Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 1629 – 1642

doi:10.3906/elk-1603-75

A neural network approach to navigation of a mobile robot and obstacle avoidance in dynamic and unknown environments

Farhad SHAMSFAKHR*, Bahram SADEGHIBIGHAM
Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences, Zanjan, Iran

Received: 08.03.2016 • **Accepted/Published Online:** 05.06.2016 • **Final Version:** 29.05.2017

Abstract: Mobile robot navigation and obstacle avoidance in dynamic and unknown environments is one of the most challenging problems in the field of robotics. Considering that a robot must be able to interact with the surrounding environment and respond to it in real time, and given the limited sensing range, inaccurate data, and noisy sensor readings, this problem becomes even more acute. In this paper, we attempt to develop a neural network approach equipped with statistical dimension reduction techniques to perform exact and fast robot navigation, as well as obstacle avoidance in such a manner. In order to increase the speed and precision of the network learning and reduce the noise, kernel principal component analysis is applied to the training patterns of the network. The proposed method uses two feedforward neural networks based on function approximation with a backpropagation learning algorithm. Two different data sets are used for training the networks. In order to visualize the robot environment, 180° laser range sensor (SICK) readings are employed. The method is tested on real-world data and experimental results are included to verify the effectiveness of the proposed method.

Key words: Obstacle avoidance, robot navigation, artificial neural networks, dimensionality reduction, function approximation

1. Introduction

Mobile robots are increasingly used in industry, agriculture, space exploration, military applications, etc. One of the most important research topics in the field of robotics is mobile robot navigation. Intelligent navigation of a mobile robot is autonomous behavior to generate a collision-free trajectory from an initial to a target position without human intervention. Sensor noise, stochastic actions, real-time response, online learning, limited training time, and situated representations are among the critical factors that make robot learning an extremely challenging problem [1]. Furthermore, the intelligent navigation of a mobile robot in dynamic environments, considering the kinematic constraints of the robot, is an especially difficult optimization problem. In order to generate a trajectory from the initial to the desired location in obstacle avoidance conditions, the robot must also possess perception, reasoning, and recognition characteristics, as well as the ability to learn and approximate any function with an appropriate generalization performance. For decades, various types of navigation methods have been developed to deal with the motion planning problem of mobile robots, such as the artificial potential field method [2,3], genetic algorithm (GA) method [4–6], particle swarm optimization (PSO) [7,8], and so on. However, these methods suffer from some inherent drawbacks. For example, the path planning

*Correspondence: farhad.shams@iasbs.ac.ir

approaches based on the artificial potential field method often get stuck in local minima, which renders the methods unsuitable for use in dynamic environments. GA methods belong to the class of random optimization processes, in which the convergence speed is low due to the high number of evolutionary process iterations to find an optimal solution. However, it should be noted that robot motion planning is not simply about avoiding collision and reaching the target point; stability in robot movements is a key property that is ignored in most related works. Stability is the result of a rational behavior, which means that the robot must not suddenly deviate from its path in any motion. It requires the steering angle of the robot to be of a reasonable value to the largest possible extent, so as not to change suddenly during navigation, which would cause the generated path to remain smooth and optimal from the start point to the target point. An appropriate use of machine learning and pattern recognition methods that results in a properly trained structure would be an excellent choice for developing such rational behavior. One of the most important machine learning approaches, which is widely and successfully used in a broad range of robotic problems, is the artificial neural network approach. Neural network robot controllers try to behave like humans and other living creatures in a navigation task. As all living creatures choose a steering angle with a continuous value in a navigation task [9], the motion planning problem can be regarded as a function approximation problem. This interesting feature motivated us to propose a method by this means. Hence, the first and main contribution of this paper is to provide an exact and fast method for intelligent control of a mobile robot in static and dynamic environments, using a neural network with a backpropagation learning algorithm based on function approximation. There are various types of neural networks [10,11]: Hopfield neural networks, recurrent neural networks, feedforward neural networks, and RBF neural networks, as well as different types of optimization techniques (such as the Levenberg–Marquardt and conjugate gradient methods), which can be used in many robotic applications. Different neural network approaches have been proposed to control mobile robots [12–18]. A biologically inspired neural network approach has been used in mobile robot path planning [19]. However, kinematic constraints of the mobile robot were not considered in this work, and the proposed method simplifies the problem of motion planning to a high extent. A pattern recognition method with a backpropagation learning algorithm for mobile robot navigation, based on computer vision, was proposed [20], where selecting the motion direction of the robot is limited to four strategies, resulting in poor navigation ability. A recurrent neural network for the control of a nonholonomic mobile robot that can learn the dynamic model of the robot was proposed [21], where the learning algorithm of the controller is computationally expensive, causing a slow convergence. A reinforcement learning method uses a neural network with Q-learning to navigate an industrial vehicle in unknown environments by avoiding collisions [22]. This approach converges faster and gives more accurate results in comparison to the traditional Q-learning method. Approximating Q-learning with a neural network has provided a better way of storing Q values compared to a look-up table. A bioinspired neural network controller for autonomous control of a vision-based land vehicle, equipped with video cameras, was learned [23].

The data obtained from the robot's sensors predominantly have high dimensions, and working with high-dimensional data sets requires applying an appropriate dimensionality reduction technique to them. On the other hand, the learning process can be affected by another major factor, namely noise. Here the term "noise" refers to a phenomenon that may be caused by inaccurate data in the training data set or by the electrical noise of the system (white noise). Training data sets are generated by collecting data from different robot observations and selecting a suitable value for the steering angle of the robot in each observation. However, because the value selection is performed by a human, the selected values are often not very precise. This is an inevitable phenomenon that leads to inaccurate data. Hence, the second contribution of this paper is to reduce

the noise and dimensions of data using an appropriate statistical dimension reduction technique. One of the most effective methods to reduce the dimensionality of a large data set is kernel principal component analysis (PCA). In this work, kernel PCA is used to compute the most meaningful features of the data set. Previously, PCA and other dimensionality reduction techniques were successfully applied in a similar manner to different fields of robotics. An approach is presented to reduce the dimension of range-based sensor data and obtain an accurate model of the environment using principal components of range data in a robot localization problem [24]. A dimension-reduction method using PCA reduced the dimensions of sonar sensor data through linear projections that preserved the multiple structures of the data [25]. A generalized neural network approach to mobile robot navigation was proposed, which can be used for various types of range sensors and robot platforms [26]. Even though dimensionality reduction techniques were applied to this work, the dimensions of the inputs were too large, which led to relatively low performance. Moreover, the kinematic constraints were not considered in that work. The remainder of this paper is organized as follows: in Section 2, we define the problem statement, including the mobile robot configuration and kinematic model. The motion planning algorithm consists of 3 parts: visualization, dimensionality reduction, and neural network structure, as described in Section 3. In Section 4, the simulated results are presented, and, finally, our conclusions are given in Section 5.

