

# EMERGENCE OF AN AUTONOMOUS ROBOT'S BEHAVIOR

<sup>1</sup>Eva Volna

<sup>2</sup>Martin Kotyrba

<sup>2</sup>Martin Žáček

<sup>1</sup>Adam Bartoň

<sup>1</sup>Department of Informatics and Computers

<sup>2</sup>Centre of Excellence IT4Innovations division of the University of Ostrava, IRAFM

University of Ostrava

70103, Ostrava, Czech Republic

E-mail: eva.volna@osu.cz

E-mail: martin.kotyrba@osu.cz

E-mail: martin.zacek@osu.cz

E-mail: P14065@student.osu.cz

## KEYWORDS

Emergence, Artificial Neural Network, Robotic System, Robot Control.

## ABSTRACT

The paper deals with the emergence of behaviour of an autonomous robot controlled by a Back-Propagation neural network. The theoretical part focuses on robotic architecture, and on the issues of emergence and connectionist networks. The output of the experimental study is a 2D application which allows to model the movements of the robot in a simulation environment with the possibility of configuration variability. In addition, several robot type designs are proposed and tested in the experimental part of the article which address the role of movement in an environment aimed at avoiding obstacles.

## THE BASIC ARCHITECTURE OF THE ROBOTIC SYSTEM

A motion control system, sensor systems, a cognitive system and operation and communication control systems are typical components of a fundamental hardware architecture that may be used in a robotic system (Figure 1). This sorting is a general sorting for autonomous mobile robot [2]. In relation to the architecture, it is appropriate to mention the term autonomy of the robot, i.e. the definition of the degree of independence of the human operator. Robotic systems are divided according to the degree of autonomy: a) *controlled* robot that does not make its own decisions - robot operations are controlled by a human operator, b) *operated* robot, performing predetermined operations, c) *autonomous* robot, reaching targets according to a predetermined procedure d) *intelligent* robot, which is able to independently set its own goals [2].

### The motion subsystem

In principle, the motion subsystem can be divided into implementer of plans performing a similar function as a controller in a computer and effectors, i.e. operational

parts as engines that perform actions in the environment. Applications of a locomotor subsystem could be divided depending on the environment - territorial, flying, water, etc.

### The sensor subsystem

The sensor subsystem consists of two basic parts: the part of the receptor and the part of retrieval and processing of data. Sensors transform different kinds of input signals into physical suitably representable information, e.g. electrical signals. Such information is pre-processed by sampling, filtering etc. according to system requirements. Sensors can be divided into internal and external ones. *Internal sensors*, which are aimed at measuring internal states of the robot such as battery, temperature, position, speed and they also allow to perform diagnostics. *External sensors* measure the environmental situation, usually for navigation - the distance of obstacles in the environment or robot positions. Sensors also could be divided into *active* and *passive* or *tactile* and *contactless* [4, 7].

### The cognitive subsystem

The brain of a robotic system creates a cognitive (control) subsystem that performs deeper analysis of data and this data is further classified, see module 'perception and understanding' in Figure 1, which implements solution of tasks and determines a sequence of actions leading to the achievement of defined objectives. The outcome is a plan (Figure 1 - block of 'a task solving and planning'). The previous tasks directly depend on the internal 'model of an environment', which is a memory of the external environment, Figure 1 - block of a 'model of environment'. Combining a block perception, block environment, block of task solving and block of planning provides the highest feedback - *cognitive feedback*. The plan of action, which is passed the motion system, can include conditional branching, which enables various alternatives for other activities implemented on the basis of data obtained in real time. The above-described scheme may be different for various robots.

### The communication subsystem

The communication subsystem is used mainly for communication with a human operator, who is allowed to manage the flow of information at a higher level and to diagnose the current state of the robot and also to allow overall communication with the robot. This is usually done using a software application or via a terminal displaying current values of sensors and overall conditions of the robot.

