Robotica: (2019) volume 37, pp. 1346–1362. © Cambridge University Press 2019
doi:10.1017/S0263574718001558

# Navigational Control Analysis of Two-Wheeled Self-Balancing Robot in an Unknown Terrain Using Back-Propagation Neural Network Integrated Modified DAYANI Approach

Animesh Chhotray* and Dayal R. Parhi

*Robotics Laboratory, Department of Mechanical Engineering, National Institute of Technology, Rourkela, Odisha 769008, India. E-mail: dayaldoc@yahoo.com*

(Accepted December 22, 2018. First published online: February 15, 2019)

**SUMMARY**

The present paper discusses on development and implementation of back-propagation neural network integrated modified DAYANI method for path control of a two-wheeled self-balancing robot in an obstacle cluttered environment. A five-layered back-propagation neural network has been instigated to find out the intensity of various weight factors considering seven navigational parameters as obtained from the modified DAYANI method. The intensity of weight factors is found out using the neural technique with input parameters such as number of visible intersecting obstacles along the goal direction, minimum visible front obstacle distances as obtained from the sensors, minimum left side obstacle distance within the visible left side range of the robot, average of left side obstacle distances, minimum right side obstacle distance within the visible right side range of the robot, average of right side obstacle distances and goal distance from the robot's probable next position. Comparison between simulation and experimental exercises is carried out for verifying the robustness of the proposed controller. Also, the authenticity of the proposed controller is verified through a comparative analysis between the results obtained by other existing techniques with the current technique in an exactly similar test scenario and an enhancement of the results is witnessed.

KEYWORDS: Two-wheeled self-balancing robot; Navigational control; Modified DAYANI; Back-propagation neural network; Path planning.

## 1. Introduction

The ever-increasing demands of mobile robots in various sectors like manufacturing, transporting, surveillance and planetary exploration have evoked a lot of interest among researchers for their autonomous control. Apart from the industrial automation, mobile robots can also be used as potential candidates for many domestic applications such as assistance in medical surgery, guidance of disable people, cleaning and monitoring like many household purposes. During accomplishment of the above sophisticated operations, these robots have to interact with their surroundings which may be static or dynamic in nature. The current work addresses the development of an autonomous navigational controller for the two-wheeled self-balancing robot (TWSBR) without any outside human assistance. The navigational controller deals with the motion planning of the robot in the workspace through continuous generation of feasible set of positions from the initial position to the target location and subsequent translation and rotations by negotiating with the intermediate obstacles.

* Corresponding author. E-mail: chhotrayanimesh@gmail.com

Based on the available information of the surrounding environment of the robot, the path planning is categorized as offline or global path planning and online or local path planning.[1] If the environment is structured in nature with known static obstacles or prior information about followed trajectory of moving obstacles is provided in advance, then it is the case of a global path planning. Whereas, local path planning deals with unstructured environments with no prior knowledge of the robot's surrounding. Thus, it relies on the online sensor readings for assessing the dynamic environment. But practically, all path planning algorithms actually begin with an off-line manner initially, and when the robot finds an obstacle in its path, the path planning switches to online mode. For this, Zhang et al.[2] have proposed an approach by combining both global as well as local path planning in order to guarantee the completeness of finding a path to the goal. The approach for solving the path planning problems can be broadly divided into classical approaches and heuristic methods.[3,4] The major classical methods comprised of Roadmap-based method, cell decomposition method, artificial potential field and mathematical programming method.[5] Although most of the motion planning problems can be resolved by the above techniques, usually these are not certainly mutually exclusive in nature. The widely accepted Roadmap methods are Voronoi diagram,[6] Visibility graph,[7] Subgoal Network and Silhouette method.[5] In the cell decomposition method, the workspace is divided into cells or grids of some specific size and the robot searches its route in free space cell graph. Simultaneous localization, mapping and planning of the surrounding for robot trajectory control are realized through an adaptive cell decomposition method incorporated with a laser range finder-based image matching.[8]

