A Bayesian Neural Network-based Obstacle Avoidance Algorithm for an Educational Autonomous Mobile Robot Platform

Anh Hoang
Hanoi University of Science and Technology, Vietnam
anh.hoang@hust.edu.vn

Son Thanh Nguyen
Hanoi University of Science and Technology, Vietnam
son.nguyenthanh@hust.edu.vn (corresponding author)

Tuan Van Pham
Vinh University of Technology Education, Vietnam
tuanvp.bk@gmail.com

Tu Minh Pham
Hanoi University of Science and Technology, Vietnam
tu.phamminh@hust.edu.vn

Linh Viet Trieu
Hanoi University of Science and Technology, Vietnam
linh.trieuviet@hust.edu.vn

Trung Thanh Cao
Hanoi University of Science and Technology, Vietnam
trung.caothanh@hust.edu.vn

Received: 22 August 2023 | Revised: 21 September 2023 | Accepted: 29 September 2023

 DOI: https://doi.org/10.48084/etasr.6304

ABSTRACT

Autonomous mobile robots belong to automatically controlled objects that are designed and produced for various demands. This study aimed to develop an inexpensive platform of autonomous mobile robots that can be used for educational and research purposes in technical universities. The robot was built based on popular ultrasonic sensors to detect obstacles and a Raspberry Pi 4, which is a Linux-embedded computer. An effective obstacle avoidance algorithm for the robot was developed using a Bayesian neural network for classification. Training a Bayesian neural network does not require a validation dataset separate from the available data. In addition, the Bayesian approach can effectively handle the uncertainty of the system and result in the best generation for the network when inferring the unseen data. Training data are generated using robot-to-obstacle distances and the corresponding navigation modes. The commands to control the left and right motors of the robot are generated by the pretrained Bayesian neural network for three modes of navigation, forward, left, and right. Finally, this system can be useful as it can be conveniently integrated with advanced robot control algorithms.

Keywords-autonomous mobile robots; obstacle avoidance; Bayesian neural network

I. INTRODUCTION

Autonomous mobile robots have been widely used in various industrial applications. In the last decades, there has been a lot of attention on motion planning for autonomous mobile robots. The motion planning problem can be divided into path planning and trajectory planning [1-2]. Path planning is related to the generation of obstacle-free paths with

geometric characteristics of obstacles and the kinematic constraints of the robot. Meanwhile, trajectory planning takes into account the robot's dynamics and moving obstacles (obstacles not known in advance). Basic navigation of mobile robots can be divided into the following tasks: (1) generating a model of the environment in the form of a map, (2) computing a collision-free path from the starting position to the target, and (3) following the generated trajectory. The crucial issue of a mobile robot is executing collision-free motions within its environment without any human intervention. Therefore, mobile robots must be equipped with distance sensors such as ultrasonic distance sensors, laser range sensors, and infrared sensors to detect obstacles and make decisions about how to navigate their environment smoothly.

The ability of mobile robots to avoid obstacles can be implemented by various algorithms. Fuzzy logic has become an appropriate solution for low-cost mobile robots that do not require complicated navigation [3-9]. In fuzzy logic, the linguistic variables of the inputs are combined to create the linguistic variables of the outputs. As a result, the fuzzy logic system needs to measure and analyze all inputs before inferring the outputs. Artificial neural networks can also be used in mobile robots to avoid obstacles [10-13]. Sonar sensors are used to obtain data for neural network training, generate training patterns corresponding to possible environment scenarios, and then train offline neural networks.

Multi-Layer Perceptron (MLP) neural networks are commonly used in various applications. The generalization of the trained neural network, or how well it can predict new cases, is the key challenge when developing an MLP neural network. Indeed, an insufficient network complexity can result in underfitting phenomena, in which significant data may be ignored. In contrast, excessive network complexity can cause overfitting, in which noisy data are also fitted. The complexity of an MLP neural network depends on the magnitudes of the weights and biases. In addition, the network size, depending on the number of hidden nodes, can also be used as a useful factor to measure network complexity. For a long period, the Bayesian approach has been used to improve the generalizability of feed-forward neural networks in the case of finite and noisy data [14-16]. In this process, network regularization is needed to prevent the network parameters (weights and biases) from becoming large, because large values of weights and biases usually cause the overfitting phenomenon to downgrade the performance of the network after being trained. When training feed-forward networks, the values of the regularization parameters can be easily found by using the Bayesian technique. Feed-forward neural networks that can be trained using the Bayesian technique are also called Bayesian Neural Networks (BNNs).

