Computing and Informatics, Vol. 29, 2010, 823–847

THE APPLICATION OF SPIKING NEURAL NETWORKS IN AUTONOMOUS ROBOT CONTROL

Peter Trhan

Department of Computer Science
Faculty of Natural Sciences, University of Matej Bel
Tajovského 40, 974 01 Banská Bystrica, Slovakia
e-mail: petertrhan@pobox.sk

Manuscript received 4 September 2008; revised 18 November 2008

Communicated by Vladimír Kvasnička

Abstract. Artificial neural networks have a wide range of applications nowadays in which they are used for intelligent information processing. This paper deals with an application of spiking neural networks in autonomous mobile robot control. The topology of the implemented spiking neural networks was developed through a modified genetic algorithm and through the process of autonomous interaction with the scene environment. Since the genetic algorithm did not use a crossover operator we adapted the mutation operator adding a constraint that prevented creation of a new generation of population with weak individuals in comparison with the previous generation of population. The paper proposes a parallel combination of both left and right local spiking neural network as well as a practical implementation of this proposition in the form of an intelligent navigation system in an autonomous mobile robot. This design enhances the implemented navigation system with a new cognitive property of intelligent information processing using a spiking neural network. Having been adapted to the scene environment, the navigation system was able to make right decisions, change its direction and refrain from collision with the scene walls.

Keywords: Spiking neural network, genetic algorithm, population, navigation system, mobile robot, trajectory control

1 INTRODUCTION

In our experiments the application of spiking neural networks for autonomous robot control has been realized on a software basis. The proposed architecture, using a parallel combination of two local spiking neural networks, is the core of the navigation system of an autonomous robot. This model of the navigation system takes its inspiration from biology, integrating evolutionary methods, neural networks and hardware equipment. Moreover, the implemented navigation system is capable of self-development through the interaction with the scene environment without human intervention. With these characteristics, the proposed system complies with the principles and methodology of evolutionary robotics [17].

The proposed software implementation of the navigation system lends itself to implementation in reconfigurable hardware devices such as EPGA (Field Programmable Gate Array) or PLD (Programmable Logic Devices). This kind of implementation can be regarded as evolutionary hardware [19].

1.1 Examples of Applications of Neural Networks in Intelligent Control Systems

Some interesting examples of the application of recurrent neural networks in autonomous robot trajectory control were realized by Ziemke [27]. He compared and evaluated four architectures of recurrent neural network, which he implemented in the control system of a miniature robot named Khepera. The robot moved within a rectangle-shaped scene measuring 1 by 0.6 meter. The recurrent neural network received input information via 8 optical sensors with a range of 55 milimetres while two output neurons controlled the movement of a left and a right engine. During the adaptation of recurrent neural network to the scene environment, four starting positions were randomly selected. The task of the recurrent neural network was to control the movement of the robot without causing it to collide with the walls of the scene and to park the robot in a designated zone. One of the sensors evaluated the intensity of the reflected light within the scene and monitored whether the robot reached the designated zone. The best results of the tested architectures of recurrent neural networks were achieved by an extended sequential cascaded network. In another series of experiments the author changed the environment of the scene and made the task of the robot more challenging. He added to the scene several circular signs, some within and others outside the designated zone. In this case, the task of the robot was to avoid the signs outside the zone and gradually pass over the signs within the designated zone. Similarly to the first series of experiments, the best results were achieved by the extended sequential cascaded network.

Researchers at the Laboratory of Intelligent Systems in Switzerland [14] have been developing intelligent navigation systems for autonomous mobile robots for a long time. The core of their navigation systems consists of neural networks, which adapt to the scene environment through a genetic algorithm using autonomous

interaction of the system with its environment [6]. The Laboratory of Intelligent Systems has been developing and experimentally testing the reliability of control system prototypes which feature cognitive properties. Their systems are extensively inspired by nature and draw especially on natural principles used by flying insects. Currently, the Laboratory of Intelligent Systems is involved in the development of intelligent flying robots, which boast low energy consumption and autonomous as well as intelligent behavior. Previously, they implemented their navigation systems into the miniature wheel robots named Kephera and Alice, a small airship named Blimp2b and the ultra light airplanes C4 and F2 [28]. The most useful, from point of view of our work, is the implementation of navigation systems in the wheeled robots Khepera and Alice. The intelligent element of the navigation systems in these two robots is a spiking neural network. The evolutionary development of the employed spiking neural network within the scene environment took place without human intervention. Experiments confirmed that this type of neural network is capable of controlling the movement of a mobile robot without letting it collide with the obstacles in the scene [4, 5].

Cellular neural networks are based on a different type of architecture and the spatial location of its building units as well as on a different principle of signal transmission among these units. The architecture and mathematical model of cellular neural networks are designed for parallel processing of information. They perform better in the cases where traditional methods are unable of delivering required speed for parallel processing of information, e.g., when processing video signal in real time. The cellular neural networks have been successfully applied to solving tasks in signal processing, image processing (filtering of image information, edge and object detection, modeling optical flow, sign and object recognition, etc.); they have been used to analyze 3D surfaces and solved differential equations [26].

The Computer and Automation Research Institute at the Hungarian Academy of Science has been actively involved in the research in the field of cellular neural networks. Among other things, researchers there have analyzed spatial and time qualities of movement, modeled optical flow and explored object recognition. In solving these tasks, they have successfully applied cellular neural networks. The knowledge acquired is implemented in the navigation systems of autonomous mobile robots and airplanes. Their system is capable of detecting objects by means of modeling optical flow from a sequence of images with low resolution and, at the same time, it is able to recognize objects from images with high resolution [1, 20].

Having considered the above described three types of neural networks we decided to apply a spiking neural network and implement it in a navigation system of an autonomous robot. Our choice was influenced by the research results achieved so far as well as by the availability of necessary hardware, which we needed for implementation and for conducting experiments.

1.2 Applied Hardware and Software

Currently, a financially affordable generation of robots is available on the market. The use and programming of these robots do not require advanced knowledge in robotics. What is more, these devices have light weight, low energy consumption, easy maintenance, they use wireless communication and are equipped with basic sensors.

Our proposed navigation system was designed to be used with the Lego Mindstorms NXT robotics kit [15] (henceforth referred to as NXT). On the basis of our experience we can say that the choice of this financially affordable robot for our experiments was a correct one. The communication between our desktop computer and the NXT robot took place via Bluetooth technology, which proved sufficiently fast for this implementation of neural networks. The applications used with the NXT were programmed in the Eclipse [3] freeware development platform using the Java programming language and Java Runtime Environment (version 1.6) running on the Windows XP 32 bit operational system.

The iCommand freeware library offers Java classes for controlling the NXT via Bluetooth. This library works directly with the LEGO NXT firmware and we did not have to substitute the original NXT firmware for the leJOS NXJ. The latter is an interpreter of the platform-free Java bytecode for the NXT. Our implementations of neural networks used iCommand, version 0.7 [16]. This technology enabled us to create applications for the NXT in Java on a desktop computer while maintaining robot mobility, i.e. we could use computing and memory capacity of the desktop computer and made full use of robot features at the same time.