It has been reported from most of the literature that though classical approaches can generate an obstacle-free path, they still suffer from many drawbacks like greater computational complexity, more execution time, trapping in local minima and providing lesser optimal solution.[9] Therefore, various heuristic methods have been developed to overcome the inefficiency of the classical methods.[10] Kim[11] has proposed a virtual escaping route method based on artificial potential functions for avoiding a trap situation in local minima. Ismail et al.[12] have developed an idea for reducing the number of steps involved for robot motion from start to goal point by using genetic algorithm in a static grid environment. Similarly, Kala et al.[13] have proposed a modified version of A* algorithm for path planning in a static environment by implementing a multineuron heuristic search method based on probability-based fitness function. A comparison of the performance of navigational controllers by considering three different types of fuzzy membership functions has been presented by Pradhan et al.[14] for multiple mobile path planning in completely unknown environments. Rao et al.[15] have proposed an improved krill herd method for generating safe way points in order to traverse a collision-free path in dynamic environments. Deepak et al.[16] have introduced two fitness functions dealing with the distances between each swarm agent to that of the target position and adjacent obstacle in their particle swarm optimization-based mobile robot navigation system for determining the global best position. Savkin and Li[17] have presented a randomized navigation algorithm for complete search and map building in a closed obstacle muddled surroundings with the help of range finder sensors of Pioneer3D-x robot. Likewise, various hybrid optimization algorithms have been implemented in both static and dynamic environments for single as well as multiple robots path planning in complex cluttered environments.[18–20] Recently, Kumar et al.[21,22] have demonstrated the path planning of humanoid robots in static as well as dynamic environments by implementing both statistical and heuristic methods.

An artificial neural network infused rule-based heuristic method has been implemented by Parhi and Singh[23] for autonomous robot motion in a cluttered environment. Here, back-propagation algorithm helps the training of the neural network. Bassil[24] has come with a three-layered artificial neural network for autonomous navigation of a rover in planetary terrain. The off-line training of the network is realized through a supervised back-propagation learning approach. Both off-line and subsequent online learning are carried out for training of a recurrent neural network by Al-Sagban and Dhaouadi[25] for avoiding obstacles in case of differential drive robots. The real-time autonomous mapping of grid environments with the help of restricted sensor readings is realized through a biologically inspired neural network by Luo et al.[26] for dynamically building the exact map of the immediate surrounding for navigation. Boukens and Boukabou[27] have obtained satisfactory performance in tracking the desired trajectory of nonholonomic two-wheeled robots by designing a robust controller through application of a neural network system. Recently, Farias et al.[28] have also implemented artificial neural networks for establishing an obstacle detection model by the fusion of the

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

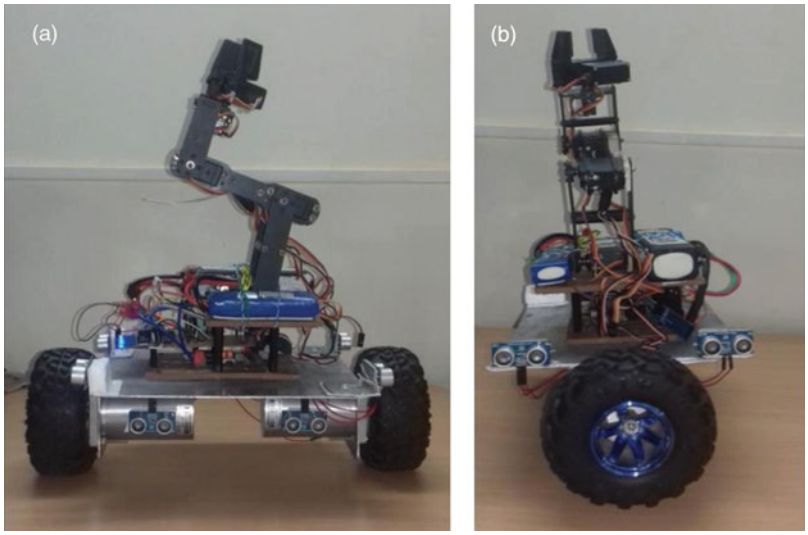


Fig. 1. Lab-built TWSBR. (a) Front view and (b) side view.