As BNNs can be effective for various classification problems, this study aimed to provide a procedure for deploying a classification BNN to navigate the direction of an autonomous mobile robot with the following aspects:

- Explore the evidence framework to allow all available data to be used to train BNNs. This work is needed when the collection of relatively large data is expensive or time-consuming.
- As the traditional gradient descent algorithm with a fixed step size and search direction to search local minima usually causes an excessively large network training time, this study focused on the quasi-Newton optimization technique, which is an advanced optimization training algorithm to automatically adjust the step size and search direction to optimal values.
- As finding the optimal network architecture is often a time-consuming task, this study also used the Bayesian model comparison concept to choose the best network architecture. This was carried out by evaluating the evidence for different candidate BNNs with varying numbers of hidden nodes. The network architecture that can give the highest value of the evidence will be chosen for the final use.

II. CLASSIFICATION BAYESIAN NEURAL NETWORK

A. Feed-Forward Two-Layer Propagation

Feed-forward two-layer neural networks have been widely used in engineering applications. This type of neural network can be used for non-linear regression and classification problems. The feed-forward propagation of the network can be expressed as follows:

$$y = f_1(w_1 x + b_1) = f_1(a_1) \quad (1)$$

where x is the input vector, w_1 is the weight and bias vector on the connection from the input nodes to the hidden nodes, b_1 is the bias of the hidden nodes, a_1 is the activation vector of the hidden nodes, and f_1 is the activation function of the hidden nodes, and y is the vector of output values of the hidden nodes. The output values of the output nodes are given by:

$$z = f_2\left(y\overline{w}_2 + \overline{b}_2\right) = f_2(\overline{a}_2) \quad (2)$$

Similarly, $\overline{w}_2$ is the vector of weights and biases on the connection from the hidden nodes to the output nodes, $\overline{b}_2$ is the vector of the biases of the output nodes, $\overline{a}_2$ is the vector of the activation of the output nodes, and f_2 is the activation function of the output nodes. In the network, w_1, b_1, $\overline{w}_2$ and $\overline{b}_2$ are four vectors corresponding to four groups of weights and biases.

For both regression and classification problems, the activation function of the hidden nodes, f_1, is the "hyperbolic tangent function" (tanh):

$$y = f_1(a_1) = tanh(a_1) = \frac{exp(a_1) - exp(-a_1)}{exp(a_1) + exp(-a_1)} \quad (3)$$

For multi-class classification problems, the activation function of the output nodes is the "softmax" function:

$$z_j = f_2\left(\overline{a}_j\right) = \frac{exp(\overline{a}_j)}{\sum_{i=1}^{c} exp(\overline{a}_i)} \quad (4)$$

where $\overline{a}_j$ is the activation of the j-th output, $c$is the number of outputs, and z_j is the output value of the j-th output node. For multi-class classification problems, the error data function has the following form:

$$E_D(w) = -\sum_{n=1}^{N}\sum_{k=1}^{c} t_k^n \, ln\left(z_k^n(w)\right) \quad (5)$$

where E_D is called the "entropy" function, z_k^n is the output of the k-th output node corresponding to the n-th training pattern, t_k^n is the target corresponding to the k-th output node and the k-th training pattern, and $w = [w_1, b_1, \overline{w}_2, \overline{b}_2]$ is the vector of weights and biases in four groups.

B. Network Regularization

To avoid the overfitting phenomenon for the feed-forward neural network when it is trained on noisy and finite training data, a weight penalty function should be used by adding it to the data function to obtain a cost function as follows:

$$S(w) = E_D(w) + \sum_{g=1}^{G} \xi_g \|w_g\|^2 \quad (6)$$

where w_g is the vector of weights and biases in the g-th group, and ξ_g represents the regularization parameters or hyperparameters that can be determined using the Bayesian approach. Equation (6) is based on the principle of penalizing the large weights and biases that can cause overfitting to the network when the training data are very noisy and finite.