2. Problem statement

2.1. Mobile robot configuration

Figure 1a shows the configuration of a mobile robot in a two-dimensional Cartesian space. The robot consists of two independently driven wheels in the rear with a servo motor, which allows for precise position and velocity control for motion planning; one unpowered free wheel for support in the front; a SICK laser scanner (LMS 200-30106) for detecting the obstacles; and encoders for measuring the wheel velocity. The initial position of the robot is denoted by (x_0, y_0). The angle between the positive x-axis and the line passing through the center of the robot is called the target angle (θ_t).

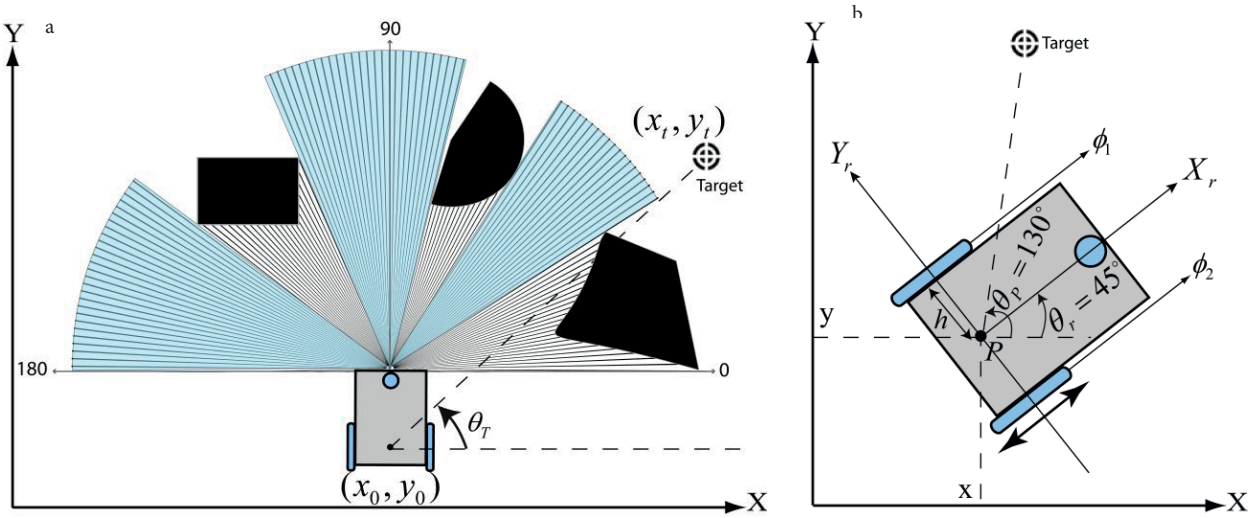

Figure 1. (a) Configuration of the mobile robot, (b) the kinematic model.

2.2. Kinematic model

The kinematic model of the mobile robot is depicted in Figure 1b. In order to specify the position of the robot, we select a point P on the rigid body of the robot as the position reference point. This point is located at the center of the line that passes through the middle of the wheels. Both wheels have the same diameter r, and h is the distance between P and the wheel. Let $\phi\cdot_1$ and $\phi\cdot_2$ be the angular velocity of the two driving wheels, where ϕ_1 and ϕ_2 are the positions of the left and right wheels, respectively. If we assume that there is no slip between the robot wheels and the surface, the linear velocity and angular velocity can be obtained as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2h} & -\frac{r}{2h} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix} \tag{1}$$

The linear velocity of each wheel must not exceed the maximum speed.

$$\left|\dot{\phi}_1\right| \leq \dot{\phi}_{1,\max} \tag{2}$$

$$\left|\dot{\phi}_2\right| \leq \dot{\phi}_{2,\max} \tag{3}$$

θ_p is a target angle with respect to the current position of the mobile robot. The position of the robot can be described in an inertial Cartesian frame {O, x, y} by $\xi = [\mathrm{x\ y\ } \theta_r]$. Let (x, y, θ_r) be the position and orientation of the mobile robot regarding the inertial frame, where (x, y) are the coordinates of the reference point P, and θ_r is the robot angle that indicates the orientation of the mobile robot. For simplicity, let us ignore the center-of-gravity (COG) shifts. We denote the distance between the position of the COG on the rigid body of the robot and P by d. The kinematics of a differential-drive mobile robot, described in the inertial frame, can be specified by the following Jacobian matrix of transformation [27]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -d\cos\theta \\ \sin(\theta) & d\sin\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4}$$

Let us assume that the position of the robot at any time depends only on its odometry measurement and can be estimated by integrating the robot movements during a time interval (Δ_t).

3. Motion planning algorithm

3.1. Perception

The first section of any mobile robot navigation method is to acquire information about the robot's environment. Various types of sensors have been used for this purpose. In this study, the laser range sensor SICK LMS 200-30106 was used to obtain a description of the robot's surroundings. Although these sensors suffer from poor calibration, they are very fast (seventy-five 180° scans per second); hence, they are much better than ultrasonic sensors in cases of speed of wave propagation, angular resolution, and distance traveled [27]. The laser range finder scans the environment around the robot with a range from 10 cm to 80 m. It provides a sequence of 181

values at every sensor scan, indicating the distance between the robot and the obstacles around it. Figure 2a shows a simulation of the SICK sensor, performed in the MATLAB environment. The intersection points of the laser beams with the obstacles are marked with small circles in the figure.