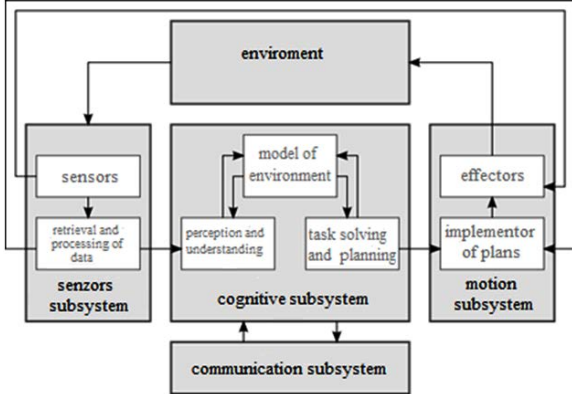


Figure 1: Block diagram of the robot [2]

### EMERGENCE

Emergence refers to the existence or formation of collective behaviors — what the constituent parts of the system do together that they would not do alone. In describing collective behaviors, emergence refers to how collective properties arise from the properties of parts, how behavior at a larger scale arises from the detailed structure, behavior and relationships at a finer scale. The emergent phenomenon is not pre-planned, although we can predict its presence [3]. We can view emergence from two levels: from the perspective of the analysis, which says that emergence is only an appearance because of the ignorance of the equations describing the problem and therefore we cannot make conclusions about the system or predict its behavior. The opposite view is phenomenologically describing emergent effect as an act of consciousness, the means for grasping the phenomenon and maintaining its cohesion.

### BACKPROPAGATION NEURAL NETWORKS

The very general nature of the Back propagation training method means that a Back propagation net (a multilayer, feedforward net trained by Back propagation) can be used to solve problems in many areas [5, 6] describes the network in detail. It is simply a gradient descent method to minimize the total squared error of the output computed by the net. The training of a network by Back propagation involves three stages: the feedforward of the input training pattern, the calculation and Back propagation of the associated error, and the adjustment of the weights. After training, application of the net involves only the computations of the feedforward phase. A multilayer neural network with one layer of hidden units (the  $Z$  units) is shown in Figure 2. The output units (the  $Y$  units) and the hidden units also may have biases (as shown). The bias on a typical output unit

$Y_k$  is denoted by  $w_{0k}$ ; the bias on a typical hidden unit  $Z_j$  is denoted  $w_{0j}$ . These bias terms act like weights on connections from units whose output is always 1.

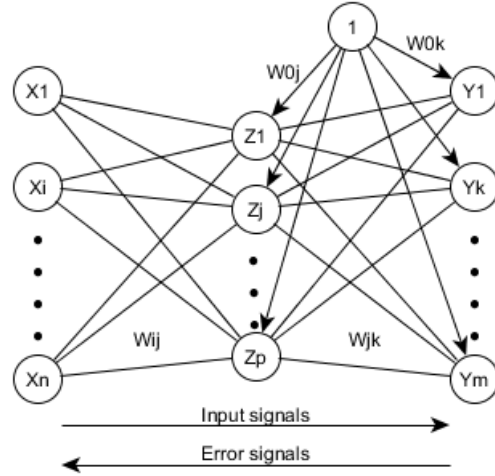


Figure 2: A Back propagation network architecture

The neurons in the network use an activation function to compute their response to the input. One of the most common activation functions is the binary sigmoid (1)

$$f(x) = \frac{1}{1 + \exp(-x)} \quad (1)$$

with

$$f'(x) = f(x)[1 - f(x)] \quad (2)$$

During feedforward, each input unit  $X_i$  receives an input signal and broadcasts it to all of the hidden units  $Z_1, \dots, Z_p$ . Each hidden unit in turn computes its activation (3)

$$z_j = f(w_{0j} + \sum_i x_i w_{ij}) = f(y_{in_k}) \quad (3)$$

to each output unit. Each output unit ( $Y_k$ ) computes its activation (4)

$$y_k = f(w_{0k} + \sum_j z_j w_{jk}) = f(z_{in_j}) \quad (4)$$

which forms the output of the network. During training, each output unit compares its computed activation  $y_k$  with its target value  $t_k$  to determine the associated error for that pattern with that unit. Based on this error, the factor  $\delta_k$  ( $k = 1, \dots, m$ ) is computed (5).