distance estimation information gathered from different proximity sensors. A multilayer feed forward neural network has been developed by Singh and Parhi[29] for optimization of path length and time elapsed during navigation. Various cognitive tasks like learning, training, adaption and generalization can be dealt with by the use of this back-propagation neural network. Singh and Thongam[30] also have demonstrated a novel method for generating a near-optimal path during navigation avoiding static and moving obstacles in a varying surrounding conditions using multilayer neural network system.

From the above extensive literature analysis, it has been revealed that the controllers developed by combining the traditional path planning algorithms with the heuristic control strategies have shown better performance in terms of path length and obstacle avoidance behavior. Therefore, the current analysis deals with the development of a navigational controller for successful path planning of a TWSBR in unknown cluttered terrain by implementing back-propagation neural network integrated with modified DAYANI approach. Unlike global path planning approach of DAYANI arc contour method[31] in static environment, here the local path search is realized through the ultrasonic range finders. As TWSBRs are statically unstable in nature, most of the available literatures are on dynamic modeling and stability control analysis of the vehicle.[32–34] Articles representing the trajectory tracking and navigational behavior of the TWSBR are very less till date and most of them only deal with forward or backward motion with yaw orientation in simulation platform only.[35–39] But, in this paper, real-time obstacle avoidance and target reaching capability of a lab-built TWSBR are realized through experimental verification by incorporating the proposed control architecture. Also, the developed controller is compared with some of the existing controllers in terms of path length travelled while traversing from the initial start to goal position in a complex obstacle cluttered environment.

## 2. Modeling of TWSBR

### *2.1. System modeling*

The lab-built TWSBR consists of a three-layered structural assembly where the bottom layer is comprised of two differentially driven DC motors mounted on the common axis and connected with two wheels on both sides. The microcontroller units MK20DX256VLH7 and ATMEGA 328P along with the IMU sensor unit MPU6050 form the middle layer of the prototype. The batteries and the manipulator are placed at the upper layer for sifting the center of gravity towards upward direction as shown in Fig. 1.

The attitude estimation of the TWSBR in the form of pitch angle and pitch angular velocity has been realized through the accelerometer and gyroscope of the inertial measurement unit. Also, the optical encoders provide the position and linear velocity information about the robot to the

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

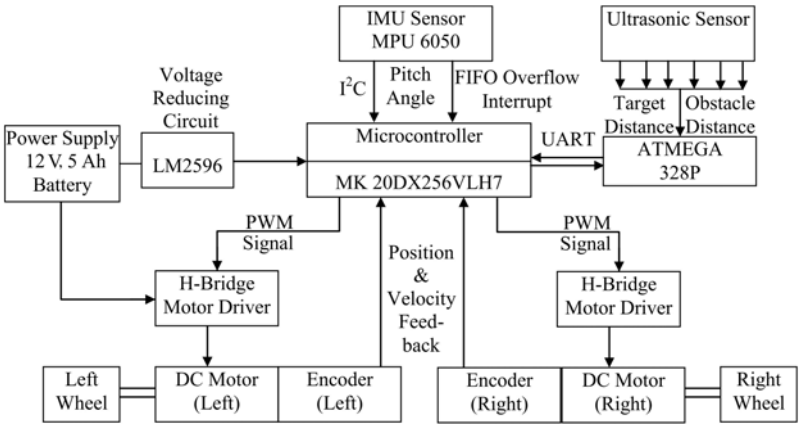


Fig. 2. Block diagram of TWSBR.