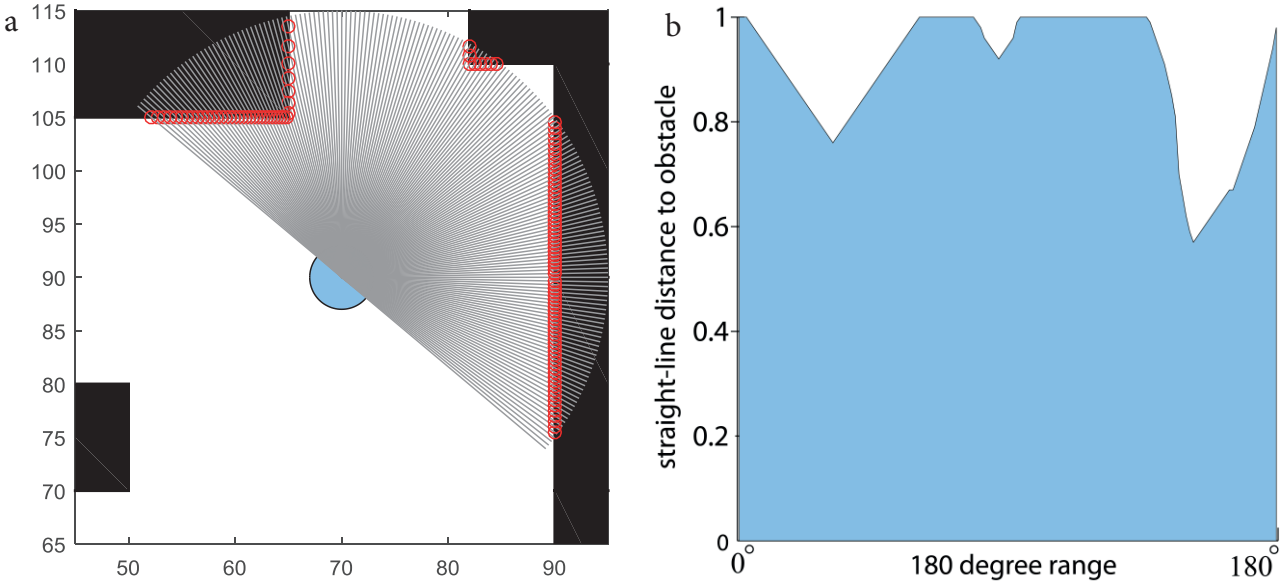

Figure 2. (a) Simulation of SICK in MATLAB, (b) the initial training pattern.

3.2. Visualization

Information obtained from the raw sensory data should be interpreted as a pattern for training the network. The visualization method in this study differs from the methods of several previous related works, in which the data received from the laser sensors were visualized onto bitmap images. Such a visualization method is not technically sound and will result in a much higher dimensional data set, which leads to slow convergence and low performance of the network. Therefore, in our proposed method, only the length and the angle of 181 laser beams, which are reachable by laser, are modeled as the training pattern. Training pattern matrix Z is an R $\times$ α binary matrix, where α is the number of laser beams of the sensor and R is an indicator of the sensor range, which can be scaled with respect to a desired value in order to determine the size of P. Hence, in our experiment, after visualization every pattern will consist of 181 distances between the robot and the surrounding obstacles. One of the patterns used in this study is depicted in Figure 2b.

3.3. Dimensionality reduction

The initial training pattern shown in Figure 2b is a complicated pattern. The more complicated patterns are, the better understood robot environments become. However, complicated patterns cause high-dimension data sets; additionally, applying them as the network input will lead to a slow convergence and learning process due to the large number of features of the training patterns. Hence, regarding the contributions of this study—accuracy, speed, and noise reduction—it is necessary to apply an appropriate dimensionality reduction method to the data set and extract the most meaningful principal features of the data that cover a reasonable percentage of the explained variance of the data set. In this work, we applied kernel PCA to the data set. PCA is a useful statistical technique that computes the most meaningful features to reexpress data in a lower dimension. If the

data set is noisy, then we can anticipate that the new features will remove the noise and represent the hidden structure. Hence, it can be said that applying PCA to the data set does not remove the noise completely, but it certainly reduces it. The standard steps of PCA can be summarized as follows:

Step 1: Subtract the mean (mean centering of the covariance matrix).

Step 2: Calculate the covariance matrix C with the following equation:

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X^T} \tag{5}$$

where X is the data set.

Step 3: Calculate the eigenvectors ($_P$) and eigenvalues of the covariance matrix.

Step 4: Choose PCs from ($_P$) and transform data on PCs with the following equation:

$$\mathbf{P_{k\times m}} = \mathbf{X_{m\times n}} = \mathbf{Y_{k\times n}} \tag{6}$$

Choose k according to the proportion of explained variance: $\frac{\sum_{i=1}^{k}\Lambda_i}{\sum_{i=1}^{m}\Lambda_i} >$ Threshold

Standard PCA only performs a linear mapping of the data, which does not perfectly represent the hidden structure of the data set in complicated high-dimensional feature spaces. It is anticipated that kernel PCA will allow us to efficiently compute the principal components of the data set when it has more complicated structures [28].

3.3.1. Kernel PCA

Kernel PCA is to perform PCA in a kernel Hilbert space by a nonlinear mapping. The term "kernel" refers to a nonlinear mapping function ϕ that maps the original N-dimensional features space into a higher dimensional space. This mapping can be written as $x \rightarrow \varphi(x)$, which is called the kernel function. Therefore, given two points x_i and x_j, the inner product (kernel) is:

$$K(x_i, x_j) = \varphi(x_j)\varphi(x_i)^T \tag{7}$$

Instead of performing increasingly expensive computations of the dot product, a kernel can be used to map observations from an original space x into a higher dimensional space ϕ. This is called the kernel trick [29]. There are various types of kernels, e.g., the polynomial kernel, Gaussian kernel, sigmoid kernel, or Laplacian kernel. However, there is no rule for choosing the right kernel; instead, doing a grid search over the values of kernel parameters and performing cross-validation for each choice can be helpful. In this study, we use the Euclidean distance kernel [30]:

$$\|\varphi(x) - \varphi(y)\|^2 = K(x,x) + K(y,y) - 2K(x,y) \tag{8}$$

As illustrated in Table 1, the Euclidean distance kernel function has covered a higher proportion of variance explained than other types of kernels. Using this kernel function has dramatically improved the performance of both neural networks used in this work (FFNN A and FFNN B). Additionally, we used this method to reduce the dimension of the initial training pattern, shown in Figure 2b, from 993 features to 30 features.

