Radial Basis Function Neural Network Controlled Mobile Robot Navigation and Obstacle Avoidance in Various Environments

Anish Pandey, Kavita Burse, Shalini Pandey
Department of Electronics and Communication Engineering
Oriental Group of Institutes, Bhopal, India
E-mail: anish06353@gmail.com

Received 06 March 2017, received in revised form 11 March 2017, accepted 15 March 2017

Abstract: **Navigation and Obstacle avoidance are two fundamental problems in robotics.This article presents the sensor based Radial Basis Function Neural Network (RBFNN) controlled mobile robot navigation and obstacle avoidance in various environments. This RBFNN controller provides the smooth and continuous steering angle commands to the robot using infrared range sensors data interpretation. The inputs of the RBFNN controller are the obstacle distance received from the output of the sensors, and output from RBFNN controller is steering angle of the mobile robot. Graphical simulations are designed through MATLAB graphical user interface (GUI) on the computer and implemented it in real time by using C/C++ language running Arduino microcontroller based differential-drive four-wheeled mobile robot. Successful robot navigation and obstacle avoidance in computer simulation and experiments verify the effectiveness and efficiency of the RBFNN controller.**

Keywords: **Navigation; Obstacle avoidance; Sensors; Steering angle; Graphical simulation.**

1. INTRODUCTION

Obstacle avoidance is very important for successful navigation of autonomous mobile robot.During navigation and obstacle avoidance, the mobile robot faces basically two types of environments i.e. static and dynamic environments. In the static environment, obstacles are in stationary condition. In contrast, in the dynamic environment, obstacles move with some velocity. Thenavigationand obstacle avoidance problems of an autonomous wheeled mobile robot arebeing solved by many researchers in the past two-three decades using the various soft computing techniques such as Neural network [1], Fuzzy logic [2], and Adaptive neuro-fuzzy[3] etc.

Neural network technique is motivated from the human brain, which is being appliedby many researchers in the different fields such as signal and image processing, pattern recognition, robotics, and business, etc.In reference [1], the authors have solved the trajectory tracking problem of unicycle wheeled mobile robot by using RBFNN controller. Yang and Meng [4] have applied the biologically inspired neural network to generate a collision-free path in a nonstationary environment. Rossomando and Soria [5] have designed an adaptive neural network PID controller to solve the trajectory tracking control problem of a mobile robot.In [2], the authors have proposed type-2 fuzzy neural network (IT2FNN) to solve the obstacle avoidance and position stabilization problems of wheeled mobile robots. Pandey et al. [3] have implemented the adaptive neuro-fuzzy technique for mobile robot navigation and hurdle avoidance in the various environments.Motivated by the aforementioned literature reviews, the primary objective of this article is to control the steering angle of the robot in the various environments using RBFNN based autonomous sensor-actuator technique.

2. RADIAL BASIS FUNCTION NEURAL NETWORK (RBFNN) CONTROLLER FOR THE MOBILE ROBOT NAVIGATION AND OBSTACLE AVOIDANCE

In this section, the RBFNN will be applied to control the steering angle of the mobile robot usingdifferent training patterns, which are listed in Table 1. The proposed RBFNN controller has three inputs and a single output. The inputs of the RBFNN controller are the obstacle distance received from the output of the multiple sensors, and output from RBFNN controller is steering angle of the mobile robot. The three inputs are addressed by Front Sensor Reading (F_s), Left Sensor Reading (L_s), and Right Sensor Reading (R_s), respectively. Similarly, the output of RBFNN controller is denoted by steering angle (S_a). Figure 1illustrates the general structure of a RBFNN controller. The RBFNN is similar to the traditional feed forward neural network (FNN). The RBFNN consists of three layers such as input layer, intermediatelayers, and the output layer; all the layers are connected to each other by the neurons [6].The intermediate layer is the combination of Radial Basis Function (RBF) units and bias. The RBF units are also connected to the output layer.In the present study, Gaussian transfer function has selected as an RBF. This RBF calculates the Euclidean distance between the input variables and the center of RBF. The hidden layer equation of the RBFNN can be written as follows [7]: -

$$\varphi_i(u) = \exp\left[-\frac{1}{2}\left(\frac{\|u-c_i\|^2}{\sigma_i^2}\right)\right] \qquad \left(i = 1,\ldots,n_h\right) \tag{1}$$

where φ_i denotesi the output of the ith RBF unit, u is the input variables, and $\|\cdot\|$ is the Euclidean norm. For the ith RBF unit, c_i and σ_i are the center and width, respectively. n_h is the number of hidden neurons.