$$\delta_k = (t_k - y_k) f'(y_{in_k}) \quad (5)$$

$\delta_k$  is used to distribute the error at output unit  $Y_k$  back to all units in the previous layer (the hidden units that are connected to  $Y_k$ ). It is also used (later) to update the weights between the output and the hidden layer. In a similar manner, the factor  $\delta_j$  ( $j = 1, \dots, p$ ) is computed for each hidden unit  $Z_j$  (6):

$$\delta_j = (\sum_k \delta_k w_{jk}) f'(z_{in_j}) \quad (6)$$

It is not necessary to propagate the error back to the input layer, but  $\delta_j$  is used to update the weights between the hidden layer and the input layer. After all of the  $\delta$  factors have been determined, the weights for all layers are adjusted simultaneously. The learning rate parameter  $\alpha \in (0,1]$  is used to regulate the network's sensitivity. The adjustment to the weight  $w_{jk}$  (from hidden unit  $Z_j$  to output unit  $Y_k$ ) is based on the factor  $\delta_k$  and the activation  $z_j$  of the hidden unit  $Z_j$ , (7)

$$\Delta_{w_{jk}} = \alpha \delta_k z_j \quad (7)$$

The adjustment to the weight  $w_{ij}$  (from input unit  $X_i$  to hidden unit  $Z_j$ ) is based on the factor  $\delta_j$  and the activation  $x_i$  of the input unit, (8).

$$\Delta_{w_{ij}} = \alpha \delta_j x_i \quad (8)$$

The bias weights are adjusted in the same way bearing in mind that the bias value is 1, (9).

$$\Delta_{w_{0j}} = \alpha \delta_j, \Delta_{w_{0k}} = \alpha \delta_k \quad (9)$$

## EXPERIMENTAL STUDY

An entity represents a virtual robot having  $n$  defined proximity sensors that detect obstacles. An entity is capable of three basic operations for movement in a virtual environment: *move forward*, *turn right*, *turn left*. Due to the nature of the task, i.e. avoiding objects, reactive architecture was chosen. It means that the entity has no internal state or memory and responds 'only' to stimuli from the outside world. The chosen architecture is shown in Figure 3.

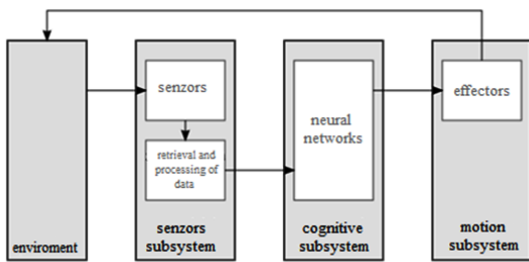


Figure 3: Designing the architecture of the entity [1]

An entity can be represented graphically as a circle. Since the entity moves forward, it is a significant the angle of its 'view'. This angle is graphically represented by plotting a straight line starting at the center of the entity and ending at the edge in the direction of rotation of the entity. Each entity can have  $n$  proximity sensors. Proximity sensors are non-contact devices that provide information about the presence of objects without having to interact with them. In order to place the sensors at different locations on the surface, they dispose by angle's setting from which will be drawn

from the surface of the entity. When creating sensors of an entity, it is possible to follow the diagram from Figure 4, where the entity is presented 'looking' at an angle  $a$ , having one sensor with an angle  $0^\circ$ . The sensor is defined by its range  $r$ , number of rays  $n$  outgoing from sensors and forming its detection part by coefficient of a beam spread  $s$ . The sensitivity of the sensor is given in percentage and determines the number of rays that have to collide with an obstacle to sensor detect obstacles.

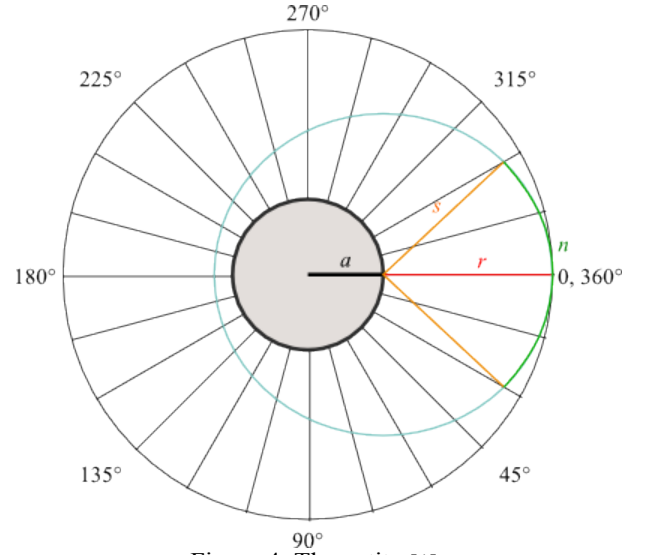


Figure 4: The entity [1]

Sensor rays are implemented by firing the sensor rays from the point that defines the angle of the sensor position to a remote point detectable range of the sensor  $r$ . If it bumps into an obstacle and does not reach the range of  $r$ , then the ray is terminated and the collision is detected. Preprocessing of the input signal to the neural network can be performed by sensors due to the faster response of the network as shown in Figure 3.