Table 1. The results of applying PCA and different types of kernels to the data set.

Type of kernel	Proportion of variance explained by 30 PCs	Best validation performance obtained for FFNN A	Best validation performance obtained for FFNN B
Standard PCA (no kernel)	88.412%	34.7523	19.3030
Polynomial kernel	90.213%	29.3794	17.8632
Gaussian kernel	91.011%	26.6285	14.7108
Laplace kernel	89.423%	32.8661	15.2316
Euclidean distances kernel	96.001%	14.9539	9.7456

The standard steps of kernel PCA can be summarized as follows:

Step 1: Construction of kernel matrix using Eq. (8).

Step 2: Centering kernel matrix.

Step 3: Normalizing kernel matrix with the following equation:

$$\mathbf{C} = \mathbf{XX^T} - \mathbf{XX^TJ} - \mathbf{JXX^T} + \mathbf{JXX^TJ} \tag{9}$$

where $\mathbf{J} = \frac{\mathbf{1}}{\mathbf{n}}\mathbf{1_{n\times n}}$.

Step 4: Compute the PCs by solving the eigenvalue problem.

There has always been a trade-off between performance and convergence speed in training neural networks when selecting the number of features. Decreasing the number of features achieves better performance but reduces accuracy in detecting obstacles. Increasing the number of features leads to weak convergence and lower performance of the network. By applying kernel PCA to the data set, the best possible trade-off between the two factors is achieved. It can be seen from the scree plot shown in Figure 3 that by using the Euclidean distances kernel, the data set can be described perfectly by only 30 features.

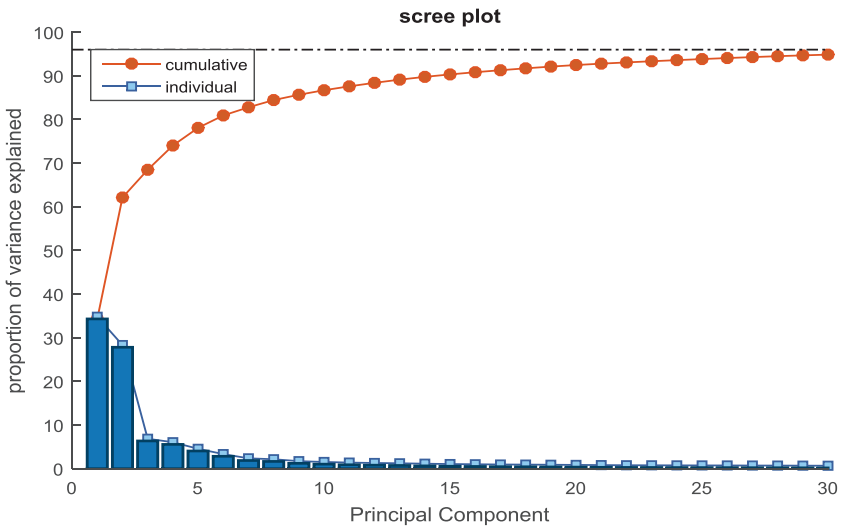

Figure 3. Optimum number of PCs.

3.4. Structure of the feed-forward neural network

In order to achieve intelligent navigation for a mobile robot in a static/dynamic environment, the multilayer feedforward neural network is used to approximate an appropriate trajectory between the initial position and

target position. To achieve better performance, we used two neural networks (FFNN A and FFNN B) with different training data sets. The two neural networks were trained by being presented with 3000 patterns for each network. These patterns represented typical scenarios that a robot may encounter in real environments. An appropriate architecture is selected for the feedforward neural networks. FFNN A has an input layer consisting of 31 nodes in the input layer, 17 nodes in the first hidden layer, 9 nodes in the second hidden layer, and an output layer that indicates the corresponding steering angle. FFNN B has an input layer consisting of 31 nodes in the input layer, 6 nodes in the first hidden layer, 3 nodes in the second hidden layer, and the output layer. The numbers of hidden neurons are empirically found to be convergent of error to a minimum threshold error. The numbers of hidden layers are also found empirically, as with one hidden layer it is difficult to learn the network within a specified error limit. The network architecture for the two neural networks is depicted in Figure 4a.

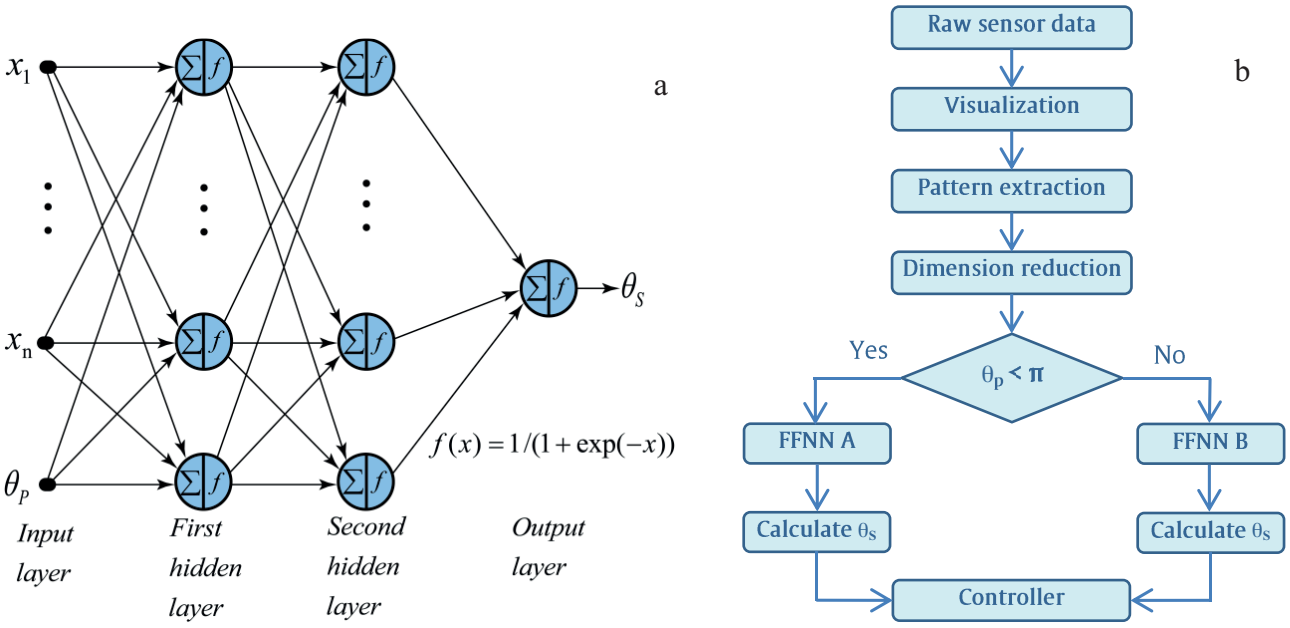

Figure 4. (a) The neural network architecture, (b) the system flowchart.

