Control of autonomous robot behavior using data filtering through adaptive resonance theory

Adam Barton¹ · Eva Volna¹ · Martin Kotyrba¹

Received: 1 August 2017 / Accepted: 7 October 2017 / Published online: 2 November 2017
© The Author(s) 2017. This article is an open access publication

Abstract The aim of the article is to use neural networks to control autonomous robot behavior. The type of the controlling neural network was chosen a backpropagation neural network with a sigmoidal transfer function. The focus in this article is put on the use adaptive resonance theory (ART1) for data filtering. ART1 is used for preprocessing of the training set. This allows finding typical patterns in the full training set and thus covering the whole space of solutions. The neural network adapted by a reduced training set has a greater ability of generalization. The work also discusses the influence of vigilance parameter settings for filtering the training set. The proposed approach to data filtering through ART1 is experimentally verified to control the behavior of an autonomous robot in an unknown environment with varying degrees of difficulty regarding the location of obstacles. All obtained results are evaluated in the conclusion.

Keywords Adaptive resonance theory (ART) · Big data · Control neural network · Data filtering

1 Introduction

Nowadays, we recognize a wide range of robotic devices. It is still the subject of research their program part, namely arti-

cial intelligence. An interesting topic is the human brain as a model for such an intelligent system. Inspired by the structure of the human brain, an artificial neural network emerges. Like the brain, the artificial neural network contains many simple computational units, neurons, interconnected mutually to allow the transfer of the signal from the neuron to neurons. These local interactions between neurons then result in execution of global functions. Artificial neural networks are used to solve various problems with good results compared to other decision algorithms.

There are many different artificial neural networks, but we focus on feedforward networks. Through them, we control the behavior of a robotic device that should be able, on the basis of defined behavior, to derive an appropriate response to unknown situations. To build a physical robotic equipment, we used Lego Mindstorms EV3.

1.1 Important statement

This paper is an extended version of our conference paper [3], which was at least 30% new. We have extended the abstract and the text of chapter 5 and 6. Chapters 1, 2, and 4 were added. We have also added corresponding references.

2 Autonomous robot control

This work focuses on autonomous robots. Such a system can be defined as follows: “The robot is an autonomous system existing in the physical world that can detect the environment and take action to achieve the goal” [5].

The word “autonomous” expresses that such a system can make its own decisions without any intervention or control of a person in the real world. The perception of the environment of the system is ensured by various sensors providing

✉ Eva Volna
eva.volna@osu.cz

Adam Barton
adam.barton@osu.cz

Martin Kotyrba
martin.kotyrba@osu.cz

¹ Department of Informatics and Computers,
University of Ostrava, 30. dubna 22, 70103 Ostrava,
Czech Republic

data. For such a system to make a decision, there is a need for a cognitive element that assures the “thinking” of the robot.

Each robot should contain the following components [4].

- *Physical structures* existing in the real universe, i.e., the body of the robot, through which it can move, manipulate objects, etc.
- *Sensors*, i.e., an environmental sensing device providing information about the robot itself and the environment around it.
- *Effectors*, which perform actions in the environment, allow movements of the robot, interactions with the environment, etc.
- *Control system* defining a decision system, esp. autonomy of the system.

The control system provides a tool that creates autonomy of a robotic system, using available information, data for event planning, and execution of events. This is the “brain” of the robot, which may not be the only one (different sensors may be used by different control systems).

The control architecture provides organization of the control system and allows defining the behavior of the robot. It uses algorithms to solve problems. The tools for implementing the control architecture are programming languages, while the architecture itself only describes the frame.