C. Bayesian Inference

The purpose of conventional neural network training is to minimize a data error function to obtain specific weight and bias values. Meanwhile, the use of the Bayesian approach in neural network training considers the distribution of weights and biases. Bayesian inference is quite easy to understand. Using the Bayes rule, the posterior distribution of weights and biases, given the vector of hyperparameters $\psi = [\xi_1, \xi_2, \ldots, \xi_g]$ and the training data D, can be expressed as:

$$p(w|D,\psi) = \frac{p(D|w,\psi)p(w|\psi)}{p(D|\psi)} \quad (7)$$

where $p(w,\psi)$ is the prior distribution of weights and biases, $p(D|w,\psi)$ is the dataset likelihood, $p(D|\psi)$ is called the evidence, and $p(w|D,\psi)$ is the posterior distribution of weights and biases. By assuming that the prior distribution is a zero-mean Gaussian distribution:

$$p(w|\psi) = \frac{1}{Z_W(\psi)} exp\left(-\sum_{g=1}^{G} \xi_g E_{W_g}\right) \quad (8)$$

$$Z_W(\psi) = \prod_{g=1}^{G} \left(\frac{2\pi}{\xi_g}\right)^{\frac{W_g}{2}} \quad (9)$$

where $Z_W(\psi)$ is the normalization constant and W_g is the number of weights and biases in group g. The dataset likelihood has the following form:

$$p(D|w,\psi) = exp(-E_D) \quad (10)$$

It was also assumed that at the most probable vector of weights and biases, w_{MP}, the cost function can be approximated as a quadratic function by a Taylor series expansion as follows:

$$S(w) = S(w_{MP}) + \frac{1}{2}(w - w_{MP})^T A (w - w_{MP}) \quad (11)$$

where $S(w_{MP})$ is the cost function evaluated at w_{MP}, and A is the Hessian matrix of the cost function, which is computed as follows:

$$A = \nabla\nabla S(w_{MP}) = H + \sum_{g=1}^{G} \xi_g I_g \quad (12)$$

where $H = \nabla\nabla E_D(w_{MP})$ is the Hessian matrix of the data error function at w_{MP}, and I_g is the diagonal matrix having ones along the diagonal that picks weights and biases in the g-th group. It can be also assumed that the posterior distribution of weights and biases is a Gaussian distribution as follows:

$$p(w|D,\psi) = \frac{1}{Z_S(\psi)} exp\big(-S(w)\big) \quad (13)$$

where $Z_S(\psi)$ is the normal constant for the Gaussian approximation, given by:

$$Z_S(\psi) \approx exp\big(S(w_{MP})\big)\,(2\pi)^{W/2}(det\ A)^{-1/2} \quad (14)$$

Using Bayes' theorem, the posterior distribution of the hyperparameters can be computed as follows:

$$p(\psi|D) = \frac{p(D|\psi)p(\psi)}{p(D)} \quad (15)$$

As $p(D)$ does not depend on the network parameters, (15) becomes:

$$p(\psi|D) \equiv p(D|\psi)p(\psi) \quad (16)$$

where $p(\psi)$ is the prior distribution of the hyperparameters. This distribution can be assumed to be uniform and can be ignored subsequently. Therefore, the values of the hyperparameters will be determined according to the maximum of $p(D|\psi)$. Re-arranging (7) gives:

$$p(D|\psi) = \frac{p(D|w,\psi)p(w|\psi)}{p(w|D,\psi)} \quad (17)$$

In (17), all terms of the right-hand side can be determined using (8), (10) and (13), therefore (17) yields:

$$p(D|\psi) = \frac{Z_S(\psi)}{Z_W(\psi)} \quad (18)$$

Substituting (9) and (14) into (18) gives:

$$p(D|\psi) =$$

$$exp\big(S(w_{MP})\big)\,(det\ A)^{-1/2} \prod_{g=1}^{G} \big(\xi_g\big)^{W_g/2} \quad (19)$$

Taking the derivative of $ln\,p\,(D|\psi)$ for ξ_g results in:

$$\frac{\partial}{\partial \xi_g} ln\,p\,(D|\psi) = \frac{W_g}{2\xi_g} - E_{W_g} - \frac{1}{2} tr(A^{-1}) I_g \quad (20)$$