There are two path-planning strategies in our approach: normal action and wall-following action. While θ_p is less than π (the target is in front of the robot), the robot takes normal action. However, when it encounters a wide or U-shaped obstacle while moving forward (i.e. when the target is located behind the robot), it is required to keep only one direction (the nearest side of the obstacle) and follow the wall in order to get past the obstacle and avoid getting stuck in an infinite loop. Because these two actions are quite different behaviors, learning their behavioral function by only one neural network is quite challenging and significantly reduces the performance of the network. In order to gain better results, we used two different neural networks with two different data sets to approximate two different functions. While θ_p is less than π, FFNN A will be employed for approximate θ_s (steering angle); otherwise, FFNN B is used. There is an appropriate steering angle for each pattern. The purpose of the paper is to find the most precise value for θ_s so that the robot does not collide with obstacles in its path during the navigation. It is assumed that the exact position of the target (x_t, y_t) is given at any time during the navigation. Due to the different positions of the obstacles and

the targets in different situations, the target angle θ_T has a crucial role to play. Our proposed algorithm is illustrated in Figure 4b.

3.4.1. An appropriate algorithm for training the feedforward neural network

In this study, we use the backpropagation algorithm, which employs the conjugate-gradient technique to train the network. Through experiments using the conjugate-gradient technique, we succeeded in acquiring a much better performance. As can be seen from Figure 4a, each neuron j in layer n receives signals from the previous layer of neurons and sends signals to the next layer. Therefore, the output of neurons in layer n can be calculated from the outputs of neurons in layer $n-1$ with the following equation:

$$Out_i^{(n)} = f(\sum_i Out_i^{(n-1)} W_{ij}^{(n)}) \tag{10}$$

where f is a logistic function and W is the weight matrix consisting of connections between the nodes of the network. The training of a neural network refers to a method that repeatedly modifies the weight of the connections between the nodes in t epochs so that the network output has maximum similarity to the target of the corresponding samples in the training set. This similarity is measured by the mean squared error function E:

$$E(W_{ij}^{(n)}) = \frac{1}{2} \sum_p \sum_k (y_k^p - Out_{k(x_i^p)}^{(N)})^2 \tag{11}$$

Since our method is a function approximation method, then Out_3^1 will be the only network output and will refer to **Actual** $\theta_{\mathbf{S}}$. The target of a training sample y refers to **Expected** $\theta_{\mathbf{S}}$. Therefore, in each epoch, the backpropagation learning algorithm attempts to change the weight in a way that the difference between the expected $\theta_{\mathbf{S}}$ and the real $\theta_{\mathbf{S}}$ is minimized.

The network weights at epoch t are $\mathrm{W}_{ij}(\mathrm{t})$. Then, by amount $\Delta \mathrm{W}_{ij}(\mathrm{t})$, we compute weights at epoch t + 1:

$$W_{hl}(t+1) = W_{hl}(t) + \Delta W_{hl}(t) \tag{12}$$

Hence, the weight rule updates recursively by Eqs. (10), (12), and the following equations:

$$\Delta w_{hl}^{(n)} = -\eta \frac{\partial E(w_{ij}^{(n)})}{\partial w_{hl}^{(n)}} = \eta \sum_P (y_l - Out_l^{(n)}).Out_l^{(n)}.(1 - Out_l^{(n)})(t).Out_h^{(n-1)}(t) + \alpha \Delta w_{hl}^{(n)}(t-1) \tag{13}$$

Here η and α are the learning rate and momentum, respectively. Through experimentation, it is found that η = 0.006 and α = 0.0021 give better convergence.

4. Simulation results

It would be ideal to test the performance of the proposed method on real-world data. The data set is the domestic rooms (DR) data set, which contains various robot observations of several domestic environments in the form of SICK sensor readings, which are collected from 24 different homes and consist of houses and flats from 4 different cities. The data set was originally used in [31]. All data sets were collected when the Pioneer P3-DX robot, equipped with laser range-finder Hokuyo URG with a 5.6-m sensor range and 240° field of view, was guided through each environment. The complete implementation of our algorithm was carried out in a MATLAB environment. By putting the robot and the goal in various environments and testing the

robot motion behavior in different situations, we demonstrated the good performance of the proposed method. Performance plots for FFNN A and FFNN B are depicted in Figures 5a and 5b, respectively. As can be seen from the figures, performance plots for training, validation, and test sets show similar characteristics, which means that the networks are very well trained and have a good generalization performance.