An autonomous robot must be able to process sensor data, define the state of the outside world, and plan future actions to carry out the action in real time with limited resources. These operations are described by a control architecture defining the elements required to achieve the desired behavior, especially in the following areas [1]:

- *Programmability*—the robot should not be able to perform only one specific task, it should be able to achieve more tasks, and a set of tasks defined by one global goal.
- *Autonomy and adaptability*—the robot should be able to adapt the required tasks, actions, and behavior to meet the defined global goal.
- *Reactivity*—the robot should be able to decide at the right time and respond to events requiring an immediate or near immediate reaction.
- *Robustness*—a robot’s architecture should be able to handle unexpected problems in its functionality, task priorities, behavior, activities, conflict resolution, etc.
- *Scalability*—the implementation of new functionalities should not negatively affect the performance of the architecture; it should be easily expandable and understandable.

3 Data filtering using adaptive resonance theory

3.1 Big data

Big data have been one of the current and future research frontiers. Big data concern large-volume, complex, growing data sets with multiple, autonomous sources. With the fast development of networking, data storage, and the data collection capacity, big data are now rapidly expanding in all science and engineering domains, including physical, biological, and bio-medical sciences.

The most fundamental challenge for the big data applications is to explore the large volumes of data and extract useful information or knowledge for future [7]. In many situations, the knowledge extraction process has to be very efficient and close to real time, because storing all observed data is nearly infeasible. Four things are required to mine data effectively [9]: high-quality data, the “right” data, an adequate sample size, and the right tool. There are many tools available to a data mining practitioner. These include decision trees, various types of regression, and neural networks. In more practical terms, neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs. Using neural networks as a tool, we can harvest information from big data [6].

This article describes an effective way of data filtering through adaptive resonance theory, which is experimentally verified to control the behavior of an autonomous robot in an unknown environment.

3.2 Adaptive resonance theory

Adaptive Resonance Theory (ART1) is composed of two layers and a control neuron R , Fig. 1. Layer F_1 includes computational neurons X_i from the input layer F_1 (a) and neurons I_i from the interface layer F_1 (b). Neurons Y_i form layer F_2 . Synapses between F_1 and F_2 are oriented bottom-up, and synapses between F_2 and F_1 are oriented top-down. This approach enables to control the degree of similarity between weight values of synapse top-down and the input vector. Deciding is provided by neuron R (reset), which that separates states of neurons in F_2 three possibilities: active, inactive—acceptable for competition, and inactive—unacceptable for another competition [8].

Using a heuristic of self-organizing networks for preprocessing of the training set, we are able to find typical patterns and thus cover the entire space. We utilize ART1 to filter a training set, which works with a binary input. The present input is compared based on parameter vigilance $\rho \in [0, 1]$ with stored patterns in the network. If it differs significantly, this pattern is stored in the network as a new class. ART1 is able to create groups of patterns and the user has the oppor-

Fig. 1 ART1

tunity to influence the level of relative similarity of patterns stored in the same group.

To create a filtered training set, the basic algorithm ART1 [8] is modified as follows:

1. Initialization of synaptic weights, bottom-up b_{ij} , top-down t_{ij} , and parameter vigilance $\rho \in [0, 1]$:

$$b_{ij} = \frac{1}{(1+n)}, \quad t_{ij} = 1, \quad \rho \in [0, 1], \quad (1)$$

where n is the number of input neurons.

2. The submission of input $\mathbf{x} = [x_1, \dots, x_n]^T$ to input layer F_1 .
3. Activation of neurons in layer F_2 and calculation the response according to:

$$y_j = \sum_{i=1}^n b_{ij} x_i, \quad (2)$$

where y_j is the response of the j th neuron in F_2 and x_i is the i th element of input vector $\mathbf{x}$.

4. Selection of the neuron with the greatest response $y_{\max} = \max(y_j)$.
5. Calculation of similarity δ of input pattern with back weighting factor $t_{i_{\max}}$ regarding the neuron $y_{\max}$:

$$\delta = \frac{\sum t_{i_{\max}} x_i}{\sum x_i}. \quad (3)$$

6. If $\delta \geq \rho$, a resonance of input with expectations occurs and the candidate $y_{\max}$ is accepted, the algorithm proceeds to step 7. Otherwise, the candidate is rejected and its output is set to -1 and other suitable candidate will be chosen and the algorithm proceeds to step 3.
7. Weights' adaptation $b_{i_{\max}}$ and $t_{i_{\max}}$ of neuron $y_{\max}$ according to formulas (Weber Law):

$$b_{i_{\max}} = \frac{L t_{i_{\max}} x_i}{L - 1 + \sum t_{i_{\max}} x_i}, \quad (4)$$

$$t_{i_{\max}} = t_{i_{\max}} x_i, \quad (5)$$

where L is a constant (excelling unit), its value is usually $L = 2$.