Figure 1 depicts the communication between the NXT and the desktop computer. It represents the implementation of a parallel combination of the left and right local spiking neural network, which receive coded information from ultrasonic sensors. The communication takes place in real time via Bluetooth technology. The transmission of information itself takes place during each sensory-control cycle of networks activity as the system receives current information from the sensors and directs the movement of the robot accordingly. The wireless information transfer is shown as a dashed line. The applied type of communication does not limit mobility of the robot. When activated, the proposed navigation system initiates communication between the desktop computer and the robot. Before becoming inactive, communication is terminated. During operation, the navigation system is not hampered by the communication, it is stable and, in case we need to access current data from its sensor or to change robot direction, we can use the appropriate method from the iCommand library.

All the implementations were carried out using one or two ultrasonic sensors and one optical sensor. When using only one ultrasonic sensor, we placed it on the front of the robot, in a forward direction of its movement. When two sensors were used they were positioned on the left and right arms of the robot. The robot's small arms were located on the front side at an angle of $\pm 45^\circ$ to its longitudinal axis. The position of arms was adjusted according to need. The optical sensor was

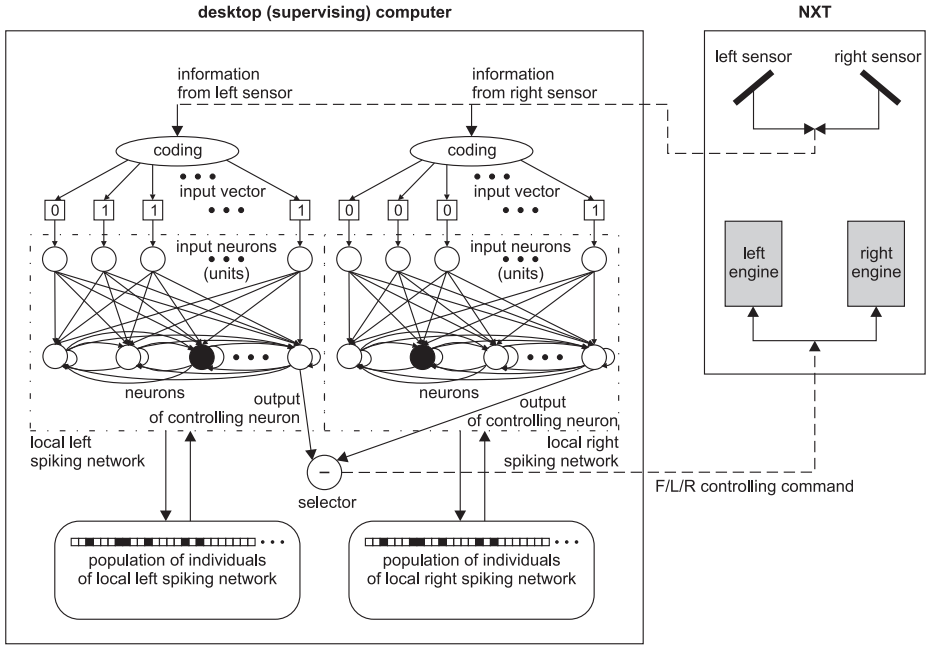

Fig. 1. Communication between the desktop computer and the mobile NXT robot via the iCommand library and Bluetooth technology

placed in the middle of the front side facing downwards, evaluating the intensity of the light reflected from the scene floor. By means of the optical sensor we were able to monitor the situation when the robot was leaving the designated zone (by passing over the black line). Leaving the designated zone would in practice result in an undesired contact with the wall of the scene. In our experiments, this served as a simulated collision of the robot.

In our last implementation we added a miniature wireless camera to the system. Ajoka Aj-007S miniature camera was attached to the front side of the mobile robot (Figure 2, part 2). It ran on 9 volt battery, which was placed behind the NXT intelligent cube. The analogue signal from the camera was transmitted on a radio wave into an audio and video receiver. The radio receiver is purchasable in a package with the camera. In addition, we used a Trust Surveillance Interface 801 analogue video and audio digitizer for this implementation. It served as an interface between an analogue source of video signal (i.e. our radio receiver) and the desktop computer. The camera, receiver and digitizer provided a wireless transmission of image information from the analogue camera directly into the desktop computer in a digital form.

In order to acquire and process image for the input vector of the neural networks in the Java language we used the freeware Java Media Framework classes,

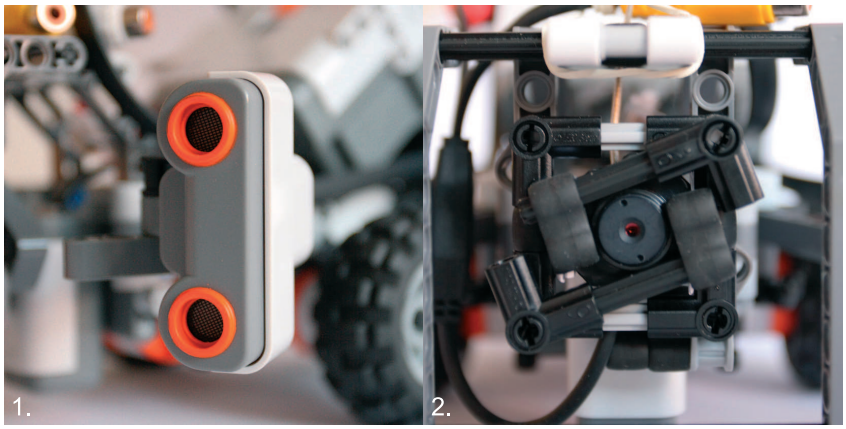

Fig. 2. Part 1) the ultrasonic sensors; Part 2) the wireless camera

version 2.1.1e [22] and the Vision class, which is also a part of the iCommand library.

2 THE MODEL AND DYNAMICS OF A SPIKING NEURON

The models of spiking networks differ from one another in the definition of their membrane potential of a neuron. The membrane potential represents a state of the spiking neuron and characterizes neuronal dynamics. The incoming impulses from presynaptic neurons increase (the impulses passing through excitatory synapses) or decrease (the impulses coming through inhibitory synapses) the membrane potential of the postsynaptic neuron [8].

2.1 The Mathematical Model of the Implemented Spiking Neuron

The implemented model of the neuron which we used in our applications of spiking neural networks is based on the SRM_0 model [8]. This model, in contrast to the original SRM_0 one, uses only linear functions, which denote the current state of the neuron as well as the individual response kernels of the presynaptic and postsynaptic impulses. The weight of a synapsis is coded with the rate of one bit (value 1 corresponds to the existing synaptic connection between neurons, value 0 represents the absence of synaptic connection). Thus, the created model is easy-to-calculate and lends itself to the application in control systems with low-power consumption [4, 5].

The membrane potential u_i of the neuron i is updated at discrete time intervals. The kernel ε_{ij}, which denotes the postsynaptic potential, represents the current increment of the membrane potential of the neuron at a given time interval. The increment of neuron i of the presynaptic neuron j is represented by the following

function (1):

$$\varepsilon_{ij}(t) = o_j(t-1)w_{ij}s_j \tag{1}$$

where $o_j(t-1) = 1$, if the presynaptic neuron j has generated an impulse at the preceding time interval $t-1$, otherwise (if the neuron j has not generated an impulse) $o_j(t-1) = 0$. The s_j variable (s_j is the sign of neuron j) determines whether the postsynaptic neuron potential j is excitatory ($s_j = 1$) or inhibitory ($s_j = -1$). The w_{ij} variable defines the weight of the synapse between the j presynaptic neuron and i postsynaptic neuron (if there is connection from j neuron into i neuron, therefore $w_{ij} = 1$, otherwise $w_{ij} = 0$).