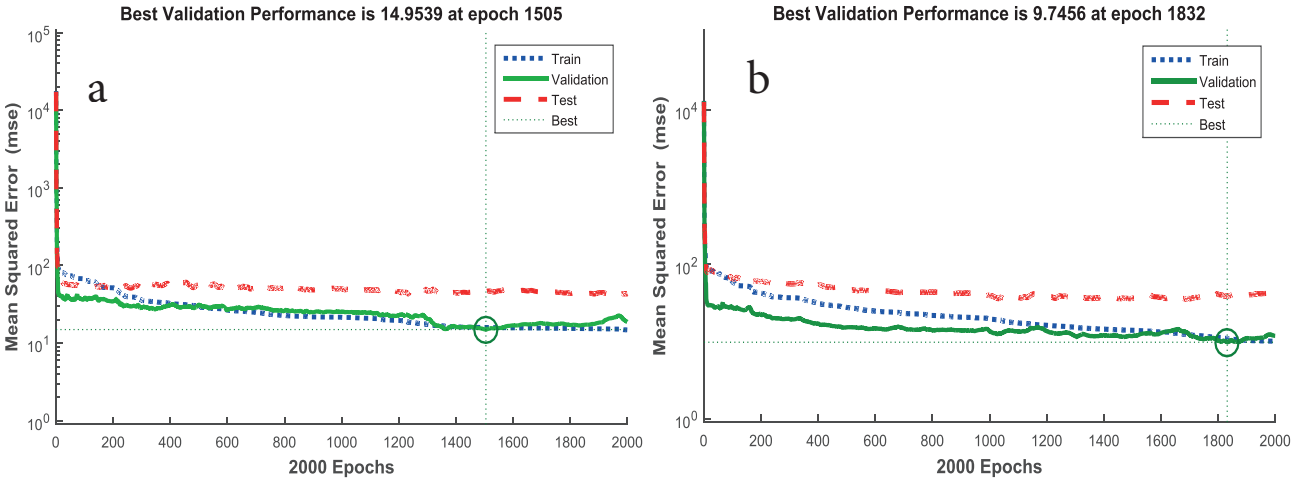

Figure 5. (a) The performance plot for FFNN A, (b) the performance plot for FFNN B.

As we discussed earlier, the focus of the study is mainly on fast and robust navigation. Thus, it would be better to use a long-range distance sensor that is faster than the Hokuyo URG. In order to make the data set applicable for SICK, it is necessary to perform data preprocessing before going into the navigation simulation of the robot. After preprocessing, the sensor field of view is limited to 180° and the sensor range equals 20 m. The simulation results of the motion planning, based on obstacle avoidance for a mobile robot equipped with a SICK sensor, are illustrated in Figures 6 and 7. The dimensions of the map are 200 × 200 m. The obstacles can be static or dynamic, and the robot has no prior knowledge of the environment. The traces in the figures show robot movements. Table 2 demonstrates the total number of time steps and the total elapsed time in robot navigations for different scenarios in Figures 6 and 7. The control time step is 0.5 s, and the robot moves with a maximum speed of 0.5 m/s towards the target. As can be seen from the speed time graphs, the robot has a minimum moving speed of 0.0012 m/s and a maximum moving speed of 0.49033 m/s during the navigations. Because of the smoothness of the generated trajectories, the average speed for each scenario is larger than 0.4 m/s. Taking the most accurate values of steering angles of the robot at any time increases the average moving speed during the navigation. The robot does not decelerate its velocity even when it encounters a wide or U-shaped obstacle while moving towards it, or when it moves along corners. Moving at high speed along the corner of an object is not an easy task and is virtually impossible, even if the robot is very well trained. As can be seen from the following simulation result, achieving the right steering angle leads to the generation of a smooth and safe path from the start point to the target position.

Table 2. Average moving speeds and total elapsed time for all four simulations in Figures 6 and 7.

Simulation	Figure 6a	Figure 6b	Figure 7a	Figure 7b
Average speed (m/s)	0.4288	0.4119	0.4006	0.4290
Total time steps	278	280	345	354
Time (s)	124.2	129.34	160.4	158.02

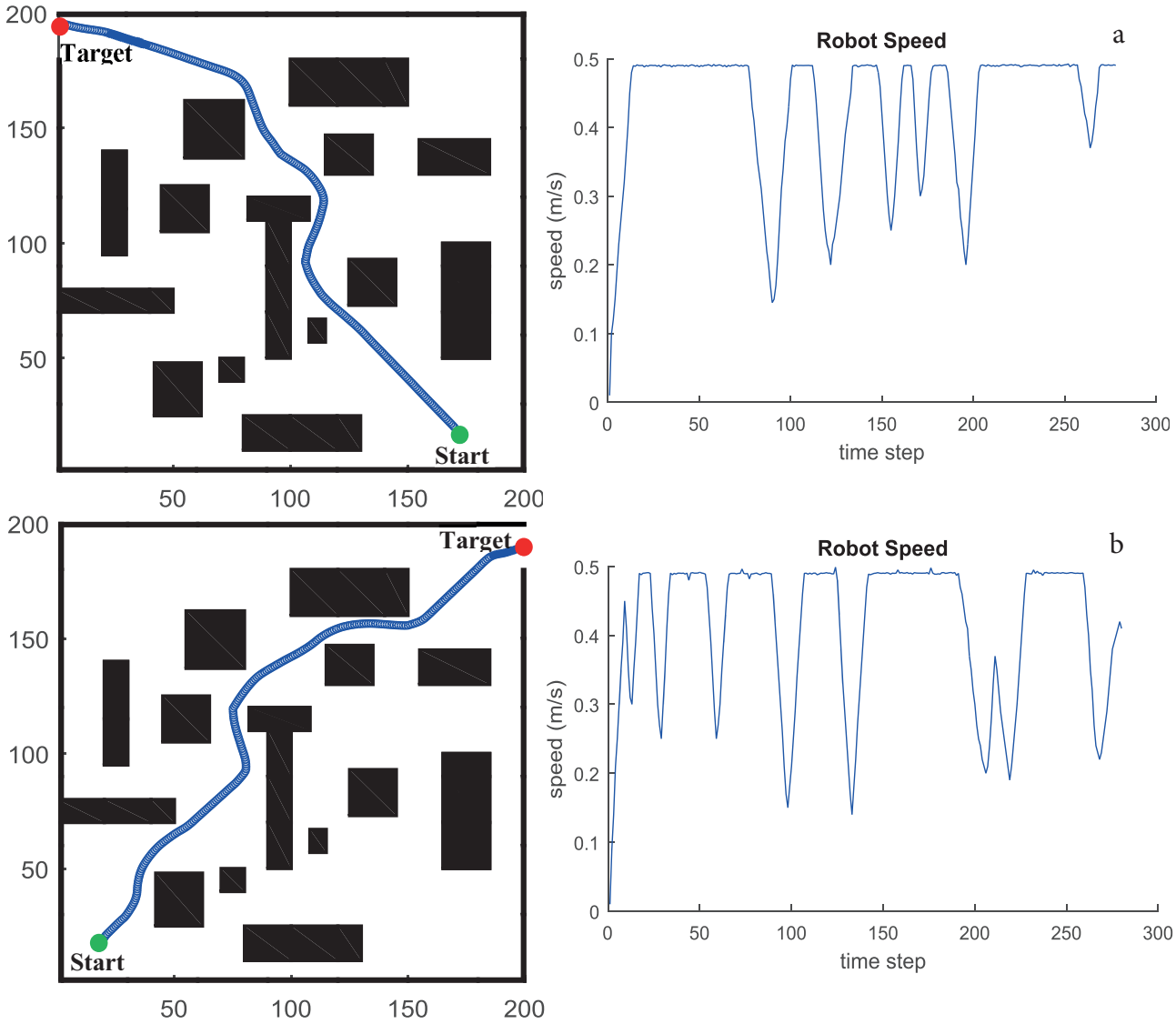

Figure 6. Velocity and trajectory of the Pioneer P3-DX robot for two different simulations in a static environment.