By letting this derivative be zero, the following relationship can be obtained:

$$2\xi_g E_{W_g} = W_g - \xi_g tr(A^{-1}) I_g \quad (21)$$

The right-hand side of (21) is equal to γ_g defined as:

$$\gamma_g = W_g - \xi_g tr(A^{-1}) I_g \quad (22)$$

γ_g is the number of "well-determined" parameters in group g. Substituting (22) into (21) gives:

$$\xi_g = \frac{\gamma_g}{2E_{W_g}} \quad (23)$$

For the most probable vector of weights and biases, ξ_g and γ_g can be used to obtain the log of the evidence of the network as follows [15-16]:

$$ln\, E\, v(X_i) = -S(w_{MP}) + \sum_{g=1}^{G} \frac{W_g}{2} ln\, \xi_g - \frac{1}{2} ln(det\, A) + ln\, M\,! + M\, ln\, 2 + \sum_{g=1}^{G} \left[\frac{1}{2} ln\left(\frac{4\pi}{\gamma_g^{MP}} \right) - G\, ln(ln\, \Omega) \right] \quad (24)$$

where M is the number of hidden nodes and G is a minor constant for all networks. Therefore, this factor does not affect the relative comparison of different candidate neural networks. The network with the highest evidence can be the network having the optimal number of hidden nodes.

D. Quasi-Newton Method

The neural network training requires the use of an effective optimization method to minimize the cost function over the space of weights and biases. The Newton method is a modification of the conjugate gradient method to obtain fast convergence. This method is based on Newton's formula as follows:

$$w_{m+1} = w_m - A^{-1}(g_{m+1} - g_m) \quad (25)$$

where w_m is the weight and bias vector at the m-th iterative step, and w_{m+1} is the weight and bias vector at the m+1-th iterative step. g_m is the gradient of the cost function at the m-th iterative step and g_{m+1} is the gradient of the cost function at the m+1-th iterative step. $A^{-1} = F$ is the inverse Hessian matrix of the cost function and can be approximated using the quasi-Newton method such as the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) method as follows [13-14]:

$$F_{m+1} = F_m + \frac{pp^T}{p^T v} - \frac{(F_m v)v^T F_m}{v^T F_m v} + (v^T F_m v)uu^T \quad (26)$$

where p, v, and u are defined as:

$$p = w_{m+1} - w_m \quad (27)$$

$$v = g_{m+1} - g_m \quad (28)$$

$$u = \frac{p}{p^T v} - \frac{F_m v}{v^T F_m v} \quad (29)$$

III. EDUCATIONAL AUTONOMOUS MOBILE ROBOT AND OBSTACLE AVOIDANCE

This section describes an autonomous mobile robot with obstacle avoidance capability using a classification BNN. As small two-wheeled robots are easy to get into robotics, the educational robot in this study was formed by a popular two-wheeled mobile robot chassis, as shown in Figure 1, consisting of two driving wheels attached to each side of the robot chassis and driven by two DC motors. To detect obstacles, the robot was equipped with three HC-SR05 ultrasonic sensors, as shown in Figure 2, for the following reasons:

- Ultrasonic sensors can sense all material types.
- Ultrasonic sensors are not affected by atmospheric dust, rain, snow, etc.
- Ultrasonic sensors can work under any adverse conditions.
- Ultrasonic sensors have higher sensing distances compared to inductive/capacitive proximity sensor types.
- Ultrasonic sensors provide good readings when sensing large-sized objects with hard surfaces.

A Raspberry Pi 4 embedded computer was used as the main controller. Raspberry Pi 4 is the latest series of the Raspberry Pi, including more robust performance compared to the previous series. The use of Raspberry Pi allows for the convenient data acquisition from the ultrasonic sensors corresponding to different navigation modes of the robot. In addition, Raspberry Pi can facilitate the quick and easy development of control algorithms using the Python programming language and explore various Python libraries of machine learning, such as Scikit-learn, Keras, and TensorFlow. An Uninterruptible Power Supply (UPS) HAT for Raspberry Pi is used to power it. Figures 3 and 4 show the Raspberry Pi 4 and the whole robot. The robot and a PC/laptop can communicate through a Wi-Fi router and RealVNC software. WinSCP software was used to transfer data from the PC to the robot and vice versa.