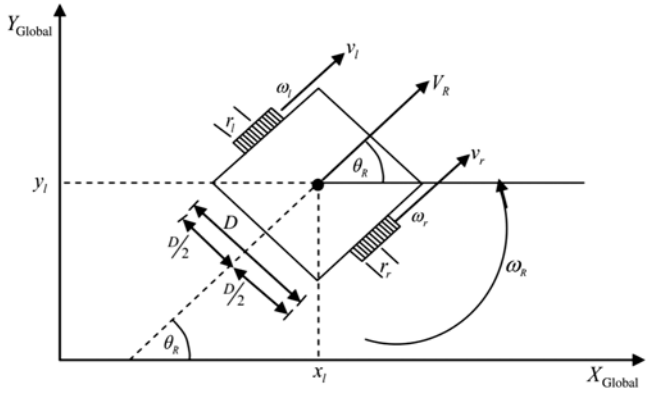


Fig. 3. Kinematic modeling of TWSBR.

microprocessor. Simultaneously, the array of ultrasonic sensors gives obstacle and the target distances. The microcontroller processes the data received from the sensors by the implementation of the control algorithm and delivers the pulse-width modulated signals to the motor driver circuits for the subsequent motion of the robot. The DC motors receive the power supply from the batteries of 12 V and 5 A capacity and are regulated by the motor drivers as commanded by the processor. The requisite communication with the remote PC-like control program implementation and outcome extraction is established through USB interface and a Bluetooth module. The complete block diagram of the lab-built TWSBR is depicted in Fig. 2.

### *2.2. Kinematic modeling*

In order to understand the mechanical behavior of the robot, a prior knowledge of its proper kinematic model is highly essential. This will help in establishing the relation between robot's position and orientation while performing some desired tasks.[40] Before formulating the kinematic model of the actual TWSBR, some assumptions are to be followed for simplification. The robot must be considered as a rigid body and its wheels follow a pure rolling motion without any slippage or skidding. The friction between the floor and rotating parts of the robot is negligible with a point contact. The operational plane should always be horizontal and the steering axis of the robot remains orthogonal to it.

The TWSBR has two independently driven wheels with a common axis of rotation. The radius of the wheel is '$r$' and the distance between the wheels is '$D$' as shown in Fig. 3. The linear and angular velocities of each individual wheels are denoted as $v_l$, $\omega_l$ and $v_r$, $\omega_r$ for left and right wheels, respectively. Then the TWSBR's linear and angular velocities are represented as

$$V_R = \frac{v_r + v_l}{2}, \ \omega_R = \frac{v_r - v_l}{D} \tag{1}$$

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

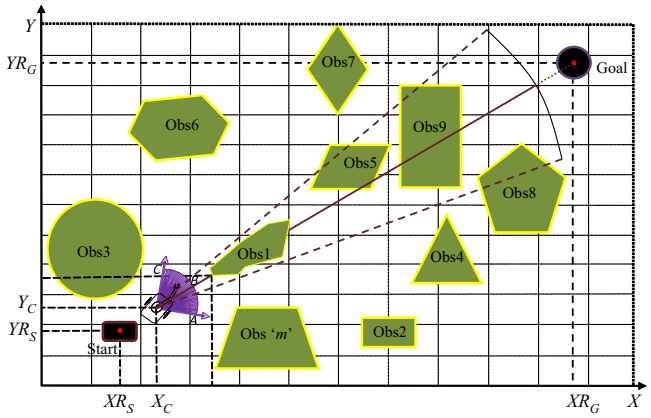


Fig. 4. Scenario consisting of robot, obstacles and intersecting obstacles within visible range.