The jth output (v_j) for input variables (u) of the RBFNN can be computed as follows: -

$$v_j(u) = \sum_{i=1}^{n_h}\left(w_{ij}\cdot\varphi_i(u)\right)+b_j \qquad \left(j = 1,\ldots,n_o\right) \quad (2)$$

Where w_{ij} is the weight connection from the ith hidden neuron to the jth output neuron, n_o is the total number of output neurons, and b_j is the bias parameter of the jth output neuron.

Table 1: The different training patterns for mobile robot navigation and obstacle avoidance

F_s (meter)	L_s (meter)	R_s (meter)	S_a (degree)	Robot direction
0.2	1.15	0.2	74.3	Left
0.2	0.2	1.5	-65.9	Right
0.25	0.75	0.5	55	Left
0.4	1.2	0.6	59.4	Left
0.25	0.5	1.2	-22.9	Right
0.22	0.25	0.22	73.4	Left
0.5	0.25	0.25	0	Straight
1	0.28	0.25	0	Straight
0.25	0.21	0.22	77.2	Left
1.5	0.25	1.15	-70.5	Right
1.5	0.2	0.25	0	Straight
1.5	1	1	-70.4	Right

The overall performance of the RBFNN controller is evaluated by the calculation of mean squared error (MSE), and root mean square error (RMSE) between target output value and neural network output value: -

$$MSE(\%) = \left[\frac{1}{r}\sum_{1}^{r}\left(v_{(t)} - v_{(n)}\right)^2\right]\times 100 \quad (3)$$

$$RMSE(\%) = \sqrt{\frac{1}{r}\left[\sum_{1}^{r}\left(v_{(t)} - v_{(n)}\right)^2\right]}\times 100 \quad (4)$$

where $v_{(n)}$ is the target (actual) output value, $v_{(n)}$ is the neural network (predicted) output, and r is the number of compounds in the analyzed set.

3. SIMULATION WITH EXPERIMENTAL RESULTS

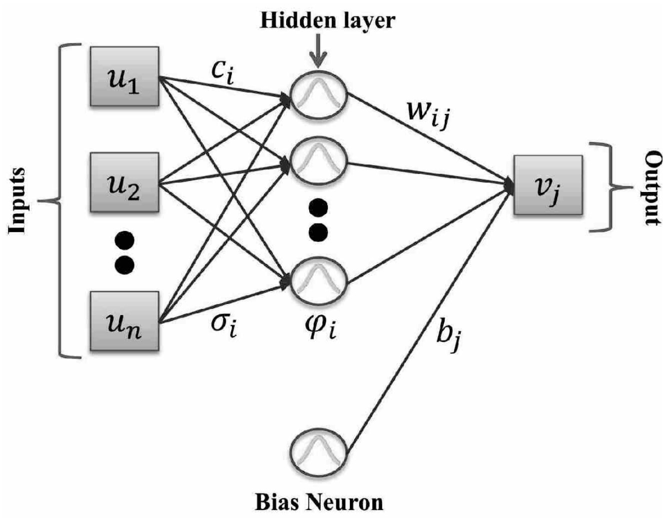

Figure 1: The general structure of the proposed RBFNN Control

To show the effectiveness of the RBFNN controller, the various simulations and experiments are conducted through MATLAB graphical user interface (GUI) on the computer and implemented it in real time by using C/C++ language running Arduino microcontroller based four-wheeled mobile robot (see the Fig. 2). The experiment robot has a four-wheeled differential drive. The motion and orientation of the robot are controlled by independent four separate DC geared motors, which provides the necessary torque to all driving wheels. The width of the robot chassis is 20cm, length is 25cm, the track width of wheels is around 26cm, wheel diameter is 5cm, and width of the wheels is 1cm. The three groups of the infrared range (IR) sensors are attached in the front, left, and right side of the robot and its sensing range between 0.2m to 1.5m. These sensors read the front, left and right obstacles. The top speed of the wheels is 0.18 m/sec. In every scenario of the simulation and experimental, it is assumed that the environment is completely unknown for the mobile robot except goal point.