A. Data Generation

To develop a BNN-based obstacle avoidance algorithm for the robot, a dataset is needed to train and test it. The data consists of information from ultrasonic sensor readings and the corresponding navigation modes, which can also be known as targets. This step can be carried out by human dexterity in manually navigating the robot, but this takes significant time, and sometimes it is not effective. Instead, an algorithm for navigating the robot without obstacle collisions in some terrains is proposed to obtain the robot navigation data.

Fig. 1. Two-wheeled mobile robot chassis.

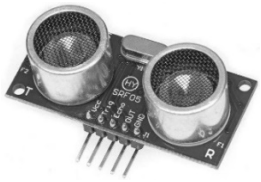


Fig. 2. The HC-SR05 ultrasonic sensor.

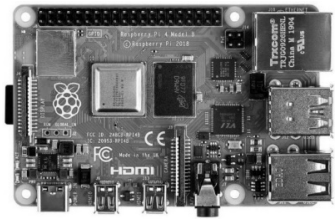


Fig. 3. Raspberry Pi 4 embedded computer.

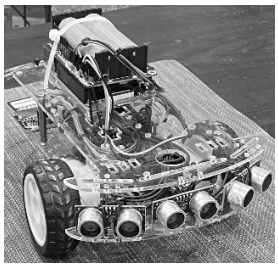


Fig. 4. The whole mobile robot.

Three ultrasonic sensors were used to measure three distances from the robot to obstacles, according to three directions: left, central, and right. A threshold was then defined to categorize each distance variable into two states: far obstacle and near obstacle. Experimentally, the suitable value of the threshold was set to 30 mm. This means that the robot is near an obstacle if the measured distance is less than or equal to 30 mm, and if the measured distance is greater than 30 mm, the robot is far from the obstacle. There are only three navigation modes: forward, left, and right movements. For left and right movements, the robot is needed to move backward before it can turn left or right.

Table I shows the navigation modes corresponding to three states of the three distances. The time required for each navigation mode is experimentally determined to avoid too fast or too slow response during the robot's navigation around the terrain.

TABLE I. NAVIGATION MODES ACCORDING TO DISTANCE STATES

Distance state			**Navigation mode**
Left distance	**Central distance**	**Right distance**	
Far obstacle	Far obstacle	Far obstacle	Forward movement
Near obstacle	Far obstacle	Far obstacle	Backward and then right movement
Far obstacle	Near obstacle	Far obstacle	Backward and then left movement
Near obstacle	Near obstacle	Far obstacle	Backward and then right movement
Far obstacle	Far obstacle	Near obstacle	Backward and then left movement
Near obstacle	Far obstacle	Near obstacle	Forward movement
Far obstacle	Near obstacle	Near obstacle	Backward and then left movement
Near obstacle	Near obstacle	Near obstacle	Backward and then right movement

Table II is used to generate the training data for the classification BNN, while Table III shows several patterns, including three distance sensor values and corresponding navigation target modes.

TABLE II. NAVIGATION MODE AND CORRESPONDING TARGET VECTORS

Navigation mode	**Target vectors**		
Forward movement	1	0	0
Backward movement and then left movement	0	1	0
Backward movement and then right movement	0	0	1

TABLE III. SEVERAL INPUT PATTERNS AND CORRESPONDING TARGET VECTORS

Distances from the robot to obstacles (mm)			**Target vectors**		
Left	**Central**	**Right**			
65.32	20.09	19.89	0	1	0
117.12	63.04	47.49	1	0	0
20.73	24.69	124.07	0	0	1
81.65	43.43	29.93	0	1	0
88.82	61.87	47.58	1	0	0

Based on Table I, the robot was programmed to move in an environment with some obstacles. Then the data from the three ultrasonic sensors corresponding to the robot navigation modes were recorded as shown in Table IV.