## Simulation environments

To test the robot, we have designed five different simulation environments representing different types of situations in which the entity is placed. Simulation environment are raster Figures 5 – 9. The basic operation of an entity is its movement in the environment, i.e. a forward movement.

The test whether the entity will move forward with no or minimal lateral movements was performed in the environment in Figure 5, representing the 'corridor'. The aim of this environment is to test the movement forward and perform turn at the end of the 'corridor' without bumping into 'wall'.

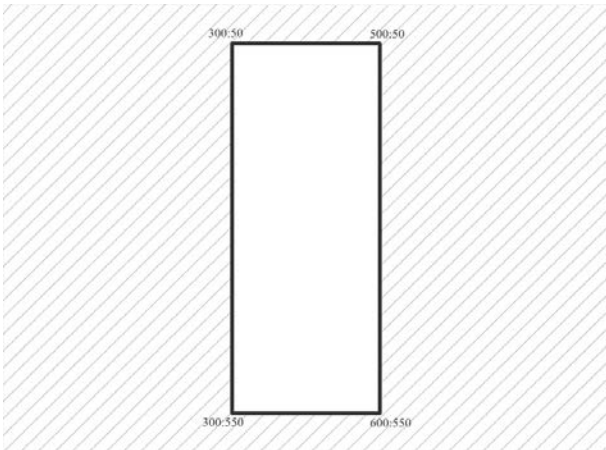


Figure 5: The simulation environment – the corridor (S1)

The aim of the following environments Figure 6 is to verify whether the entity is able to turn round into the empty space to the right and then to continue to the end of the environments, where a rotation (without colliding with the critical corner) will be performed.

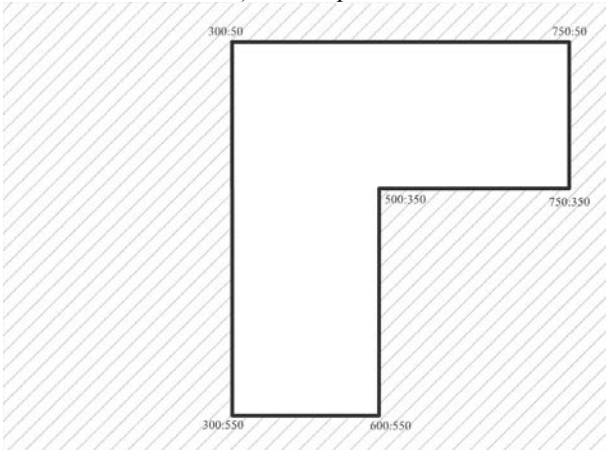


Figure 6: The simulation environment – the turn (S2)

The following environment Figure 7 tests a case when an entity chooses between two possible routes. The entity should always perform rotation in the same direction. It is also desirable that the entity is able to get back to the starting area.

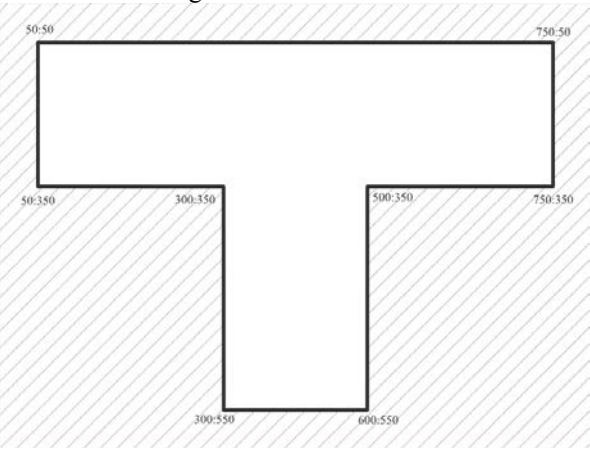


Figure 7: The simulation environment – the crossroad (S3)

We have tested in this environment Figure 8, if the entity is able to avoid an obstacle in the forward direction. It is necessary to get around an obstacle while not hitting it.