The synaptic connections and signal direction in Figure 4 are shown as oriented edges. The signal corresponding to output (o_i or i_i) from neuron i is transmitted through an oriented edge starting from neuron i.

The incremental value of neuron i from input neuron j which is a neuron response to the external input is expressed by the function (2):

$$\kappa_{ij}(t) = i_j(t-1)w_{ij} \tag{2}$$

where $i_j(t-1)$ is an output of input neuron j at the preceding time interval $t-1$. During the activity of the implemented neural network the external input does not update at each time interval but at regular intervals, every time after a certain number of time intervals. At the time when updating of the input neurons does not occur $i_j(t) = 0$.

The functions (3) define kernel η_i for the phase of the action potential (of impulse generation) and the hyperpolarisation phase of neuron i. Kernel $\eta_i(t)$ carries out the following operations:

$$\begin{aligned} o_i(t) = 1 \ \wedge \ u_i(t) = 0 \qquad & \text{if} \ \ u_i(t) > \nu + r(t) \\ o_i(t) = 0 \qquad & \text{if} \ \ u_i(t) \leq \nu + r(t) \end{aligned} \tag{3}$$

where $r(t)$ represents a randomly generated relatively small value from the interval with centre in 0. As a result of adding random value $r(t)$ to the threshold ν of neuron i, the level of neuron threshold changes dynamically, the random value $r(t)$ brings noise to the neuron and improves the properties of the neural network [4, 5]. If the membrane potential of neuron i at time t exceeds its threshold value ν which changed by adding up value $r(t)$, the neuron generates an impulse (the output of neuron i is set to value 1, i.e. $o_i(t) = 1$) and the value of its potential drops to low negative value η_0. In our case, we did not use low negative value, but we set neuron potential to 0, i.e. $u_i(t) = 0$. The hyperpolarisation phase corresponds to neuronal refractoriness after an impulse has been generated in the following time interval $t+1$.

The membrane potential fluctuation causes a decrease of the membrane potential in the absence of presynaptic impulses. This property of neuron is defined by subtracting of the constant value from the membrane potential of the neuron at each time interval.

Putting all the above kernels ε_{ij}, κ_{ij}, η_i together (Equations (1) to (3)) produces the mathematical model of the implemented impulse neuron (4):

$$u_i(t) = \sum_j \varepsilon_{ij}(t) + \sum_k \kappa_{ik}(t) \quad \text{and the realization of kernel } \eta_i(t) \tag{4}$$

while the summation of kernels ε_{ij} goes through all the presynaptic neurons and the summation of kernels κ_{ik} passes through all input neurons.

2.2 Algorithmization of the Implemented Spiking Neuron Model

The behavior of the spiking neuron (employed in our implementations and described in the preceding subchapter 2.1) can be summarized by the following algorithm [5, 25]:

1. If at the preceding time interval $t-1$ neuron i generated an impulse, its membrane potential u_i will be blocked (hyperpolarization phase) and in the following time interval $t+1$ it will pass into a resting state and will be able to get updated.
2. Updating of the potential of neuron (if the membrane potential is not blocked) by the incoming impulses from presynaptic and input neurons (5):

$$u_i(t) = \begin{cases} u_i(t-1) + \sum_j \varepsilon_{ij}(t) & \text{if } u_i(t-1) + \sum_j \varepsilon_{ij}(t) + \sum_k \kappa_{ik}(t) \geq u_r \\ \quad + \sum_k \kappa_{ik}(t) & \\ u_r & \text{if } u_i(t-1) + \sum_j \varepsilon_{ij}(t) + \sum_k \kappa_{ik}(t) < u_r \end{cases} \tag{5}$$

   where $u_i(t)$ represents the new updated membrane potential of neuron i in time t and $u_i(t-1)$ stands for its preceding potential at time $t-1$. Kernels $\varepsilon_{ij}(t)$ and $\kappa_{ik}(t)$ are calculated through the equations (1) and (2). The resting value u_r is, in our case, set to 0.
3. The phase of impulse generation set by the output $o_i(t)$ of neuron i at time t according to (3). If neuron i was inactive (has not generated an impulse), its potential will be updated at the following time interval $t+1$.
4. Realization of the fluctuation of the potential of neuron i (6):

$$u_i(t) = \begin{cases} u_i(t) - k & \text{if } \; u_i(t) - k \geq u_r \\ u_r & \text{if } \; u_i(t) - k < u_r \end{cases} \tag{6}$$

   where k is a small value (experimentally chosen) which does not change during the network activity.

3 THE MODEL OF A SPIKING NEURAL NETWORK

On the basis of the published results by the Laboratory of Intelligent Systems [4, 5, 28] we implemented their model of a spiking neural network in our conditions and we conducted a series of experiments.

The spiking neural network learnt to control the movement of a mobile robot on the basis of autonomous reaction with the environment, i.e. the spiking network adapted to the created scene environment through evolutionary development of individuals of the population, which coded synaptic connections as well as the neuron signs. The implemented evolution adapted to the neural network in a relatively short time and the network was capable of navigating the mobile robot without collisions with scene walls. The robot moved in straight lines until it approached a wall of the scene (or other scene obstacle); then it turned round and moved on.

In our conditions, however, the implemented spiking network in the navigation system of the robot was not able to turn in the correct direction. For example, if the robot approached the wall from the left side in the direction of its movement, that is to say if the angle between the longitudinal axis of the robot and the direction vector of the wall plane was smaller on the left side of the robot (in Figure 3, $\alpha < \beta$) then it would be desirable to avoid collision with the wall by taking a right turn. For this reason we proposed a new architecture of the spiking neural network and realized a new series of experiments.

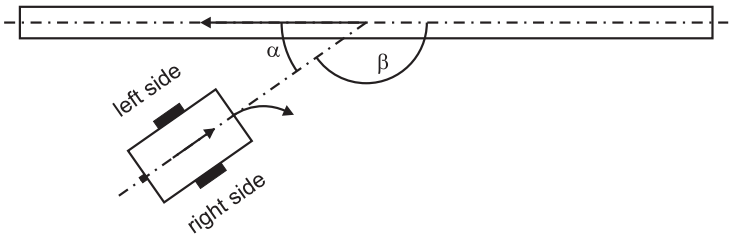

Fig. 3. Turning the robot in the correct direction in front of the scene wall

3.1 The Architecture of the Spiking Neural Network

Our proposed architecture of a parallel combination of two local spiking neural networks was influenced by the following assumptions and conditions:

- Inspiration from nature – the choice of the type of an artificial neural network. Flying insects perceive the image of the environment through a pair of compound eyes. The communication in insects between the eye and the brain in the form of impulses is often ten times higher as it is in humans. Owing to this fast communication insects are able to detect movement quickly. Movement of objects in the environment represents impending danger [18]. The model of the implemented navigation system to control the movement of a robot (the core of this system is a spiking neural network) is similar to the biological system of vision and communication as well as the evaluation of input information from the environment.