8. Test for gradual addition of input vectors into a reduced training set (RTS):

- (a) If there is no vector in the given group of RTM yet, the currently presented vector $\mathbf{x}$ is stored as a typical representative of the group.
- (b) In the case that the group is not empty and also the currently submitted vector $\mathbf{x}$ carries a greater amount of information than the vector stored, as the representative of the group, vector $\mathbf{x}$ will replace it.

9. Checking the termination condition. If it is satisfied, the algorithm proceeds to step 11.
10. Suppression of neurons which were inhibited and the algorithm proceeds to step 2.
11. Stopping the algorithm.

The key step in the modified algorithm ART1 is Step 8, where the reduced training set (RTM) is compiled based on the classification of the network. The user chooses a rule setting of a vector with a higher content of carried information. In this work, we accept vectors containing more critical states, i.e., elements with value 1.

3.3 Impact of vigilance

ART1 requires parameter vigilance ρ , which defines the degree of similarity for assigning the pattern to the same class. This parameter is defined by the user. Figure 2 shows the effect of parameter ρ on pattern matching $p = [p_1, \dots, p_n]^T \in \{0, 1\}$ into classes c .

If a low value of ρ is set, it can lead to the classification of different patterns in the same class and the number of classes is small. On the contrary, a big value of ρ results in the formation of a large number of classes. In this work, ρ is set, so that its value is from the interval $[0.5; 0.7]$. This leads to the creation of a big number of groups or to an inadequate classification of the input pattern.

4 Related works

The work [11] presents the electronic design and motion planning of a robot based on decision-making regarding its straight motion and precise turn using artificial neural network (ANN). The ANN helps in learning of robot, so that it performs motion autonomously. The robot was designed to work in real time. The weights calculated were implemented in microcontroller. The learning algorithm was backpropagation with hyperbolic tangents activation-function, and the network was adapted through Easy NN-plus. The training set consisted of five inputs and one output determining the action: right, left, forward, and stop. It contained 14 designs expertly assembled, but the article does not mention values

Fig. 2 Influence of the vigilance parameter on the training set clustering

of other parameters. An autonomous robot has been tested in a maze environment to avoid obstacles. The performance has been tested to be excellent.

The work [12] presents the development and control of a disc-typed one-wheel mobile robot, called GYROBO. Two actuators are used for balancing control by virtue of gyro effect and one actuator for driving movements. To compensate for uncertainties in robot dynamics, a neural network was added to the non-model-based PD-controlled system. The reference compensation technique was used for the neural network controller to help GYROBO to improve balancing and tracking performances. Experimental studies of a selfbalancing task and a line tracking task are conducted to demonstrate the control performances of GYROBO. Experiments with a PD controller and a neural network showed that the generalization ability of the neural network can be a powerful tool. In this case, the robot was not controlled by the neural network itself.

The work [13] describes the concept of the navigation system for a mobile robot. The system is using a navigation algorithm based on self-learning neural network, necessary to form a movement plan for a robot. The algorithm is adapted and implemented to navigate real platform of a mobile robot equipped by two independent wheel drives, encoders and a set of short-range sonars. Navigation algorithm is placed into a PC, which is connected to mobile robot by wireless and wired links. Experiments have shown ability of collision-free navigation of mobile robot in real time. The designed robot scanned its surrounding with a collection of sensors from which it was able to represent this surrounding in a two-dimensional map that was also the input to a self-learning neural network. The robot needs a defined goal to move and then manages the optimal path in real time.