Figure 3 and 4 shows the snapshots of the successful mobile robot navigation results. The width and height of the environments are 2.5m and 2.5m, respectively. In figure 3, the starting position (x, y) of the mobile robot is (2.0, 0.5) m. The position of the goal is (0.1, 2.0) m. The initial angle between the mobile robot and the goal is 38.29°. Similarly, in figure 4, the starting position (x, y) of the mobile robot is (0.05, 0.5) m. The position of the goal is (0.9, 0.5) m. The initial angle between the mobile robot and the goal is 0°.It is assumed that the environment is completely unknown for the mobile robot except of start and goal points. The average moving speed of the robot is 0.09 m/sec.In the programming, we have set a minimum threshold distance between the mobile robot and obstacle.If the robot finds the obstacle within the threshold range, then the proposed controller will be activated, and it will provide appropriate, and continuous steering angle commands to the mobile robot.From the Table 1, if the robot found obstacles at a distance 0.2m to the front, 1.15m to the left, and 0.2 m to the right, then the mobile robot turns at an angle of 74.3° means positive steering angle to reach the goal with obstacle avoidance. If the robotfound obstacles at a distance 0.2m to the front, 0.2m to the left, and 1.5m to the right, then the mobile robot turns at an angle of -65.9°means negative steering angle to reach the goal in an environment with obstacle avoidance.

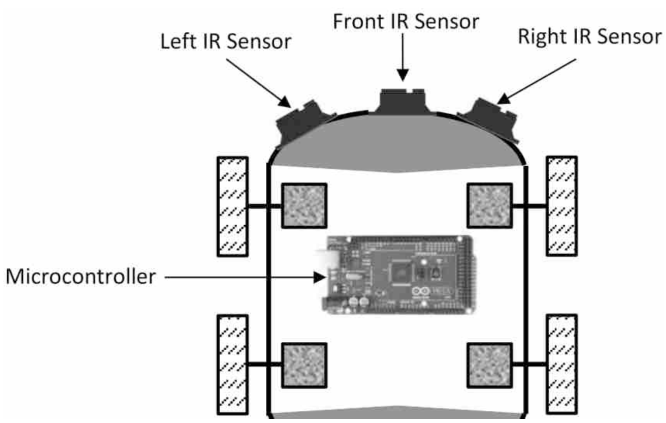

Figure 2. Four-wheeled experimental mobile robot

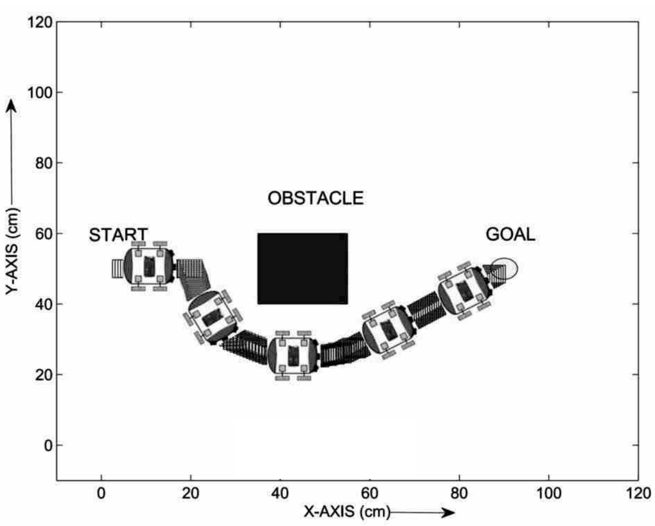

Figure 3. Successful mobile robot navigation between the obstacle.