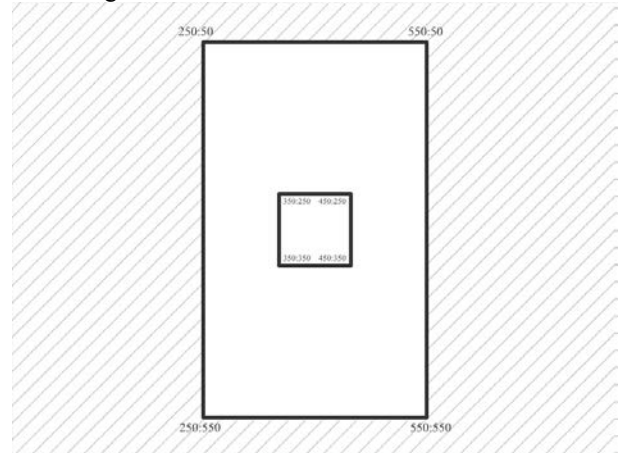


Figure 8: The simulation environment – the obstacle (S4)

The following environment tests an ability to move the entity in a 'random' environment Figure 9. In particular, it should not become stagnant or collide with obstacles. The environment includes obstacles in space, critical corners of 'room', and a circular area with difficult access (the entity should be able to leave the area after entry).

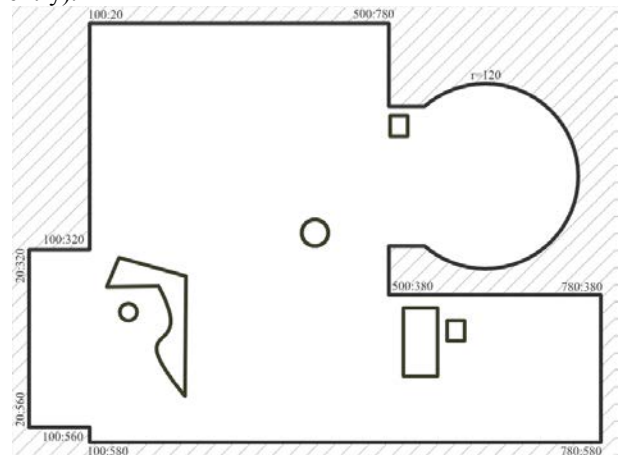


Figure 9: The simulation environment – a free movement (S5)

### Parameters of the simulation

In our experimental study, we have tested the entity with different numbers of sensors: 2, 3 and 5. There were chosen the following criteria for assessing the effectiveness of entities: *number of collisions with obstacles*, *number of steps forward*, and *number of turn to the left and to the right*. The parameters of sensor entities in all experiments are as follows:

- default position  $x$ : 50
- default position  $y$ : 50
- angle of a view: 0
- speed: 3

The parameters of neural networks in all experiments are as follows:

- learning rate: 0.85
- momentum: 0.9
- slope coefficient:  $\langle -1, 2 \rangle$

*The 2-sensors' entity* (Figure 10). This entity has two sensors located on the surface at angles  $338^\circ$  and  $22^\circ$ . This entity should be able to move and avoid obstacles. The intersection of the detection parts of both sensors results into a detection area, which is suitable to move forward, if collision of sensors is not detected.

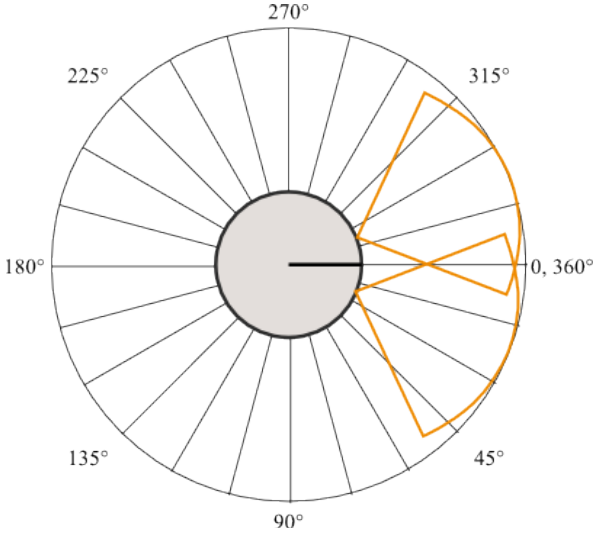


Figure 10: The 2-sensors' entity [1]