TABLE IV. PATTERNS OF TRAINING DATA

Navigation mode	**Number of patterns**
Forward movement	315
Left movement	40
Right movement	44
Total	399

B. Network Training Procedure

The network training was performed using the Netlab software [16], which is an open-source MATLAB neural network toolbox with the following features:

- Netlab was developed using pure MATLAB programming language and does not depend on other MATLAB toolboxes.
- It was developed with the automated neural network regularization using the Bayesian technique.
- Based on Netlab, users can rank and compare different neural networks with different numbers of hidden nodes to finally choose the most suitable number of hidden nodes for their applications.
- Instead of using a fixed value of the learning rate, advanced neural network training algorithms in Netlab can be used to obtain true convergences when minimizing highly nonlinear cost functions.

The raw data need to be normalized into a range from 0 to 1 to be used in the network training process. In particular, the measured distance sensor values were divided by 1500. The structure of the BNN consists of three input nodes, an appropriate number of hidden nodes, and three output nodes. The network training procedure includes the following steps:

- Step 1: Values of the network weights and biases were randomly initialized using a zero-mean Gaussian distribution.
- Step 2: Initial values of the four hyperparameters constraining magnitudes of weights and biases in the four groups were also assumed to be small.
- Step 3: Quasi-Newton optimization was used to minimize the cost function. This step can be also known as the weight and bias update process.

- Step 4: When the cost function reaches a local minimum, the hyperparameters can be re-estimated using the following equations:

$$A = H + \sum_{g=1}^{G} \xi_g^{old} I_g \quad (30)$$

$$\gamma_g^{old} = W_g - \xi_g^{old} tr(A^{-1}) I_g \quad (31)$$

$$\xi_g^{new} = \frac{\gamma_g^{old}}{2E_{W_g}} \quad (32)$$

Direct calculation of the Hessian matrix of the data error function H can be alternated by an efficient method that does not require time-consuming computations [17]. Steps 3 and 4 were repeated until the hyperparameter values did not change significantly in subsequent iterations.

C. Optimization of Network Structure

To find the most suitable network structure corresponding to the optimal number of hidden nodes, seven networks with different numbers of hidden nodes, ranging from 2 to 8, were trained and evaluated. As shown in Figure 5, the network with four hidden nodes provided the highest log evidence. Therefore, the optimal number of hidden nodes is four.

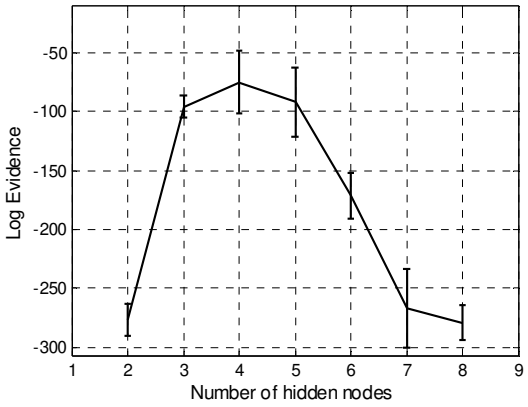


Fig. 5. Log evidence versus the number of hidden nodes.

Table V shows the confusion matrix of the network performance with the training data. The overall accuracy was 97.24%. After training the network, the robot operated on a new terrain. During this period, the information from the ultrasonic sensors and the corresponding navigation modes were recorded to form the test data, as shown in Table VI. Table VII shows the confusion matrix of the network performance with the test data. The overall accuracy of the navigation mode classification was also relatively high, with a successful rate of 96.29%.

TABLE V. CONFUSION MATRIX ON THE TRAINING DATA

		Predicted classification		
		Forward movement	**Left movement**	**Right movement**
Actual classification	**Forward movement**	309	4	2
	Left movement	3	36	1
	Right movement	0	1	43
Overall accuracy (%)		97.24		

TABLE VI. PATTERNS OF THE TEST DATA

Navigation mode	**Number of patterns**
Forward movement	161
Left movement	15
Right movement	40
Total	216

TABLE VII. CONFUSION MATRIX ON THE TEST DATA

		Predicted classification		
		Forward movement	**Left movement**	**Right movement**
Actual classification	**Forward movement**	156	1	4
	Left movement	2	11	0
	Right movement	1	0	39
Overall accuracy (%)		96.29		

IV. CONCLUSIONS

This study demonstrated an interesting exploration of BNN classification in developing an effective obstacle avoidance algorithm for an educational autonomous mobile robot platform. The BNN was trained offline using the quasi-Newton optimization technique, and its structure was also optimized based on the Bayesian approach. The proposed BNN can handle the uncertainty of the training data, so that it can generalize well with unseen data. The future direction of this research is to investigate obstacle avoidance algorithms based on other machine learning algorithms, including Decision Trees, Support Vector Machines, Gaussian Naive Bayes, K-Nearest Neighbors, and Random Forests. Ffurther studies should also be performed using other input devices, such as laser sensors and cameras, to develop more effective obstacle avoidance methods.

