$$y_1^3 - y_2^3 = \frac{1}{4}\tilde{y}(\tilde{y}^2 + 3(y_1+y_2)^2).$$

We obtain

$$\begin{aligned}
\dot{\tilde{y}} &= -\frac{\alpha_{2,1}}{4}\tilde{y}(\tilde{y}^2 + 3(y_1+y_2)^2) + \alpha_{2,2}\tilde{y} \\
&\quad + \alpha_{2,3}\tilde{z}_1 - \alpha_{2,4}\tilde{z}_2 - \gamma^*\tilde{y} + \Delta_1, \\
\dot{\tilde{z}}_1 &= -\alpha_{2,8}\tilde{y}(y_1+y_2) - \alpha_{2,9}\tilde{y} - \alpha_{2,10}\tilde{z}_1 + \Delta_2, \\
\dot{\tilde{z}}_2 &= \alpha_{2,11}\tilde{y} - \alpha_{2,13}\tilde{z}_2 + \Delta_3,
\end{aligned}$$

where, for $\delta_j := \alpha_{1,j} - \alpha_{2,j}$, $\gamma^* = \gamma(\alpha_{2,7} + (1-\sigma)\delta_7)$ and

$$\begin{aligned}
\Delta_1 &= -\delta_1 y_1^3 + \delta_2 y_1 + \delta_3 z_{1,1} - \delta_4 z_{1,2} - \delta_5 + \delta_6 I, \\
\Delta_2 &= -\delta_8 y_1^2 - \delta_9 y_1 - \delta_{10} z_{1,1}, \\
\Delta_3 &= \delta_{11} y_1 + \delta_{12} - \delta_{13} z_{1,2}.
\end{aligned}$$

We observe that $\Delta_1, \Delta_2, \Delta_3$ are uniformly bounded as $\|x_1\| \leq \sqrt{2}B$.

Consider the function

$$W(\tilde{y}, \tilde{z}_1, \tilde{z}_2) = \frac{1}{2}\tilde{y}^2 + \frac{c_1}{2}\tilde{z}_1^2 + \frac{c_2}{2}\tilde{z}_2^2$$

with $\alpha_{2,11}c_2 = \alpha_{2,4}$ and positive constant c_1 to be determined. The derivative of W (with respect to t) is given below,

$$\dot{W}(\tilde{y}_1, \tilde{z}_1, \tilde{z}_2) = \Delta_1\tilde{y}_1 + c_1\Delta_2\tilde{z}_1 + c_2\Delta_3\tilde{z}_3 - \frac{\alpha_{2,1}}{4}\tilde{y}_1^4 - c_2\alpha_{2,13}\tilde{z}_2^2 - \begin{pmatrix}\tilde{y}\\ \tilde{z}_1\end{pmatrix}^\top \underbrace{\begin{pmatrix} \gamma^* + \frac{3}{4}\alpha_{2,1}\eta^2 - \alpha_{2,2} & \frac{1}{2}(c_1\alpha_{2,8}\eta + c_1\alpha_{2,9} - \alpha_{2,3}) \\ \frac{1}{2}(c_1\alpha_{2,8}\eta + c_1\alpha_{2,9} - \alpha_{2,3}) & c_1\alpha_{2,10} \end{pmatrix}}_{=:Q(\gamma^*,\eta)} \begin{pmatrix}\tilde{y}\\ \tilde{z}_1\end{pmatrix}$$

with $\eta := y_1 + y_2$. Let us show that there is a $\overline{\gamma}$ such that for $\gamma^* > \overline{\gamma}$ the matrix $Q(\gamma^*, \eta)$ is positive definite for all η. It is straightforward to see that $\overline{\gamma} > \alpha_{2,2}$. We have

$$\begin{aligned}
&4\det(Q(\gamma^*,\eta)) \\
&\quad = c_1(3\alpha_{2,1}\alpha_{2,10} - c_1\alpha_{2,8})\eta^2 \\
&\qquad - 2c_1\alpha_{2,8}(c_1\alpha_{2,9} - \alpha_{2,3})\eta \\
&\qquad + 4c_1(\gamma^* - \alpha_{2,2}) - (c_1\alpha_{2,9} - \alpha_{2,3})^2 \\
&\quad =: p_2\eta^2 + p_1\eta + p_0(\gamma^*).
\end{aligned}$$

Choose c_1 small enough to ensure that $p_2 > 0$. Note that $4\det(Q(\gamma^*,\eta))$ attains its minimum at $\eta = \frac{-p_1}{2p_2}$. Let $\overline{\gamma}$ be such that

$$\begin{aligned}
p_0(\overline{\gamma}) &= 0, \quad \text{if } p_1 \geq 0, \\
4p_2p_0(\overline{\gamma}) &= p_1^2, \quad \text{if } p_1 < 0.
\end{aligned}$$

Then for any $\gamma^* > \overline{\gamma}$ we have $\det(Q(\gamma^*,\eta)) > 0$, hence there exist positive constants κ_1, κ_2 such that

$$\dot{W} \leq -\kappa_1 W + \kappa_2\sqrt{W}.$$

Again by Theorem 4.1.16 of [Burton, 1985] (see also, for instance, Theorem 4.18 of [Khalil, 2002]), we have that the synchronization errors are uniformly bounded and uniformly ultimately bounded. It is important to note that constant κ_2 is proportional to the uncertainties $\Delta_1, \Delta_2, \Delta_3$, such that a decrease in the uncertainties results in a decrease in the bound on the synchronization errors. In fact, if $\Delta_1 = \Delta_2 = \Delta_3 = 0$ the synchronization errors converge to zero exponentially.

Finally, consider

$$\begin{aligned}
\dot{\tilde{y}} &= -\frac{\alpha_{2,1}}{4}\tilde{y}(\tilde{y}^2 + 3(y_1+y_2)^2) + \alpha_{2,2}\tilde{y} \\
&\quad + \alpha_{2,3}\tilde{z}_1 - \alpha_{2,4}\tilde{z}_2 - \gamma^*\tilde{y} + \Delta_1
\end{aligned}$$

and recall that $y_1, y_2, \tilde{z}_1, \tilde{z}_2, \Delta_1$ are all bounded. Consider the function

$$W_1(\tilde{y}) = \tilde{y}^2.$$

Clearly there exist a positive constant $\kappa_4 = \|\alpha_{2,3}\tilde{z}_1 - \alpha_{2,4}\tilde{z}_2 + \Delta_1\|$ and a strictly increasing function $\kappa_3(\gamma^*) = \gamma^* - \alpha_{2,2}$, such that

$$\dot{W}_1 \leq -2\kappa_3(\gamma^*)W_1 + 2\kappa_4\sqrt{W_1}.$$

Using the arguments provided above we conclude that for $\gamma^* > \overline{\gamma}$ the synchronization error $\tilde{y}$ is uniformly bounded and uniformly ultimately bounded with a bound $\epsilon = \epsilon(\gamma^*)$, ϵ is strictly decreasing with $\overline{\gamma}$ and $\lim_{\gamma^* \to \infty} \epsilon(\gamma^*) = 0$.