- Inspiration from nature – the solution of the problem of learning to turn into a desired direction before an obstacle through individual processing of information from the left and right side of the environment in the direction of movement. In most animals (in contrast to humans) the fields of vision of individual eyes do not overlap and each eye "sees separately" [2], i.e. the eye independently evaluates the information from its field of vision – from its point of view.
- The results of our experiments with the implemented architecture of a spiking neural network according to the Laboratory of Intelligent Systems [14]. The system detected impending collision with an obstacle, but lacked the ability of turning into a desirable direction before an obstacle.
- Adaptive and hierarchical mixtures of local feedforward neural networks which delegate classification of objects among several neural networks. A gating neural network produces coefficients of proportionality for corresponding input vectors. On the basis of the highest value of the coefficient of proportionality the selector realizes the selection of an output vector of the corresponding local neural network [9, 10].
- The deterministic principle of neural networks which asserts that every output impulse has to have its cause in the preceding input impulses [13]. In coding input vectors we worked on the above deterministic principle, according to which closer proximity from an obstacle causes a higher activity of input neurons (input units) and, in turn, this higher activity of input units causes a higher activity of output neurons (what is important to us is the activity of controlling neurons; the controlling neurons are those which are used in coding output information from the neural network).
- The simplifications of the algorithmization of the system in relation to the hardware – the availability of sensors and their ability to monitor the environment, the functionality of the applied communication libraries, the features allowing to control the trajectory of robot movement through the information from the network output.
- Taking into account the spatial limits of the physical surroundings during the realized experiments – the size and shape of the created scene in relation to the dimensions of the mobile robot.

The proposed architecture consisting of a combination of two local spiking networks and the flow of information from the sensors including the controlling output from the selector are shown in Figure 4. This architecture does not use a gate neural network. The selector makes a selection according to the prevailing activity of the controlling neuron of the corresponding spiking network (i.e. it selects a more active local network). Another important information from the selector is the position of the more active local network (either left or right). The left local spiking network is defined as the one which receives the input vectors coded from the current information on the "left side" of the scene environment in the direction of robot movement.

The right local spiking network receives the information on the "right side" of the scene environment.

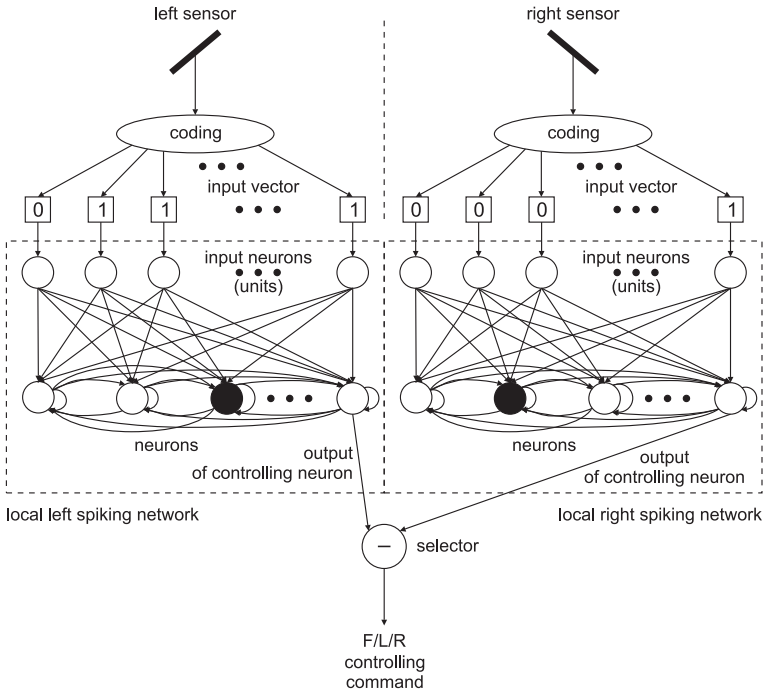

Fig. 4. The architecture of a paralel combination of two local spiking neural networks

Different input vectors for the left and right local networks (both input vectors activate the input neurons of the spiking networks at the same time), the absence of a gate neural network, the type of neural networks as well as their target use are the main differences in comparison to an adaptive or hierarchical mixture local feedforward neural networks [9, 10].

In our experiments we used two ultrasonic distance detection sensors located on the left and right front side of the robot at a $\pm 45^{\circ}$ angle with the longitudinal axis of its movement direction. Similarly to most animals, the left and right vision fields do not overlap.

The applied coding of input vectors generates a higher activity of input units in relation to a decreasing distance from an obstacle. According to the deterministic principle of neural networks, a higher activity of input units causes an increased activity of neurons (our local networks have only two layers – the input and the output one). These working assumptions and the defined spatial topology of the proposed architecture comprising a parallel combination of two local spiking networks enables us to determine the location of a nearby obstacle by comparing the activities of the left and right local networks.

In our approach, both left and right local spiking networks have the same parameters; the only prerequisite being one controlling neuron in the output layer of both networks. The output information from the controlling neuron is coded according to the number of generated impulses between two subsequent sensory-controlled cycles (a sensory-controlled cycle is the cycle activity of the neural network when sensors and engines get updated). This information is a response of the neural network to one input vector. The higher activity of the controlling neuron of the left local spiking network in comparison to the activity of the controlling neuron of the right spiking network signals an impending collision on the left side, i.e. the robot approaches an obstacle or the wall in the movement direction from the left and it is desirable for the robot to avoid collision by turning right. In this case selector output is the controlling command to turn right ("R" in Figure 4). Analogically, a prevailing activity of the right local network signals an impending collision on the right side and selector output is a controlling command to turn left ("L" in Figure 4).

If there are no obstacles or a wall in robot direction of movement the elements of input vectors of both local networks have zero value, neurons are inactive and do not generate impulses. In this case, selector output is a controlling command to carry on in the forward direction ("F" in Figure 4).

3.2 The Evolutionary Adaptation of a Parallel Combination of Local Spiking Networks

The adaptation of spiking neural networks by means of evolutionary optimization algorithms is an alternative to the traditional gradient descent methods. The models of spiking neurons are described by complex mathematical equations and the application of gradient descent optimization methods makes the equations more complex still [8, 11, 21]. Due to the use of these equations, the adaptation of spiking neural networks is more calculation demanding and its implementation, in most cases, requires simplifications to fit the applied hardware.

In our implementations we use the evolutionary adaptation of spiking networks. In particular, we chose the most popular, genetic algorithm. The genetic algorithm that the evolutionary model of our navigation system uses searches for the fittest individual on the basis of the information from the scene environment and the defined objective function (fitness function[1]).

Graphic representation of the navigation system is shown in Figure 5. The navigation system acquires information on the scene environment through sensors or through image information from the camera. Once determined, the topology of the local spiking networks, coded by the fittest individuals from two populations (each local network has its own population of individuals) creates a navigation system

[1] The fitness function is the function you want to optimize. For standard optimization algorithms, this is known as the objective function [7].

which is capable of controlling the mobile robot without collisions with the obstacles of the scene.