ACKNOWLEDGMENT

The authors would like to acknowledge Professor Ian Nabney at the University of Bristol, United Kingdom, for spending his valuable time supporting the free open-source Netlab software used in this research.

REFERENCES

[1] O. Esan, S. Du, and B. Lodewyk, "Review on Autonomous Indoor Wheel Mobile Robot Navigation Systems," in *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, Durban, South Africa, Dec. 2020, pp. 1–6, https://doi.org/10.1109/icABCD49160.2020.9183838.

[2] S. Kumar, M. Majeedullah, A. B. Buriro, and Rohibullah, "Autonomous Navigation and Real Time Mapping Using Ultrasonic Sensors in NAO Humanoid Robot," *Engineering, Technology & Applied Science Research*, vol. 12, no. 5, pp. 9102–9107, Oct. 2022, https://doi.org/10.48084/etasr.5180.

[3] M. I. Ibrahim, N. Sariff, J. Johari, and N. Buniyamin, "Mobile robot obstacle avoidance in various type of static environments using fuzzy logic approach," in *2014 2nd International Conference on Electrical, Electronics and System Engineering (ICEESE)*, Kuala Lumpur, Malaysia, Sep. 2014, pp. 83–88, https://doi.org/10.1109/ICEESE.2014.7154600.

[4] L. Ren, W. Wang, and Z. Du, "A new fuzzy intelligent obstacle avoidance control strategy for wheeled mobile robot," in *2012 IEEE International Conference on Mechatronics and Automation*, Chengdu, China, Dec. 2012, pp. 1732–1737, https://doi.org/10.1109/ICMA.2012.6284398.

[5] A. Pandey, R. K. Sonkar, K. K. Pandey, and D. R. Parhi, "Path planning navigation of mobile robot with obstacles avoidance using fuzzy logic controller," in *2014 IEEE 8th International Conference on Intelligent Systems and Control (ISCO)*, Coimbatore, India, Jan. 2014, pp. 39–41, https://doi.org/10.1109/ISCO.2014.7103914.

[6] S. H. A. Mohammad, M. A. Jeffril, and N. Sariff, "Mobile robot obstacle avoidance by using Fuzzy Logic technique," in *2013 IEEE 3rd International Conference on System Engineering and Technology*, Shah Alam, Malaysia, Dec. 2013, pp. 331–335, https://doi.org/10.1109/ICSEngT.2013.6650194.

[7] Y. Chen, Y. Wang, and X. Yu, "Obstacle avoidance path planning strategy for robot arm based on fuzzy logic," in *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, Guangzhou, China, Sep. 2012, pp. 1648–1653, https://doi.org/10.1109/ICARCV.2012.6485438.

[8] B. Kasmi and A. Hassam, "Comparative Study between Fuzzy Logic and Interval Type-2 Fuzzy Logic Controllers for the Trajectory Planning of a Mobile Robot," *Engineering, Technology & Applied Science Research*, vol. 11, no. 2, pp. 7011–7017, Apr. 2021, https://doi.org/10.48084/etasr.4031.

[9] H. Medjoubi, A. Yassine, and H. Abdelouahab, "Design and Study of an Adaptive Fuzzy Logic-Based Controller for Wheeled Mobile Robots Implemented in the Leader-Follower Formation Approach," *Engineering, Technology & Applied Science Research*, vol. 11, no. 2, pp. 6935–6942, Apr. 2021, https://doi.org/10.48084/etasr.3950.

[10] W. Budiharto, "Intelligent Surveillance Robot with Obstacle Avoidance Capabilities Using Neural Network," *Computational Intelligence and Neuroscience*, vol. 2015, May 2015, Art. no. e745823, https://doi.org/10.1155/2015/745823.