5. Conclusion

In this paper, a feedforward neural network, based on function approximation that uses a backpropagation algorithm with a conjugate-gradient method, was utilized for the intelligent navigation of a mobile robot and for obstacle avoidance in dynamic and unknown environments. Using kernel PCA equipped with the Euclidean distances kernel function, we have effectively reduced the dimension of the data set to 30. Our proposed method has been successfully applied to real-world data. Due to the low number of input features, and using an appropriate algorithm for training the feedforward neural network, a good generalization performance for training, validation, and test data sets was achieved. The average validation performance of both neural networks is 12.34975, which proves the efficiency of the proposed method. Our algorithm, considering the kinematic constraint of the robot, developed a rational behavior for a mobile robot. Navigation time experiments show that achieving highly accurate values for the steering angle leads the robot to have stable movements, even

when it encounters corners, and, in this way, minimizes both the length of the path and the time of navigation in the presence of an unknown environment.

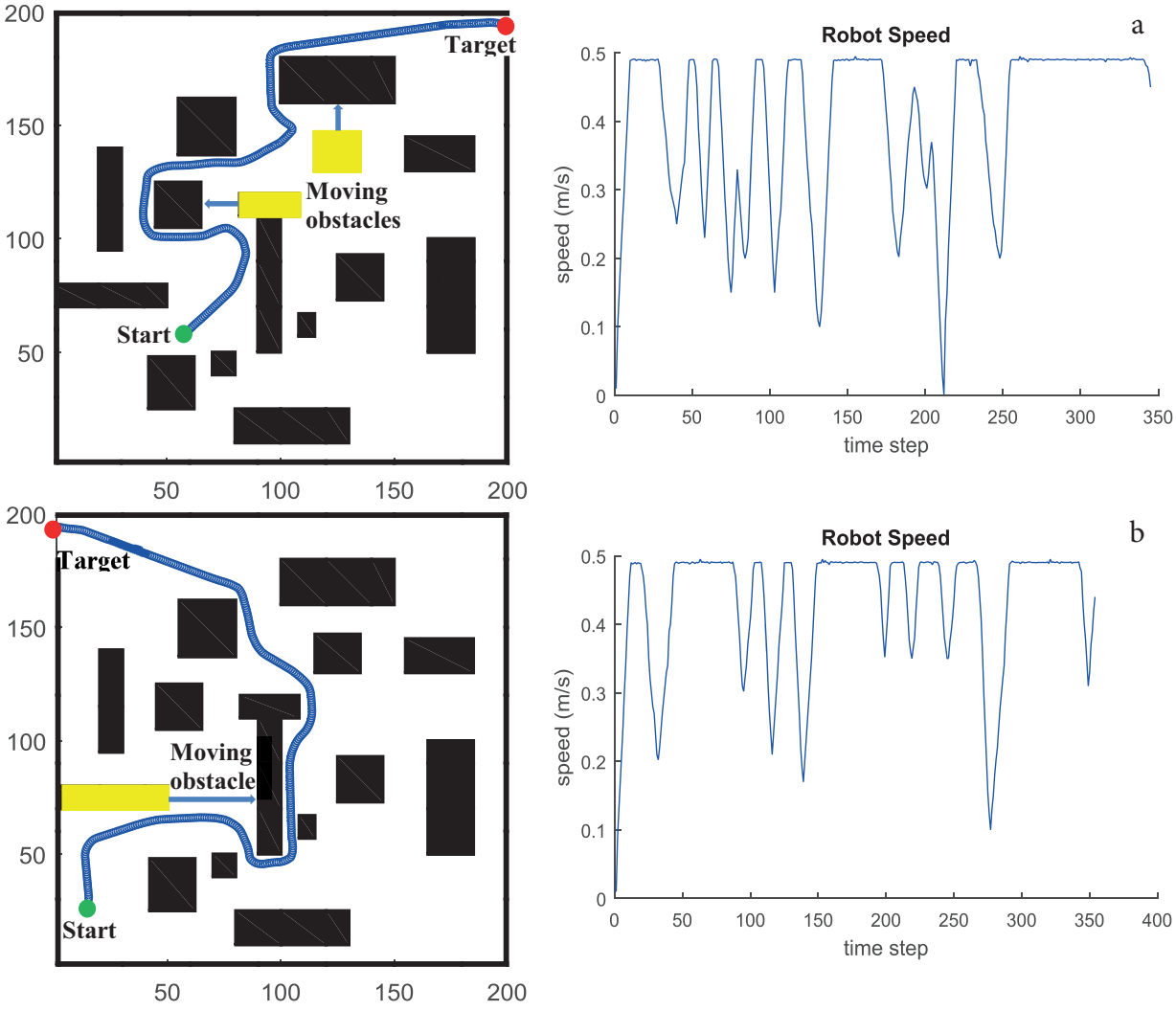

Figure 7. Velocity and trajectory of the Pioneer P3-DX robot for two different simulations in a dynamic environment.

References

[1] Mahadevan S. Machine learning for robots: a comparison of different paradigms. In: IEEE/RSJ 1996 International Conference on Intelligent Robots and Systems; 8 November 1996; Osaka, Japan. New York, NY, USA: IEEE. pp. 1-14.

[2] Tang L, Dian S, Gu G, Zhou K, Wang S, Feng X. A novel potential field method for obstacle avoidance and path planning of mobile robot. In: IEEE 2010 International Conference on Computer Science and Information Technology; 9–11 July 2010; Chengdu, China. New York, NY, USA: IEEE. pp. 633-637.

[3] Wang Y, Chirikjian GS. A new potential field method for robot path planning. In: IEEE 2000 International Conference on Robotics and Automation; 24–28 April 2000; San Francisco, CA, USA. New York, NY, USA: IEEE. pp. 977-982.