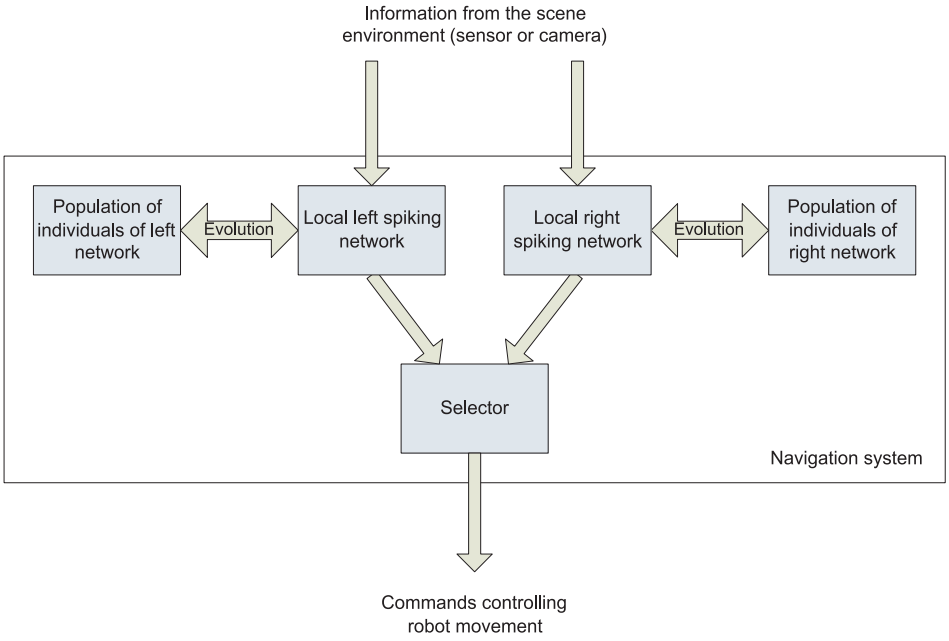

Fig. 5. Graphic representation of the implemented navigation system

One individual from the population codes the topology of a spiking neural network (active synaptic connections and neuron signs) in the form of a binary genetic string. The success rate of an individual is determined by the function value of the defined objective function [12]. In order to simplify the implementation, we defined the objective function in such a way that the optimization genetic algorithm maximized the function values of developing individuals, i.e. the fittest individual had the maximum function value of the objective function.

The objective function was defined on the basis of the following requirements that our navigation system had to meet:

- To simplify implementation the function values of the objective function have to be normalized, i.e. its range of definition should be defined by the interval $\langle 0, 1 \rangle$.
- It is desirable that the robot moves in straight lines, i.e. that it keeps a long enough distance from the obstacles (walls of the scene).
- Given the proposed coding of the input vectors of the spiking networks, a longer distance from an obstacle does not have a bearing on the activity of the network neurons and the response of the controlling neurons to such a stimulus equals zero amount of generated impulses. In response to such a stimulus the selector

evaluates the controlling command for straight movement. Another requirement for our navigation system is minimization of the number of commands for robot turns.

- A failure of the system in the form of a simulated collision, i.e. the robot passing over the black line, is undesirable. The black line marks the limits of the scene passing along its walls.
- The navigation system requires two evolutionary adaptations of local spiking neural networks.

Each individual of the population is evaluated within 15 seconds on the basis of the real information from the scene environment, reactions of the navigation system (the core of the system consists of two local spiking networks), robot trajectories and monitoring of simulated collisions. All of these are evaluated during every sensory-controlled cycle of local networks activities and the resulting value of the objective function is calculated as an average of these partial values of the objective function.

Let $1, 2, \ldots, t, t+1, t+2, \ldots, n$ be the time sequence of the sensory-controlled cycles realized within 15 seconds of the evaluation of two individuals (the robot moves within the boundaries of the scene and is controlled by the navigation system with the topology of spiking networks on the basis of currently evaluated individuals). The value of the objective function calculated at the sensory-controlled cycle t is expressed as the partial fitness value of an individual from the left local network $fitL_{ind}(t)$. Analogically, the value of the objective function calculated at the same sensory-controlled cycle t is expressed as the partial fitness value of an individual from the right local network $fitR_{ind}(t)$. Concurrently, we make the calculations of the partial fitness values $fitL_{ind}(t)$ a $fitR_{ind}(t)$ for the individuals of both local networks. The resulting fitness value of a population individual of the left local network is thus calculated through Equation (7):

$$fitL_{ind} = \frac{\sum_1^n fitL_{ind}(t)}{n} \tag{7}$$

In the same way, we synchronically calculate the evaluation of an individual of the right local network. The calculation of all parts of the partial fitness values $fitL_{ind}(t)$, $fitR_{ind}(t)$ and their gradual summation are realized at every sensory-controlled cycle. The resulting fitness values of the individuals $fitL_{ind}$ and $fitR_{ind}$ are calculated according to (7), after evaluation of individuals has been completed.

The calculation of the partial fitness value $fitL_{ind}(t)$ at the sensory-controlled cycle t was then defined, in relation to the requirements laid down for our navigation system, through Equation (8); the same procedure applied for an individual of the right local network:

$$fitL_{ind}(t) = fit_{distL}(t) fit_{directL}(t)(1 - fit_{crashL}(t)) \tag{8}$$

Equations (7) and (8) define the objective function of our genetic algorithm. The values of part $fit_{distL}(t)$ (Equation (8)) are coded according to the current in-

formation from the ultrasonic sensor located on the left side in the direction of robot movement. Part $fit_{distL}(t)$ produces a maximum value when sufficiently distanced from an obstacle; with decreasing distance, its values fall. This part is used to ensure that sufficient distance is kept from obstacles.

The other part, $fit_{directL}(t)$, of Equation (8), increases the fitness value of an individual, provided there is no obstacle on the left side in the direction of robot movement and it fulfils the requirement of minimization of the number of commands to turn the robot right.

In the same way, the parts of the partial fitness values $fit_{distP}(t)$ and $fit_{directP}(t)$ are evaluated for an individual from the right local network. The only difference is the way information on the environment from the right ultrasonic sensor and is acquired from the controlling neuron of the right local network.

The last part of the calculation of the partial fitness value of an individual in Equation (8) is $fit_{crashL}(t)$. The value of this part is equal to 1 when a simulated collision is detected, otherwise it is 0. The optical sensor, which detects a simulated collision, provides the acquired information for both individuals at the same time. A collision for the robot is undesirable, therefore we defined this part of the calculation of the partial fitness value of an individual as $(1 - fit_{crash_L}(t))$.

The above-described parts of Equation (8) for the calculation of partial fitness values of individuals were devised for an evolutionary adaptation of a parallel combination of left and right local networks with respect to available hardware as well as to the type and number of sensors.

As an evolutionary optimization algorithm a simplified genetic algorithm was employed. In order to generate new individuals (children) in the populations (of both left and right local networks) we used a modified mutation operator only. We did not use a genetic crossover operatator in our implementation. Also, the selection of individuals from the population is not realized through a quasi-random selection. All individuals in the population have an equal chance of being selected, irrespective of their fitness value. For this reason, a genetic algorithm at the beginning of an evolutionary adaptation of local networks is little effective and what is more, a mutation-generated individual can have lower fitness value than the original one.

The former deficiency was partially eliminated through a modification in mutation. The traditional mutation changes the value of only one or two elements (bits) of the genetic string of an individual. In our mutation, within the first 10 minutes of the evolutionary adaptation of the networks we dynamically determine the number of elements of the genetic string of an individual, the value of which then changes (mutates).

The latter deficiency concerning a possibility of generating a weaker individual (the one with lower titness value) in comparison to the individuals of the current population and the introduction of this individual into a new generation was eliminated by the following condition: If the fitness value of a generated individual is smaller than that of the weakest individual of the current population, then this individual will not be introduced into a population and it will be removed from further consideration.

Our simplified genetic algorithm, designed for the adaptation of a parallel combination of local spiking networks, was implemented into a navigation system (left and right local spiking networks and a genetic algorithm form the basis of the proposed navigation system) of a mobile robot and was experimentally tested in customized conditions.

4 EXPERIMENTAL VERIFICATION OF THE APPLICATIONS OF SPIKING NEURAL NETWORKS

In order to explore the reliability of the implementation of the proposed architecture of a parallel combination of left and right spiking neural networks, we conducted a series of 4 experiments. Prior to experiments proper we carried out simulation in the Matlab development environment. Thus, we were able to experimentally set the parameters of the applied spiking neural network as well as the neuron model.