Neural network parameters are the following:

- Network topology: 2 – 7 - 3
- training set, see Table 1

Table 1: The 2-sensors' entity - the training set (4 patterns)

Nr.	$I1$	$I2$	$O1$	$O2$	$O3$
1	0	0	0	1	0
2	1	0	0	1	1
3	0	1	1	1	0
4	1	1	0	0	1

In Table 1 columns  $I1$  and  $I2$  correspond to the states of sensors when the value of 1 means that the sensor detects an obstacle, and value of 0 means that the sensor does not detect an obstacle. The remaining columns denote actions of the entity. The column labeled  $O1$  corresponds to turn left,  $O2$  corresponds to move forward, and  $O3$  corresponds to turn right. In these columns a value of 1 indicates to perform the action and a value of 0 indicates not to perform the action. We have also used this marking in our following experiments with the 3-sensors' entity and the 5-sensors' entity. The neural network was adapted within 1435 cycles with an error of 0.001.

*The 3-sensors' entity* (Figure 11). This entity has two sensors located on the surface at angles  $315^\circ$ ,  $0^\circ$  and

$45^\circ$ . Thanks to a higher number of sensors, we have also tested cases when neural networks do not receive all possible patterns, but only a few selected patterns from the training set in order to the neural network would be able to generalize a motion of the entity .

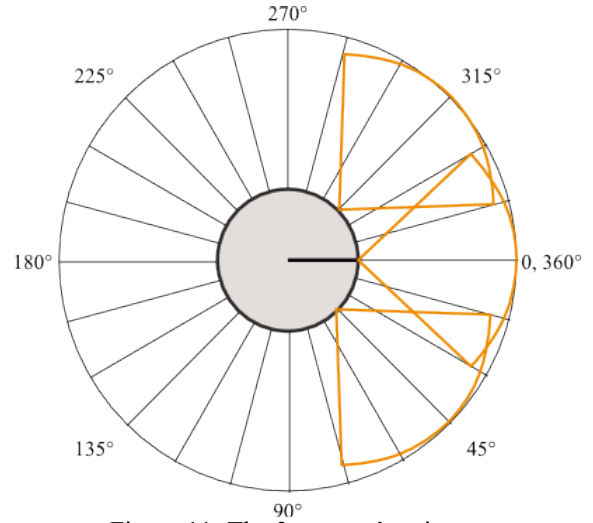


Figure 11: The 3-sensors' entity [1]

Neural network parameters are the following:

- Network topology: 3 – 10 - 3
- training set, see Table 2

Table 2: The 3-sensors' entity – the training set (8 patterns)

Nr.	$I1$	$I2$	$I3$	$O1$	$O2$	$O3$
1	0	0	0	0	1	0
2	1	0	0	0	1	1
3	0	1	0	0	0	1
4	1	1	0	0	0	1
5	0	0	1	1	1	0
6	1	0	1	0	1	0
7	0	1	1	1	0	0
8	1	1	1	0	0	1

The neural network was adapted within 1772 cycles with an error of 0.001. This network was also adapted with two incomplete training sets. The training set  $N1\_3\_Sensors$  consists of 6 patterns from Table 2 (rows Nr. 1, 2, 4, 5, 7, and 8). The network was adapted within 1448 cycles with an error of 0.001. The training set  $N2\_3\_Sensors$  consists of 4 patterns from Table 2 (rows Nr. 1, 2, 5, and 8). The network was adapted within 978 cycles with an error of 0.001.

*The 5-sensors' entity* (Figure 12). This entity has two sensors located on the surface at angles  $270^\circ$ ,  $315^\circ$ ,  $0^\circ$ ,  $45^\circ$  and  $90^\circ$ . As in the previous case, we have also tested cases, when neural networks do not receive all possible patterns but only a few selected patterns from the training set.