[4] Alander JT. On robot navigation using a genetic algorithm. In: Albrecht RF, Reeves C, Steele NC, editors. Artificial Neural Nets and Genetic Algorithms. Innsbruck, Austria: Springer, 1993. pp. 471-478.

[5] Manikas W, Ashenayi K, Wainwright R. Genetic algorithms for autonomous robot navigation. Instrum Meas Mag 2007; 10: 26-31.

[6] Geisler T, Manikas TW. Autonomous robot navigation system using a novel value encoded genetic algorithm. In: IEEE 2002 Midwest Symposium on Circuits and Systems; 4–7 August 2002; Tulsa, OK, USA. New York, NY, USA: IEEE. pp. III-45.

[7] Zhang Y, Gong DW, Zhang JH. Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing 2013; 103: 172-185.

[8] Qin YQ, Sun DB, Li N, Ma Q. Path planning for mobile robot based on particle swarm optimization. Robot 2004; 26: 222-225.

[9] Breed MD, Moore J. Animal Behavior. 2nd ed. San Diego, CA, USA: Academic Press, 2016.

[10] Haykin SS. Neural Networks and Learning Machines. 3rd ed. Upper Saddle River, NJ, USA: Pearson Education, 2009.

[11] Hagan MT, Demuth HB, Beale MH, De Jesús O. Neural Network Design. Boston, MA, USA: PWS, 1996.

[12] Wang Y, Wang S, Tan R, Jiang Y. Motion control of a wheeled mobile robot using digital acceleration control method. Int J Innov Comput I 2013; 9: 387-396.

[13] Parhi DR, Singh MK. Real-time navigational control of mobile robots using an artificial neural network. J Mech Eng Sci 2009; 7: 1713-1725.

[14] Dahm P, Bruckhoff C, Joublin F. A neural field approach for robot motion control. In: IEEE 1998 International Conference on Systems, Man, and Cybernetics; 11–14 October 1998; San Diego, CA, USA. New York, NY, USA: IEEE. pp. 3460-3465.

[15] Zou AM, Hou ZG, Fu SY, Tan M. Neural networks for mobile robot navigation: a survey. In: Wang J, Yi Z, Zurada JM, Lu BL, Hujun Y, editors. Advances in Neural Networks. Chengdu, China: Springer, 2006. pp. 1218-1226.

[16] Chi KH, Lee MF. Obstacle avoidance in mobile robot using Neural Network. In: IEEE 2011 International Conference on Consumer Electronics, Communications and Networks; 16–18 April 2011; Xianning, China. New York, NY, USA: IEEE. pp. 5082-5085.

[17] Ortega JG, Camacho EF. Mobile robot navigation in a partially structured static environment, using neural predictive control. Control Eng Pract 1996; 12: 1669-1679.

[18] Jung IK, Hong KB, Hong SK, Hong SC. Path planning of mobile robot using neural network. In: IEEE 1999 International Symposium on Industrial Electronics; 12–16 July 1999; Bled, Slovenia. New York, NY, USA: IEEE. pp. 979-983.

[19] Ni J, Li X, Fan X, Shen J. A dynamic risk level based bioinspired neural network approach for robot path planning. In: IEEE 2014 World Automation Congress; 3–7 August 2014; Waikoloa, HI, USA. New York, NY, USA: IEEE. pp. 829-833.

[20] Quiñonez Y, Ramirez M, Lizarraga C, Tostado I, Bekios J. Autonomous robot navigation based on pattern recognition techniques and artificial neural networks. In: Ferrández Vicente JM, Álvarez-Sánchez JR, de la Paz López F, Toledo-Moreo FJ, Adeli H, editors. Bioinspired Computation in Artificial Systems. Elche, Spain: Springer, 2015. pp. 320-329.

[21] Fierro R, Lewis FL. Control of a nonholonomic mobile robot using neural networks. IEEE T Neural Networ 1998; 7: 589-600.

[22] Chen Xi, El Kamel A. A reinforcement learning method of obstacle avoidance for industrial mobile vehicles in unknown environments using neural network. In: Qi E, Jiang SH, Runliang D, editors. Proceedings of the 21st International Conference on Industrial Engineering and Engineering Management. Amsterdam, the Netherlands: Atlantis Press, 2015. pp. 671-675.

[23] Pomerleau DA. Neural network based autonomous navigation. In: Charles E, editor. Vision and Navigation: The Carnegie Mellon Navlab. Pittsburg, PA, USA: Springer, 1990. pp. 83-93.

[24] Crowley JL, Wallner F, Schiele B. Position estimation using principal components of range data. In: IEEE 1998 International Conference on Robotics and Automation; 16–20 May 1998; Leuven, Belgium. New York, NY, USA: IEEE. pp. 3121-3128.

[25] Vlassis N, Motomura Y, Kröse B. Supervised dimension reduction of intrinsically low-dimensional data. Neural Comput 2002; 1: 191-215.

[26] Dezfoulian SH, Wu D, Ahmad IS. A generalized neural network approach to mobile robot navigation and obstacle avoidance. In: Lee S, Cho H, Yoon KJ, Lee J, editors. Intelligent Autonomous Systems. Jeju Island, Korea: Springer, 2013. pp. 25-42.

[27] Siegwart R, Nourbakhsh IR, Scaramuzza D. Introduction to Autonomous Mobile Robots. Cambridge, MA, USA: MIT Press, 2011.

[28] Schölkopf B, Smola A, Müller KR. Kernel principal component analysis. In: Gerstner W, Germond A, Hasler M, Nicoud JD, editors. Artificial Neural Networks. Lausanne, Switzerland: Springer, 1997. pp. 583-588.

[29] Schölkopf B, Smola A, Müller KR. Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 1998; 7: 1299-1319.

[30] Schiilkopf B. The kernel trick for distances. In: Todd KL, Thomas GD, Volker T, editors. Advances in Neural Information Processing Systems 13. Cambridge, MA, USA: MIT Press, 2000. pp. 301-307.

[31] Uršič P, Kristan M, Skočaj D, Leonardis A. Room classification using a hierarchical representation of space. In: IEEE/RSJ 2012 International Conference on Intelligent Robots and Systems; 7–12 October 2012; Vilamoura, Portugal. New York, NY, USA: IEEE. pp. 1371-1378.