In all, we have implemented four versions of applied spiking neural networks. The first version used one spiking neural network, receiving information from one ultrasonic sensor. The second version differed from the first one only in that it used two ultrasonic sensors. The third and the fourth applied versions were the prototypes of our proposed navigation system. Both of these versions of the navigation system used two local spiking networks; the first one received information about the environment from two ultrasonic sensors, the second received this information from a miniature camera. An overview of all realized implementations is shown in Table 1.

Implementation abbreviation	Number and type of sensors	Type and number of neural networks	Notes
1S 1SNN	1 ultrasonic sensor, 1 optical sensor	1 spiking neural network	
2S 1SNN	2 ultrasonic sensors, 1 optical sensor	1 spiking neural network	
2S 2LSNN	2 ultrasonic sensors, 1 optical sensor	2 spiking neural networks – parallel combination of left and right local spiking networks	Our proposed implemented prototype
C 2S 2LSNN	1 wireless camera, 2 ultrasonic sensors, 1 optical sensor	2 spiking neural networks – parallel combination of left and right local spiking networks	Our proposed implemented prototype

Table 1. Overview of all implementations

In order to realize experiments with the implemented navigation systems we created two scene types. Both had a rectangular layout. The scenes only differed from each other in the colour patterns (texture) of their surfaces and to a minimum extent in their size as well. The mobile robot moved autonomously within the

confines of the scene according to the controlling commands of the navigation system. The neural networks, which formed the core of the navigation system, surveyed the scene environment through intelligent processing of current environment information to control the trajectory of the robot.

The first three implementations shown in Table 1 acquired the information about the environment through ultrasonic sensors only. For this reason, we were able to conduct experiments using white-surface walls. The size of the created scene (first part of Figure 6) was 170×80 cm.

Fig. 6. Part 1) the scene with white walls; Part 2) the scene with a texture of black and white stripes

The other type of scene (second part of Figure 6) designed for last implementation shown in Table 1 had to be rearranged. It was desirable that the image information from the camera stimulated a higher activity of the input units of the spiking neural network in relation to the decreasing distance of the camera from the wall texture [5, 28]. The alternative experiments which employed the calculation of the optical flow for impending collision detection were realized prior to the application of the neural networks proper [23, 24]. The texture of the inner walls of the scene was formed by vertical white and black stripes, the width of which was randomly generated from the interval of $\langle 0.5, 3 \rangle$ cm. The experiments revealed that the chosen width of vertical stripes in the wall texture has a bearing on the distance between an obstacle and the zone in which the navigation system detects this obstacle. The desired texture was achieved by covering the white area of the walls with black paper stripes. The size of the created scene was 160×70 cm.

Both scenes had identical floor coverings. This was made of white paper with the dimensions of 180×90 cm. On these, a zone was created with a quasi-rectangular layout, measuring 152×62 cm. The edge of the zone was lined with a black stripe with a width of 4.5 cm. Black ground reflects less light than white one. This phenomenon enables us to monitor easily whether the optical sensor is inside the zone or on its black edge. The minimum distance between edge of the zone and the walls of the scene was 4 cm. If the optical sensor passes over the black edge of the zone, this is regarded as a failure of the neural network response to input information and the navigation system issues a controlling command to change robot direction.

Tables 1 and 2 show the basic information and settings of parameters for the implementations of spiking neural networks. For clarity, we use abbreviated forms for the implementations. The settings of the parameters for implementations *2S 2LSNN* and *C 2S 2LSNN* in Table 2 (column 4 and 5) are given for one local spiking network, the same settings apply for the other local spiking network.

Implementation	1S 1SNN	2S 1SNN	2S 2LSNN one network	C 2S 2LSNN one network
Number of input units	6	12	6	10
Number of neurons	10	10	10	10
Membrane potential threshold	5	5	5	2
Number of activity cycles to determine response to one input vector	40	40	25	25
Time for evaluation of individual [s]	15	15	15	15

Table 2. Settings of the main parameters of neural networks in conducted experiments

During the testing and fine-tuning of the program we conducted a number of experiments and the NXT mobile robot covered several kilometers. In this article we only deal with one process of adaptation of each implementation of a spiking neural network. During the process of adaptation of spiking networks, the number

of evaluated individuals, the substituted ones as well as the fitness value of the individuals of every new generation are progressively saved. After the process of adaptation has been completed the codings of the topology of all the individuals of the last population generation and the overall time of adaptation are saved too. All acquired values are saved by a program in text files.

Figure 7 shows, in graphic form, the progress of the development of the average fitness value of individuals of each population generation which developed during the process of adaptation of spiking neural networks. The abbreviated designation *2S 2LSNN – LN* refers to the left local spiking network in the implementation *2S 2LSNN*. Similar abbreviated designations were used for all local spiking networks of our implementations *2S 2LSNN* and *C 2S 2LSNN*. Starting from the top, the first pair represents the *1S 1SNN* and *2S 1SNN* implementations, the middle pair refers to the *2S 2LSNN* implementation and the last pair represents the *C 2S 2SNN* implementation. The vertical columns in the form of "I" represent the difference of the values between the fittest and the weakest individuals in the same population generation. On the basis of the behavior of the acquired values we can say that during the adaptation the average fitness value of individuals gradually increased and the difference between the fitness values of individuals simultaneously decreased (drop in diversity). The individuals[2] were becoming stable during the adaptation.

Having adapted spiking networks to the scene environment we, once again, conducted a series of experiments. In these, we observed the behavior and reliability of the navigation system after the previous adaptation of spiking networks. The experiments were realized for all four implementations of spiking networks. Each activity of the NXT mobile robot lasted 5 minutes and during this time the topology of spiking networks was coded by a selection of the fittest individuals from the last acquired generation of populations. For the *2S 2LSNN* and *C 2S 2SNN* implementations we selected the pairs of the fittest individuals from the populations of the left and right spiking networks.

During these experiments we saved the information about the number of left and right turns of the NXT mobile robot. Through the optical sensor we detected the state in which the robot left the designated zone. Thus we simulated the failure of spiking network/s response to the input information about the environment. We treated this event as an undesirable collision of the robot with the wall. For each implementation, we conducted a minimum of six series of robot activities (runs). From the acquired values we calculated average values of the events of the navigation system which controlled the movement of the NXT robot. The results of this series of experiments are shown in Table 3, which gives information on the reliability and behavior of the navigation system during 5 minutes of robot activity.

The results from Table 3 show that the *1S 1SNN* implementation had a high degree of unreliability. This was caused by the fact that it only had one ultrasonic sensor. Being placed on the front side of the robot in the direction of its longitudinal

[2] The population of one spiking neural network comprised 10 individuals. This value was constant in all implementations.