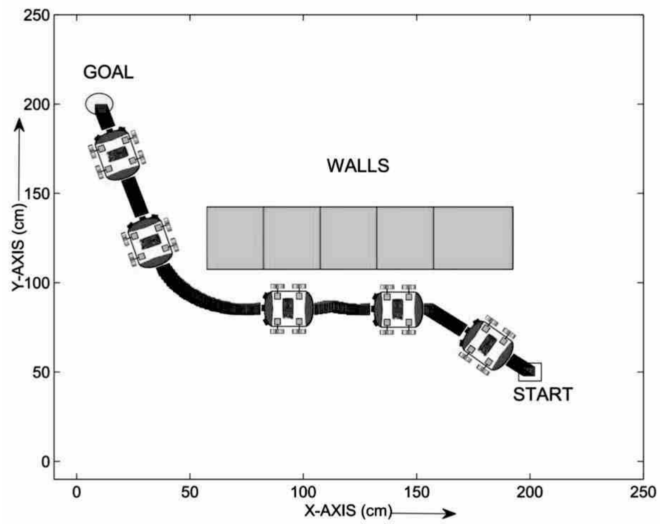

Figure 4. Successful mobile robot navigation between the walls.

4. CONCLUSION AND FUTURE SCOPE

In the current study, we have designed and implemented RBFNN controller for robot navigation and obstacle avoidance in various environments. The proposed RBFNN controller have provided the appropriate, and continuous steering angle commands to the robot using infrared range sensors data interpretation.The simulation and experimental results show that the RBFNN controlled mobile robot has successfully avoided the obstacles in the different environments.In future research, this proposed controller can be applied for multiple mobile robot navigation and obstacle avoidance.

APPENDIX

Pseudocode of the Algorithm

Program:

Take input:

start x[j], start y[j], goal x[j], goal y[j]

initialize the array of obstacles x[j], and y[j]

theta = tan^{-1}((goal y[j] - start y[j]) / (goal x[j] - start x[j]))

while (true)

start x[j+1] = start x[j] + cos (theta);

start y[j+1] = start y[j] + sin (theta);

Front Sensor Reading(F_s)calculated;

Left Sensor Reading(L_s)calculated;

Right Sensor Reading(R_s)calculated;

goal distance calculated;

and all obstacle distances calculated;

if *start x[j+1], and start y[j+1] near to obstacles x[j], and y[j]*
// Radial Basis Function Neural Network (RBFNN) Start
Input = [F_s Training; L_s Training; R_s Training]$_{M*N}$
Target = [S_a Training]$_{1*N}$
Net = newrb(Input, Target);
Sample= [$F_s$calculated; $L_s$calculated; R_s calculated]
S_a = Sim(Net, Sample);
theta = theta + S_a|| theta = theta - S_a
else
theta = tan^{-1}((goal y[j] - start y[j+1]) / (goal x[j] - start x[j+1]))
end if
if *start x[j+1], and start y[j+1] reach goal x[j], and y[j]*
break
end if
end while
end program

REFERENCES

[1] Rossomando F. G., Soria C., and Carelli R. Autonomous Mobile Robots Navigation using RBF Neural Compensator. *ELSEVIER Control Engineering Practice* 2011; 19: 215–222.

[2] Kim C. J. and Chwa D. Obstacle Avoidance Method for Wheeled Mobile Robots Using Interval Type-2 Fuzzy Neural Network. *IEEE Transactions on Fuzzy Systems* 2015, 23(3): 677–687.

[3] Pandey A., Kumar S., Pandey K. K., and Parhi D. R. Mobile Robot Navigation in Unknown Static Environments using ANFIS Controller. *ELSEVIERPerspectives in Science*, 2016, Vol. 8, pages 421–423.

[4] Yang S. X. and Meng M. Neural Network Approaches to Dynamic Collision-Free Trajectory Generation. *IEEE Transactions on Systems, Man, and Cybernetics Applications and Reviews* 2011, 31(3): 302–318.

[5] Rossomando F. G. and Soria C. M. Design and Implementation of Adaptive Neural PID for Non Linear Dynamics in Mobile Robots. *IEEE Latin America Transactions* 2015, 13(4): 913–918.

[6] Bi T., Wen F., Ni Y., and Wu F. F. Distributed Fault Section Estimation System Using Radial Basis Function Neural Network and its Companion Fuzzy System. *ELSEVIER Electrical Power and Energy Systems* 2003; 25: 377–386.

[7] Ali A. Prediction of Temperature Elevation for Seawater in Multi-Stage Flash Desalination Plants using Radial Basis Function Neural Network. *ELSEVIER Chemical Engineering Journal* 2010; 162: 552–556.

◆ ◆ ◆