In this work [14], design of an intelligent autonomous vehicle is presented that can navigate in noisy and unknown environments without hitting the obstacles in its way. The vehicle is made intelligent with the help of two multilayer feed forward neural network controllers. Hurdle avoidance controller ensures collision-free motion of mobile robot,

while goal reaching controller helps the mobile robot in reaching the destination. Both these controllers are trained offline with the data obtained during experimental run of the robot, but the approximation of the activation function generated an error, and therefore, an increasing number of neurons could be worse outcome neuronal controller. The vehicle with the proposed controllers was tested in outdoor complex environments and is found to reach the set targets successfully.

This article [15] shows how to develop a robot that can learn using the backpropagation algorithm, a basic neural network, and implementing it on a Lego Roverbot. Using both the algorithm and Java, the Roverbot, a Lego robot vehicle, can learn some basic rules for moving forward, backward, left, and right. The aim of the project was also to examine the emergent behavior of the robot, and therefore, the training set was consisted with only four patterns. This approach allowed for the robot to achieve complex behavior, but at the cost of time that was needed for calculations. The program itself should always be uploaded to Lego brick after editing.

The project OpenWorm [16] mapped the connection of the nematode worm neurons and transferred his perception and resulting behavior to the robotic body. Although the simulation itself is not accurate due to software simplification, the neural network mimics the brain of the given worm without the need for learning or programming.

5 Control of autonomous robot behavior using neural networks

In the centralized control [4], the robot has information from sensors and the base of stored knowledge. Based on this information, it creates a plan of action. It scans potentially appropriate plans and it tries to predict. This is also the reason why this approach of “strategic thinking” requires a lot of memory and time.

Fig. 3 Assembled robot. **a** a view from above; **b** a side view

When searching for appropriate control architecture for an autonomous robot, it is necessary to compromise and consider the fact that [2].

- Planning requires computing performance.
- The reaction must be fast.
- Important information is needed for planning.
- Planning allows you to avoid making mistakes.
- The environment varies and the planning result becomes inaccurate in time.

The type of the used neural network (structure of units, topology, and learning algorithm) significantly depends on the solved problem. The most widespread artificial neural networks are multilayered networks that can handle complex data for non-trivial problems. The basic problem, not only in the management of autonomous robot, is finding the weights of the connections between neurons, either on the basis of the data (supervised learning) or finding dependencies in the data (unsupervised). When creating more complex structures, the disadvantage is the amount of interconnections between neurons and the non-existent process for creating such structures. An important aspect of neural network adaptation is the training set itself, which is expertly constructed on the basis of past experience of solving similar problems.

For experimental purposes, a mobile tracked robot with two motors and a set of sensors using Lego Mindstorms was built. As a suitable neural network, a feedforward neural network was chosen and its function, global objective, respectively (i.e., avoid obstacles in the space), was decomposed using a low-level description into individual patterns in a training set. One problem area is just to build such a training set for artificial neural networks. Next, we will deal

with the problem of creating a training set more specifically that is its automatic generation and filtration. The proposed model of the controlling neural network is then able to realize real output, such as the speed for each robot engine.

5.1 Model of an autonomous robot

The proposed model of the robot is assembled from Lego Mindstorm EV3 [3], see Fig. 3. The robot uses LeJOS operating system. The robot uses two large engines that propel a tracked chassis on which is placed a programmable brick. A sensory head with four infrared sensors is attached to the chassis. The sensors emit infrared signals and detect reflections from an object which is located in front of the encoder. The strength of the reflected signal indicates the distance from the object in the range from 0 to 100 units.

5.2 Controlling neural network

The type of the controlling neural network was chosen a three-layer feedforward network with a sigmoidal transfer function, which has been adapted by backpropagation. Here are the input neurons labeled X_i , hidden neurons Z_i , and output neurons Y_i . Hidden and output neurons have a threshold equaled to value “1”.