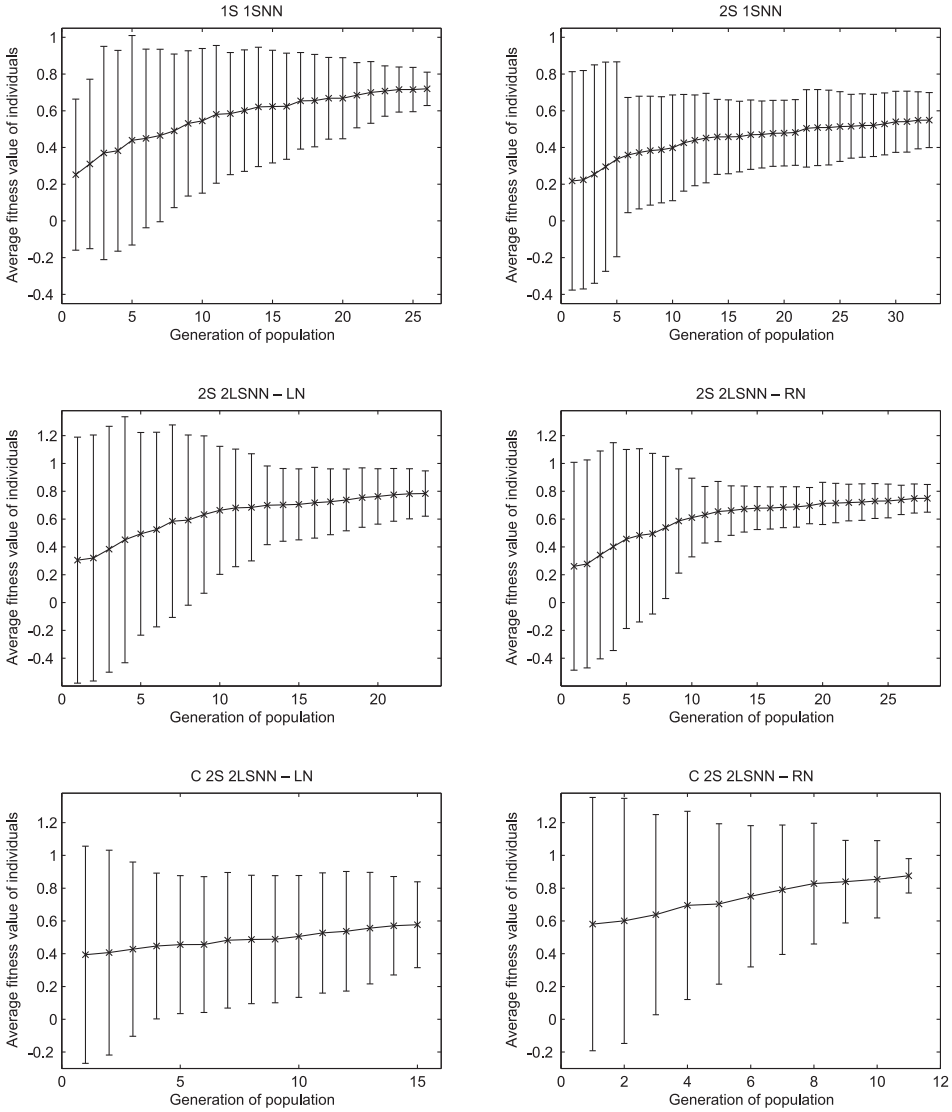

Fig. 7. The development of the fitness value of individuals during the adaptation of spiking networks

Implementation	Left turns	Right turns	Collisions detected by optical sensor
1S 1SNN	1	36	32
2S 1SNN	27	54	7
2S 2LSNN	70	34	5
C 2S 2LSNN	126	7	6

Table 3. The average values of the events of the navigation system after the adaptation of spiking networks

axis, one sensor was not able to correctly evaluate the distance from an obstacle in the cases when the robot approached the wall from either left or right side in the direction of its movement. In such cases, the neural network received incorrect information and the response of the navigation system was incorrect.

The following two implementations, *2S 1SNN* and *2S 2LSNN*, coded the information from the environment via two ultrasonic sensors. Both implementations showed a satisfactory degree of reliability and they can be used as the navigation system of a mobile robot. It has to be said that the width of the walls of the scenes is relatively small in relation to dimensions of our robot and controlling robot movement is a demanding task for the navigation system. Moreover, the *2S 2LSNN* implementation (the proposed combination of two local spiking networks) in comparison to the *2S 1SNN* implementation (using only one spiking network) has a cognitive property which is able to determine the turn of the robot correctly face-to-face with the wall of the scene.

Since we were not able to objectively record through our program the property of the navigation system to change correctly robot direction face-to-face with a scene wall, we calculated the correct and incorrect turns from video footage. This video recording contained five minutes of activity for each of the implementations. During the *2S 1SNN* implementation the robot made 77 turns, of which 60 % were correct and 40 % incorrect. During the *2S 2LSNN* implementation the robot made 94 turns, of which 93 % were correct and 7 % incorrect. While testing implementation *2S 1SNN* the robot kept turning almost exclusively in one direction, irrespectively of the fact whether it approached the wall from the left or right side to its movement direction. Conversely, the incorrect robot turns in the *2S 2LSNN* implementation occurred only in critical cases, when the robot found itself in the corners of the scene or in situations when it approached the wall in a nearly perpendicular direction.

The last implementation, *C 2S 2LSNN*, was capable of evaluating an impending collision with the wall of the scene, but it was unable to determine a correct turn of the mobile robot. The *C 2S 2LSNN* implementation differed from the previous three ones in the way it updates the input vector of spiking networks. This implementation does not use coded information from ultrasonic sensors for updating the input units of spiking networks, but uses adjusted image information from a miniature camera. The ultrasonic sensors are used only to calculate the fitness value of individuals. Because of time limitations, we did not realize possible modifications of this implementation and we did not attempt to improve it further.

5 CONCLUSIONS

In this work we applied spiking neural networks in order to control an autonomous mobile robot in a scene environment. In all, we realized 4 versions of applied neural networks. All the versions of the applications were implemented in real conditions in the form of the navigation system of an autonomous mobile robot. In two of these versions, we applied our own architecture using a parallel combination of left and right local spiking neural network.

Both versions of our proposed navigation system used two local spiking networks. One version received the information about the environment from two ultrasonic sensors and showed a satisfactory degree of reliability. Our navigation system was able to make the right decision and turned correctly in front of a detected obstacle. The second version, which received information about the environment from a miniature camera, was capable of evaluating an impending collision with the wall of the scene, but it was unable to determine a correct change of movement.

The series of conducted experiments and their results showed the reliability of the proposed combination of two local spiking neural networks in the implementations of the robot navigation system. In addition, the prototype of the proposed navigation system possesses features of an on-line adaptation of spiking neural networks and it is suitable for an implementation in evolutionary hardware. In the series of experiments, the system achieved favourable results even during the adaptation of networks themselves. The navigation system detected the presence of obstacles and with a change of the environment of the scene, the spiking networks dynamically adapted to the realized changes. In this case, the difference between the adaptation phase and the phase of neural network life began to disappear. On balance, this system property is beneficial, because the majority of applications of neural networks are limited by their adaptation phase.

What remains unsolved is the modification of the last implementation, in which the current information from the scene environment is coded through the image information from a miniature camera. In this implementation we suggest to modify the part of the objective function of the genetic algorithm that evaluates the correctness of robot turns face to face with an obstacle. In addition, we propose that the number of input neurons be increased as well as to modify the values of system parameters.

In addition, we suggest to make changes to our simplified genetic algorithm so that it becomes a fully-fledged genetic algorithm. In doing this, we would keep the modified mutation operator as well as the condition which would prevent a new generation of individuals, weaker than those from the previous population generation, from being developed. Although this change would make higher demands on the computational performance and memory capacity of applied hardware, it would speed up the evolutionary development of neural networks.