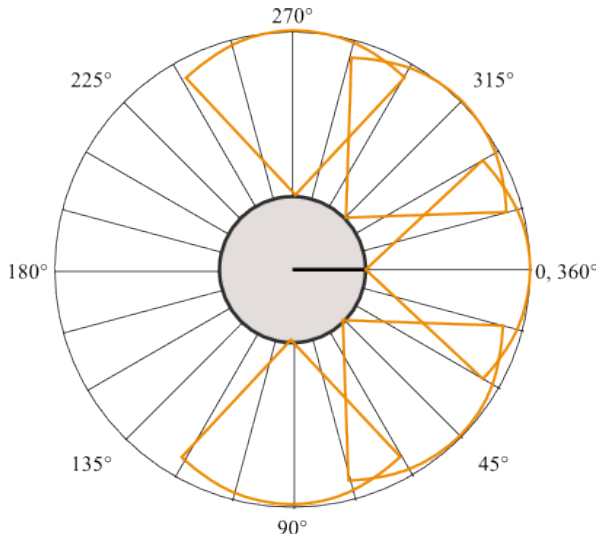


Figure 12: The 5-sensors' entity [1]

Neural network parameters are the following:

- Network topology: 5 – 16 - 3
- training set, see Table 3

Table 3: The 5-sensors' entity – the training set (32 patterns)

Nr.	I1	I2	I3	I4	I5	O1	O2	O3
1	0	0	0	0	0	0	1	0
2	1	0	0	0	0	0	1	0
3	0	1	0	0	0	0	1	1
4	1	1	0	0	0	0	1	1
5	0	0	1	0	0	0	0	1
6	1	0	1	0	0	0	0	1
7	0	1	1	0	0	0	0	1
8	1	1	1	0	0	0	0	1
9	0	0	0	1	0	1	1	0
10	1	0	0	1	0	1	1	0
11	0	1	0	1	0	0	1	0
12	1	1	0	1	0	0	1	0
13	0	0	1	1	0	1	0	0
14	1	0	1	1	0	0	0	1
15	0	1	1	1	0	0	0	1
16	1	1	1	1	0	0	0	1
17	0	0	0	0	1	0	1	0
18	1	0	0	0	1	0	1	0
19	0	1	0	0	1	0	1	1
20	1	1	0	0	1	0	1	1
21	0	0	1	0	1	1	0	0
22	1	0	1	0	1	0	0	1
23	0	1	1	0	1	1	0	0
24	1	1	1	0	1	0	0	1
25	0	0	0	1	1	1	1	0
26	1	0	0	1	1	1	1	0
27	0	1	0	1	1	0	1	0
28	1	1	0	1	1	0	1	0
29	0	0	1	1	1	1	0	0
30	1	0	1	1	1	1	0	0
31	0	1	1	1	1	1	0	0
32	1	1	1	1	1	0	0	1

The neural network was adapted within 1822 cycles with an error of 0.001. This network was also adapted with two incomplete training sets. The training set *N1\_5\_Sensors* consists of 15 patterns from Table 3 (rows Nr. 1-3, 5, 8-11, 15-20, and 29-32). The network was adapted within 1377 cycles with an error of 0.001. The training set *N2\_5\_Sensors* consists of 9 patterns from Table 3 (rows Nr. 1-5, 17, 25, 28, and 32). The network was adapted within 1515 cycles with an error of 0.001.

## OUTCOMES

Simulations with number 1 test all the mentioned entities in the environment called 'corridor' (Figure 5). Except for *N2\_entities*, all remaining entities were able to move from the starting area to the end of 'corridor' and perform turn without hitting the wall. *N2\_3\_Sensors* entity always stopped at the end of the corridor and *N2\_5\_Sensors* entity always stopped in the middle of the corridor.

The second simulation tests entities in an environment marked 'the turn' (Figure 6). Entities *N2\_3\_Sensors* and *N2\_5\_Sensors* stopped at the same place as in the previous simulation. All remaining types of entities fulfilled the test successfully.

In 'crossroads' (Figure 7) was tested decision making of the entities. Due to the absence of internal states or representation of the environment, the entities should always choose the same path. The entity always chose the same direction of movement without colliding with walls or corners in the environment but only *N1\_5\_Sensors* entity returned to the starting area. Entities *N3\_5\_Sensors* and *N2\_5\_Sensors* stopped at the same place as in previous simulations again. The remaining entities were moving straightforward between the left and right end of the simulation environment.