Considering that the robot model has four infrared sensors capable of detecting a distance of 0–100 cm, it is appropriate to divide this interval into several parts that could be characterized by words, for example, very close, close, medium-far, far, very far etc. to obtain a larger variability for robot's adaptation. We will, therefore, expand the network in width, based on the method of distribution of the detected interval of sensors into subintervals. For outputs, the use of the same

Fig. 4 Control neural network

procedure would mean that the complexity of network adaptation would grow as the number of speeds (outputs) and, depending on them, the number of hidden neurons. If we use a real-number vector, we are able to reduce the number of output neurons to four, as shown in Fig. 4, due to the use of two motors—two for each engine (forward and reverse).

For the transformation of the input signal to the output, we use the sigmoidal function $S_\lambda(x) : \mathbb{R} \rightarrow (0, 1)$ expressed by Eq. (6):

$$S_\lambda(x) = \frac{1}{1 + e^{-\lambda x}}, \quad (6)$$

where the constant λ is the slope parameter. We use the backpropagation algorithm to adapt the network. The aim of the adaptation is to find the minimal error E in the space of weights for all patterns $1, \dots, p$ defined by four output neurons as the mean square error (7):

$$E = \frac{1}{2} \sum_{i=1}^p \sum_{j=1}^4 (y_{ij} - o_{ij})^2, \quad (7)$$

where y_i is an actual output of the network after inserting the input x_i and o_i is an expected output from the training set after inserting the input x_i .

During adaptation, we minimize this error by adjusting the weight values w_i by an increment according to the following equation:

$$\Delta w_i(t) = -\alpha \frac{\partial E}{\partial w_i} + \mu \Delta w_i(t-1), \quad (8)$$

where $\alpha \in (0, 1]$ is the learning rate and the parameter $\mu \in [0, 1]$ reflects the increase in time $t-1$ and thus maintains the trend during the adaptation [8]. After adjustment (8), the increment of the weight between the input and the hidden layer is expressed as follows (9):

Fig. 5 Scanning area

$$\Delta w_j(t) = \alpha \sum_{i=1}^4 [(o_i - y_i) \lambda y_i (1 - y_i) v_i] \lambda z_j (1 - z_j) x_j + \mu \Delta w_j(t-1), \quad (9)$$

where v_i is the weight of the outgoing connection and z_j is the output of the hidden layer neuron. For the weight increment between the hidden and the output layer, Eq. (10) is obtained by adjusting (8):

$$\Delta w_i(t) = \alpha (o_i - y_i) \lambda y_i (1 - y_i) x_i + \mu \Delta w_i(t-1). \quad (10)$$

5.3 The training set filtering through ART1

The training set is automatically generated. If we consider that the model of the robot has four infrared sensors that are able to detect the distance from 0 to 100 cm, it is appropriate to decompose this input interval into several parts. We have decomposed the input interval into five parts. Therefore, the training set contains 20 inputs (each of the four sensors is associated with five terms) and four outputs. Then the number of possibilities (input patterns) that can occur is $2^{4 \times 5} = 2^{20}$. The output vector consists of real numbers, and thus, we have been able to reduce the necessary number of output neurons to four, due to the use of two motors—two neurons for each motor (forward and reverse).

We describe the principle of training set generating using an example. We consider four infrared sensors, each of which is able to detect the distance 0–100 cm. We split this distance in five equal parts, see Fig. 5. We express these parts using a set of terms: *vc*—very close (0–19 cm), *c*—close (20–39 cm), *mf*—medium-far (40–59 cm), *f*—far (60–79 cm), and *vf*—very far (80–100 cm).