Another alternative would involve the use of frequency coding based on the average number of impulses emitted by several neurons in the output layer in response to one external stimulus. From the point of view of our experiments, in which a spik-

ing neural network was used to set up autonomous navigation (a neural network forms a system for obstacle detection and determination of movement direction), the alternative of using several neurons for obstacle detection would probably work more effectively and reliably than using one neuron separately. When the calculated average exceeds of the experimentally set threshold value, the event is taken as a signal of proximity of an obstacle in the direction from which the external input was received.

Acknowledgement

The author of this article would like to thank the Slovak Grant Agency for Science (Grant No. 1/4391/07) which financially contributed to the publication of this article.

REFERENCES

[1] Analogic Lab Publications [online]. [quoted 2008-08-21]. Available on: `http://lab.analogic.sztaki.hu/publications.html`.

[2] Balog, T.: The Peculiarities of Insectal Eyes [Osobitosti očí živočíchov – in Slovak] [online]. [quoted 2008-08-21]. Available on: `http://balog.blog.sme.sk/c/119868/Osobitosti-oci-zivocichov.html`.

[3] Eclipse – An Open Development Platform [online]. [quoted 2008-08-21]. Available on: `http://www.eclipse.org`.

[4] Floreano, D.—Mattiussi, C.: Evolution of Spiking Neural Controllers for Autonomous Vision-Based Robots. In: Evolutionary Robotics from Intelligent Robotics to Artificial Life, International Symposium, ER 2001, Tokyo, Japan, October 2001, Springer 2001. ISBN 3540427376. pp. 38–61.

[5] Floreano, D.—Epars, Y.—Zufferey, J. C.—Mattiussi, C.: Evolution of Spiking Neural Circuits in AutonomousMobile Robots. In: International Journal of Intelligent Systems, Vol. 21, 2006, No. 9, pp. 1005–1024.

[6] Floreano, D.—Husbands, P.—Nolfi, S.: Evolutionary Robotics. In: Handbook of Robotics, Springer Verlag, Berlin 2008.

[7] Genetic Algorithm and Direct Search Toolbox 2 – Users Guide. The MathWorks, Inc. 2008.

[8] Gerstner, W.—Kistler, W. M.: Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press 2002. ISBN 0-521-89079-9.

[9] Jacobs, R. A.—Jordan, M. I.: Adaptive Mixtures of Local Experts. In: Neural Computation 3, Cambridge: MIT Press 1991. pp. 79–87.

[10] Jordan, M. I.—Jacobs, R. A.: Hierarchical Mixtures of Experts and the EM Algorithm. In: Graphical Models, Foundations of Neural Computation, Cambridge: MIT Press 2001, ISBN-10: 0-262-60042-0, pp. 254–288.

[11] Kistler, W. M.—Hemmen, J. L.: Modeling Synaptic Plasticity in Conjunction With the Timing of Pre- and Postsynaptic Potentials. In: Neural Comput., Vol. 12, 2000, pp. 385–405.

[12] Kvasnička, V.—Pospíchal, J.—Tiňo, P.: Evolutionary Algorithms [Evolučné algoritmy – in Slovak]. Bratislava, STU 2000, ISBN 8022713775.

[13] Kvasnička, V.—Pospíchal, J.: Connectivism and the Modeling of Cognitive Processes [Konekcionizmus a modelovanie kognitívnych procesov]. In: The Cognitive Sciences [Kognitvne vedy – in Slovak]. Bratislava: Kalligram, 2002.

[14] Laboratory of Intelligent Systems. Ecole Polytechnique Fdrale de Lausanne, Switzerland [online]. [quoted 2008-08-21]. Available on: http://lis.epfl.ch.

[15] LEGO. Mindstorms NXT [online]. [quoted 2008-08-21]. Available on: http://mindstorms.lego.com/eng/default.aspx.

[16] LEJOS. Java for LEGO Mindstorms [online]. [quoted 2008-08-21]. Available on: http://lejos.sourceforge.net/index.php.

[17] Nolfi, S.—Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Cambridge, Massachusetts: MIT Press, 2000. ISBN 0262140705.

[18] Pechlát, J.: Insect Vision [Zrak hmyzu – in Slovak] [online]. [quoted 2008-08-21]. Available on: http://hmyz.net/ocihmyzu.htm.

[19] PlaNet Alife. Introduction to the EHW [Úvod do EHW – in Slovak] [online]. [quoted 2008-08-21]. Available on: http://alife.tuke.sk/index.php?clanok=574.

[20] Rekeczky, C.—Bálya, D.—Timár, G.—Szatmári, I.: BioInspired Flight Control and Visual Search With CNN Technology. In: IEEE International Symposium on Circuits and Systems ISCAS 2003. Piscataway, N. J.: IEEE, 2003. ISBN 0780377613, pp. 774–777.

[21] Rubin, J.—Lee, D. D.—Sompolinsky, H.: Equilibrium Properties of Temporally Asymmetric Hebbian Plasticity. In: Physical Review Letters, Vol. 86, 2001. pp. 364–367.

[22] Sun Developer Network (SDN). Download JMF 2.1.1e Software [online]. [quoted 2008-08-20]. Available on: http://java.sun.com/products/java-media/jmf/2.1.1/download.html.

[23] Trhan, P.—Mala, M.—Klaučo, R.: The BioInspired Optical Flow Detector. In: Informatics '07, Proceedings of the Ninth International Conference on Informatics, Družba Hotel, Bratislava, June 21–22, 2007, SSAKI, Bratislava 2007, ISBN 978-80-969243-7-0. pp. 77–80.

[24] Trhan, P.: Bio-Inspired VisionBased Navigation Systems. In: Applied Electronics 2007, University of West Bohemia in Pilsen, Czech Republic, September 5–6, 2007, Pilsen: University of West Bohemia in Pilsen, 2007. ISBN 987-80-7043-537-3. pp. 223–226.

[25] Trhan, P.—Škultéty, J.: Simulation of Evolutionary Development of a Spiking Neural Network [Simulácia evolučného vývoja impulznej neurónovej siete – in Slovak]. In: Technical Computing Prague 2007, international conference, Czech Technical University in Prague Congress Center, Czech Republic, November 14, 2007, Humusoft, Prague 2007. ISBN 978-80-7080-658-6.

[26] Weishaupl, T.—Schikuta, E.: How to Parallelize Cellular Neural Networks on Cluster Architectures. In: International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '04), 2004.

[27] Ziemke, T.: Remembering How to Behave: Recurrent Neural Networks for Adaptive Robot Behavior. In: Recurrent Neural Networks: Design and Applications, CRC Press 2000. ISBN 0849371813. pp. 355–390.

[28] Zufferey, J. C.: Bio-Inspired Vision-Based Flying Robots. Ph. D. Thesis, Lausanne, EPF, 2005.

Peter Trhan graduated at Matej Bel University, Faculty of Natural Sciences in 1998 obtaining a M. Sc. degree in mathematics and computer science (teacher training qualifications). He works as an Assistant Professor at the Department of Computer Science at the Faculty of Natural Sciences, Matej Bel University in Banská Bystrica. He had also beem teaching at the Department of Graphics, the Faculty of Fine Arts at the Academy of Arts in Banská Bystrica and at the Institute of Applied Communication at Matej Bel University. Currently he is a Ph. D. student at the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava. His professional areas of interest include neural networks, database systems and computer graphics. He has taken part in six grant projects as a scientific co-worker. The most important of these were three Vega and one Tempus Metea projects.