Avoiding the obstacle is tested in the environment 'obstacle' (Figure 8). Simulations showed that the entities are able to avoid an obstacle, with the exception of entities *N3\_5\_Sensors* and *N2\_5\_Sensors* that always stopped at the same places. A relatively longer time, this action lasted *N1\_5\_Sensors* entity that performed a lot of steps before it managed to enter the space at a suitable angle. Then it has already mastered its further movement in the environment like other entities.

The last simulation environment is 'free movement', which simulates a 'random' environment in which entities can find themselves (Figure 9). Entities *N3\_5\_Sensors* and *N2\_5\_Sensors* stopped at the same place as in previous simulations again. All remaining entities were moving in the environment without collisions. They were able to enter and again to get out from the 'circular room' with difficult access. The only one *N1\_5\_Sensors* entity managed to enter into the difficult access places, where other entities failed to enter.

The total results of simulations are shown in Table 2, where 'Y' means the successful simulation and 'N' means unsuccessful simulation.

Table 2: The results of simulations, where ‘Y’ means the successful simulation and ‘N’ means unsuccessful simulation

Entity	Number of Patterns	The simulation environment				
		S1	S2	S3	S4	S5
2 Sensors	4	Y	Y	N	Y	Y
3 Sensors	8	Y	Y	N	Y	Y
N1 3 Sensors	6	Y	Y	N	Y	Y
N2 3 Sensors	4	N	N	N	N	N
5 Sensors	32	Y	Y	N	Y	Y
N1 5 Sensors	18	Y	Y	Y	Y	Y
N2 5 Sensors	9	N	N	N	N	N

## CONCLUSION

The experimental results showed that the most effective entity appears entity *N1\_5\_Sensors*, which was able to handle all types of the proposed simulation tasks. There is a great influence of sensor settings, determining how much the entity can approach the obstacle. A suitable ratio range and sensitivity of sensors with speed, size and ability to turn may allow a great freedom of movement of the entities. A larger number of sensors allow you to define complex behaviors, but also *2\_Sensors* entity was able to move in space without hitting the subjects. A major impact on the behavior of entities was observed in the chosen training set of neural networks. The simulation results have shown that a small number of patterns or unsuitable choice of patterns leads to an unsatisfactory behavior of the entities.

## ACKNOWLEDGMENT

The research described here has been financially supported by University of Ostrava grant SGS/PřF/2015 and by the European Regional Development Fund in the IT4 Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

## REFERENCES

- [1] Barton, A. 2014. Emergence of an Autonomous Robot's Behaviour. Master thesis (in Czech). University of Ostrava. 81 p.
- [2] Dudek, G., Jenkin, M. 2010. *Computational principles of mobile robotics*. Cambridge: Cambridge university press.
- [3] Fromm, J. 2004. *The Emergence of Complexity*. Kassel: Kassel university press.
- [4] Maaref, H., Barret, C. 2002. Sensor-based navigation of a mobile robot in an indoor environment. *Robotics and Autonomous systems*, 38(1), 1-18.
- [5] Rojas, R. 1996. *Neural Networks: A Systematic Introduction*. Berlin : Springer.
- [6] Volná, E. 2013. *Introduction to Soft Computing*. bookboon.com Ltd. (available from <http://bookboon.com/en/introduction-to-soft-computing-ebook> )
- [7] Žáček, J., Janošek, M. 2014. Programmable control of Heating for Systems with Long Time Delays. Proceedings of the 2014 15th International Carpathian Control Conference (ICCC). Velké Karlovice: VŠB - Technical University of Ostrava, pp. 705-709.



**EVA VOLNA** is an associate professor at the Department of Computer Science at University of Ostrava, Czech Republic. Her interests include artificial intelligence, artificial neural networks, evolutionary algorithms, and cognitive science. She is an author of more than 50 papers in technical journals and proceedings of conferences.



**MARTIN KOTYRBA** is an assistant professor at the Department of Computer Science at the University of Ostrava, Czech Republic. His interests include artificial intelligence, formal logic, soft computing methods and fractals. He is an author of more than 30 papers in conference proceedings.



**MARTIN ZACEK** is an assistant professor at the Department of Computer Science at University of Ostrava, Czech Republic. His interests include formal logic, knowledge representation and artificial intelligence. He is an author of more than 30 papers in proceedings of conferences.



**ADAM BARTON** is a master student at the Department of Computer Science at University of Ostrava, Czech Republic. His interests include artificial intelligence, artificial neural networks, evolutionary algorithms, and robotics.