Each sensor is associated with those five terms that represent obstacles in the sensor range for the given interval. Value “1” indicates the presence of obstacles in a given interval,

Table 1 Input patterns of reduced training set

Sensor 1					Sensor 2					Sensor 3					Sensor 4				
I_1	I_2	I_3	I_4	I_5	I_1	I_2	I_3	I_4	I_5	I_1	I_2	I_3	I_4	I_5	I_1	I_2	I_3	I_4	I_5
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 2 Number of groups ERTM depending on ρ

Vigilance ρ	Minimal value of the number of classes	Maximal value of the number of classes	The average number of groups	Mean number of groups
0	6	12	8.6	7
0.1	5	11	7.8	5
0.2	5	10	7.8	8
0.3	36	53	46.9	51
0.4	36	54	46.2	47
0.5	40	51	46.3	50
0.6	177	188	182.6	188
0.7	179	192	184.9	184
0.8	625	625	625.0	625
0.9	625	625	625.0	625
1	625	625	625.0	625

and value “0” represents a blank space. The value 100 cm is considered as an obstacle in the relevant interval.

We focus only on the part of the training set, which describes inputs. The given training set, based on the robot configuration and the distribution of the scanned area into five parts, comprises 1,048,576 patterns. The training set also contains patterns that admit a possibility of the existence of obstacles in several given intervals. Therefore, it is necessary to specify a group of inputs I_i belonging to a given sensor. In the case that the group comprises more than one value of „1“, such a pattern is not included in the training set. The reduced training set (RTS) comprises 1296 patterns, see Table 1.

After performing exclusion of groups with zero values, the reduced training set is reduced into 625 patterns (ERTS). Unfortunately, the resulting training set will increase depending on the number of sensors and on the distribution of the scanned areas. Therefore, we will process further filtration. The impact of vigilance setting for this training set is shown in Table 2. Characteristics of groups reflect ten filtering ERTS using ART for each vigilance setting.

The number of groups of the filtered training set is variable due to mixing of samples before their submission to ART1. This step influences the final training set significantly. From the view of the user, it was experimentally verified that the given training set of 625 patterns and $\rho > 0.5$ achieves the

Table 3 Training set for the adaptation of the neural network

Pattern	Evaluation of sensors with use of terms				Setting the engine speed for forward and reverse degrees (per second)			
	S4	S3	S2	S1	BF	BB	AF	AB
1	<i>vb</i>	<i>d</i>	<i>vd</i>	<i>b</i>	350	0	300	0
2	<i>d</i>	<i>s</i>	<i>s</i>	<i>vb</i>	300	0	350	0
3	<i>b</i>	<i>d</i>	<i>s</i>	<i>vd</i>	350	0	300	0
...	...	...	...	...	...	...	...	...
51	<i>vb</i>	<i>b</i>	<i>vb</i>	<i>d</i>	100	0	0	100
52	<i>vd</i>	<i>vd</i>	<i>vd</i>	<i>vd</i>	500	0	500	0

best results due to the consequent need for setting output values to these patterns manually. Therefore, $\rho = 0.5$ is used in this work. The resulting filtered training set contains 51 patterns.

For clarity, columns of associated intervals of individual sensors are combined into themselves, i.e., that each five binary input representing an obstacle in a given interval are replaced with only one linguistic value representing the distance for each sensor. For the adaptation of the neural network, the training set is expanded backwards. Table 3 describes outcomes of the training set, where S_1 , S_2 , S_3 , and S_4 indicates sensors in a given port; B_F and A_F indicates the

Fig. 6 Lifecycle of the training set

speed of motors for forward, B_B and A_B indicates the speed of motors for backward of ports A and B . Speed settings for individual run of both of engines were set experimentally with regard to limitation of maximum speed at 500 degrees/sec because of skidding belts on the smooth surface at higher speeds.

Pattern number 52, which defines the default behavior for moving forward, was added subsequently at the end of the training set. By adding the pattern, the robot is prevented from going a different direction than straight if no obstacle has been detected.

The life cycle of the training set is shown in Fig. 6. In this way, we are able to find typical patterns and thus cover the entire space of solutions. Filtration of the training set using ART1 adjusts the input vectors, so that these patterns are mixed pseudo-randomly.