[11] K. H. Chi and M. F. R. Lee, "Obstacle avoidance in mobile robot using Neural Network," in *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Xianning, China, Apr. 2011, pp. 5082–5085, https://doi.org/10.1109/CECNET.2011.5768815.

[12] B. Ko, H. J. Choi, C. Hong, J. H. Kim, O. C. Kwon, and C. D. Yoo, "Neural network-based autonomous navigation for a homecare mobile robot," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju, Korea (South), Oct. 2017, pp. 403–406, https://doi.org/10.1109/BIGCOMP.2017.7881744.

[13] K. K. A. Farag, H. H. Shehata, and H. M. El-Batsh, "Mobile Robot Obstacle Avoidance Based on Neural Network with a Standardization Technique," *Journal of Robotics*, vol. 2021, Nov. 2021, Art. no. e1129872, https://doi.org/10.1155/2021/1129872.

[14] D. J. C. MacKay, "The Evidence Framework Applied to Classification Networks," *Neural Computation*, vol. 4, no. 5, pp. 720–736, Sep. 1992, https://doi.org/10.1162/neco.1992.4.5.720.

[15] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, UK: Clarendon Press, 1995.

[16] I. Nabney, *NETLAB: Algorithms for Pattern Recognition*. Berlin, Germany: Springer Science & Business Media, 2002.

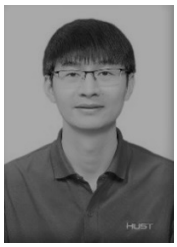
[17] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, no. 4, pp. 525–533, Jan. 1993, https://doi.org/10.1016/S0893-6080(05)80056-5.

AUTHORS PROFILE

Tuan V. Pham received a B.Eng., an M.Sc., and a Ph.D. in electrical engineering from Hanoi University of Science and Technology (HUST), Vietnam, in 2008, 2012, and 2019, respectively. He has worked as a lecturer in the Faculty of Electrical Engineering, at Vinh University of Technology Education, Vietnam. His research interests include electric machines and drives, power electronics, electrical motor parameter estimation, artificial intelligence, and machine learning.

Son T. Nguyen received a B.Eng. and an M.Sc. in electrical engineering from Hanoi University of Science and Technology (HUST), Vietnam, in 1997 and 1999, and a Ph.D. in electrical engineering from the University of Technology, Sydney (UTS), Australia in 2007. He worked as a postdoctoral researcher in computer science at the University of Kent (United Kingdom) in 2012. He holds an Australian patent for an effective hands-free control system used by severely disabled people based on head direction and electroencephalography. He is currently a main lecturer at the School of Electrical and Electronic Engineering, HUST. His research interests include electric machines, power electronics, and assistive technologies for severely disabled people.

Anh Hoang is currently a lecturer at the Hanoi University of Science and Technology. He received his Ph.D. in Sustainable Industrial Systems from Lorraine University, France. He is currently focusing on design and installation of electrical systems, sustainability, and renewable energy systems. He was also active in energy/emission auditing programs and built up roadmap for energy efficiency standards. His current research interests include prognostic, energy audit, maintenance plan, and energy management.



Tu M. Pham received a B.Eng. and an M.Sc. in electrical engineering from the Hanoi University of Science and Technology (HUST), Vietnam, in 2009 and 2012, respectively. He is currently pursuing a Ph.D. in Electrical Engineering, at the School of Electrical and Electronic Engineering, HUST. He has worked as a lecturer in the School of Electrical and Electronic Engineering, HUST. His research interests include high-efficiency electrical machines, power reactors, reactive power compensation technology, and renewable energy.

Linh V. Trieu received an M.Sc. in electrical engineering from St. Petersburg State Transport University (PGUPS), Russia, in 1993 and a Ph.D. in electrical engineering from the Russian University of Transport (MIIT) in 1997. He has worked as a lecturer at the Hanoi University of Science and Technology (HUST). His research interests include the design of electrical machines.

Trung T. Cao was born in Vietnam in 1978. He received a B.S. and an M.Sc. in automatic control from the Hanoi University of Science and Technology (HUST), in 2001 and 2010, respectively. Since 2002, he has been a lecturer at the School of Electrical and Electronic Engineering, HUST. His research interests include fault detection and isolation, optimization, and optimal control.