6 Experimental verification

The aim of the experimental study is to verify the behavior of the robot controlled by artificial neural networks. This controlling neural network is adapted by the training set that is filtered through ART1. An experimental study runs in the following steps:

1. Define the division of the sensed sensor interval and determine the number of inputs for the complete training set.
2. Generate a training set.
3. Filtration of the training set using ART1.
4. Adjustment of the output vector elements of the training set to the desired values of engine speed with respect to forward or reverse, ranging from 0 to 600 degrees/sec.
5. Create the corresponding neural network topology.
6. Adapt this neural network with a filtered training set.
7. Connect EV3 robot to the application if not already done.
8. Set the experiment parameters and response of the network inputs to the given sensor values.
9. Run the experiment.

The topology of the controlling neural network had the following configuration: 20 input, 20 hidden, and 4 output neurons. Other parameters were set as follows: learning rate $\alpha = 0.85$, momentum factor $\mu = 0.9$, and the steepness parameter $\lambda = 1$. The termination condition for adaptation has been set as $E \leq 0.0001$, Eq. (6), due to normalization

of the training set's outputs. Adaptation lasted for 109 s and included 14393 cycles.

The model of the autonomous robot was placed in a space which consists of squares 15 cm^2 . Motor acceleration was set to 600 degrees/s for all experiments. Scanning of a space was carried out once per 500 ms. The following data are recorded for the purposes of evaluation of the experimental part.

- Number of primary collisions (direct collisions of the robot with an obstacle, i.e., running into obstacles),
- Number of secondary collisions (collision only some parts of the robot such as the sensor cables, etc.),
- Number of deadlocks,
- Duration of the experiment,
- Trajectory of the robot's movement.

Experimental outcomes are shown in Table 4. The robot has completed all objectives defined in individual experiments without deadlocks as well as without a primary collision with obstacles. In some cases, it is apparent that the signal from sensors reflects the way leading to an erroneous evaluation of the measured distance, see cases of corner collision (Fig. 7).

The last experiment represents free movement of the robot in the environment $195 \times 195 \text{ cm}$, which contains a number of obstacles placed, so that they form extra "rooms", in which the robot may become trapped or cause collision by its moving inside, Fig. 5. The experiment was conducted for 30 min, i.e., 3603 cycles of environment scanning. During the experiment, there were a total of ten secondary collisions, no primary collisions, and no deadlocks. The robot explored the whole experimental environment. There were the following collisions:

- A secondary collision marked as "1" occurred three times, always after the turn round in the space behind the barrier and coming out, when the robot touched bands on the corner of obstacles.
- A secondary collision marked as "2" occurred twice when the robot traveled around the obstacles left.
- A secondary collision marked as "3" occurred twice when the robot came out from the room and its sensors touched the barrier.
- A secondary collision marked as "4" occurred once, i.e., collision of a sensor with a corner when the robot came out from the environment behind the barrier.
- A secondary collision marked as "5" occurred twice, i.e., collision of belt when the robot traveled around the obstacles left and during rotation.

A video showing the progress of the experiment is placed on <https://www.youtube.com/watch?v=r-joFp3vh1k> under a title "EV3 Neural Network Controller".

Table 4 Experimental outcomes (secondary collisions are represented by points)

Movement of the robot in the environment	Number of primary collisions	Number of secondary collisions	Number of dead-locks	Average duration of the experiment (in second)
0	1	0	17.7
0	1	0	28
0	2	0	25
0	1	0	24
0	1	0	36.3
0	0	0	47.3
0	1	0	49.7
0	0	0	67

Fig. 7 Experimental environment for free movement of the robot

7 Conclusions

The whole training set included 1048576 patterns. We obtained 51 patterns after the training set filtration using ART1. Pattern 52 was added to them, which defines the default behavior for forward movement. Vigilance parameter ρ was set to 0.5. The control neural network was adapted by this reduced training set. Experimental outcomes are the following:

- The speed of each motor was adjusted appropriately, considering the situation in which the robot is found.
- The robot moved effectively in all environments, where precision of movement was not required too much.

- The ability of the robot to move backwards was observed during the experiment. Reversing was always implemented only for a short path, e.g., when the robot had little space to turn on the spot. This ability was not a part the training set.
- It has been observed that the robot is able to move too close to obstacles without running into them.

The use of self-organizing networks heuristic (ART 1) for preprocessing of the training set allows to find typical patterns, and thus cover the whole space. There is used, based on the results of previous work [3, 10], the ability of generalization of neural networks, to reduce the possibility of user error to achieve better results compared to a complete training set. It can be stated that ART1 is a good tool for data filtering.

Acknowledgements The research described here has been financially supported by the University of Ostrava Grant SGS07/PřF/2017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the sponsors.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Alami, R., Chatila, R., Fleury, S., Ghallab, M., Ingrand, F.: An architecture for autonomy. *Int. J. Robot. Res.* **17**(4), 315–337 (1998)
2. Bartoň, A.: Control of Autonomous Robot Using Neural Networks (in Czech), Master thesis, University of Ostrava, Czech Republic (2015)
3. Barton, A., Volna, E., Kotyrba, M.: Big data filtering through adaptive resonance theory. In: Asian Conference on Intelligent Information and Database Systems ACIIDS 2017, pp. 382–391. Springer, Cham (2017)
4. Dudek, G., Jenkin, M.: Computational Principles of Mobile Robotics. Cambridge University Press, Cambridge (2010)
5. Mataric, M.J.: The Robotics Primer. Mit Press, London (2007)
6. Mironovova, M., Bila, J.: Fast fourier transform for feature extraction and neural network for classification of electrocardiogram signals. In: 2015 Fourth International Conference on Future Generation Communication Technology (FGCT), pp. 1–6, IEEE (2015)
7. Rajaraman, A., Ullman, J.: Mining of Massive Datasets. Cambridge University Press, Cambridge (2011)
8. Rojas, R.: Neural Networks: A Systematic Introduction. Springer, Berlin (1996)
9. Singh, Y., Chauhan, A.S.: Neural networks in data mining. *J. Theor. Appl. Inf. Technol.* **5**(6), 36–42 (2009)
10. Volná, E., Kotyrba, M., Žáček, M., Bartoň, A.: Emergence of an autonomous robot's behavior. In: Proc. 29th European Conference on Modelling and Simulation, ECMS 2015, Albena, Bulgaria, pp. 462–468 (2015)
11. Tripathi, G.N., Rihani, V.: Motion planning of an autonomous mobile robot using artificial neural network. [arXiv:1207.4931](https://arxiv.org/abs/1207.4931) (2012) (preprint)
12. Kim, P.K., Jung, S.: Experimental studies of neural network control for one-wheel mobile robot. *J. Control Sci. Eng.* **2012**, 12 (2012) (Article ID 194397)
13. Markoski, B., Vukosavljev, S., Kukulj, D., Pletl, S.: Mobile robot control using self-learning neural network. In: 7th International Symposium on Intelligent Systems and Informatics, 2009. SISY'09, IEEE, pp. 45–48 (2009)
14. Farooq, U., Amar, M., Asad, M.U., Hanif, A., Saleh, S.O.: Design and implementation of neural network based controller for mobile robot navigation in unknown environments. *Int. J. Comput. Electr. Eng.* **6**(2), 83–89 (2014)
15. Reynoso, J.S.C.: A neural network for Java Lego robots. Learn to program intelligent Lego Mindstorms robots with Java. Javaworld. [Online] 16 April 2005. <https://www.javaworld.com/article/2071879/enterprise-java/a-neural-network-for-java-lego-robots.html> (2005)
16. Black, L.: A worm's mind. In a lego body. I Programmer. [Online] 16 Nov 2014. <http://www.i-programmer.info/news/105-artificial-intelligence/7985-a-worms-mind-in-a-lego-body.html>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.