During navigation, the current location and orientation of the robot are described by a local reference frame. If the heading angle of the robot is $\theta_R$, then its local position ($\dot{\xi}_l$) can be mapped with the global positions ($\dot{\xi}_g$) as mentioned below:

$$\dot{\xi}_l = R(\theta_R)\dot{\xi}_g = \begin{bmatrix} \cos(\theta_R) & \sin(\theta_R) & 0 \\ -\sin(\theta_R) & \cos(\theta_R) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_g(t) \\ \dot{y}_g(t) \\ (\dot{\theta}_R)_g(t) \end{bmatrix} \tag{2}$$

where the global reference frame is related to the robot linear and angular velocities as

$$\begin{bmatrix} \dot{x}_g(t) \\ \dot{y}_g(t) \\ (\dot{\theta}_R)_g(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta_R) & 0 \\ \sin(\theta_R) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_R \\ \omega_R \end{bmatrix} \tag{3}$$

## 3. Control Architecture of Back-Propagation Neural Network Integrated Modified DAYANI Approach

In the current analysis, the TWSBR follows local path planning approach while moving from the known start position to the goal position in an unknown environment populated with static obstacles. After the onset of motion from the initial position, the robot perceives its surrounding environment by the readings received from the array of eight onboard ultrasonic sensors. Once it detects obstacles present inside the visible range of the sensors, the back-propagation neural network integrated modified DAYANI algorithm forces into action. This determines the most feasible next point of motion during the motion from start to target location so that the robot moves without collision with the obstacles in-between. The following section describes how the various inputs are calculated by implementing modified DAYANI approach and subsequent five-layered back-propagation neural network architecture for calculation of several weight factors involved in the calculation of the overall weight of each feasible point of the arc contour.

### *3.1. Analysis of modified DAYANI approach*

In the following section, modified DAYANI method has been discussed for determining various inputs to the neural network system for calculating the weight factors used in navigational path analysis of TWSBR.

*3.1.1. Calculation of number of intersecting obstacles within the possible movement direction of robot.* As shown in Fig. 4, the obstacles present in the concerned environment are denoted as obstacle 1 (Obs1) to obstacle *m* (Obs '*m*'). The particular robot locations like initial position, current position and goal position are represented by their Cartesian coordinates such as ($XR_S$, $YR_S$), ($X_C$, $Y_C$) and ($XR_G$, $YR_G$), respectively. Here, the shortest path traversed by the robot at any instance will be the line joining the robot position ($X_C$, $Y_C$) at that instance with the goal position ($XR_G$, $YR_G$).

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

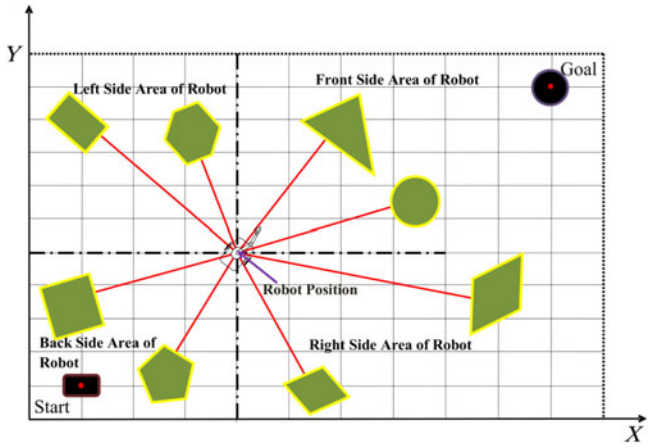


Fig. 5. Scenario showing the obstacles position around the robot.

Robot follows that shortest path until it encounters any obstacles in its visible range. The most feasible next position of motion of the robot ($X_{C+1}$, $Y_{C+1}$) is to be determined from the '$n+1$' number of possible points of the arc 'ABC'. The arc 'ABC' is disintegrated into '$n/2$' equal segments on both side of the line joining the robot current position ($X_C$, $Y_C$) and goal position ($XR_G$, $YR_G$). The total number of obstacles present in the front visible range of the robot is denoted as '$RQ$'. In the present case, $RQ = 4$ (Obs1, Obs5, Obs8 and Obs9).

*3.1.2. Finding the minimum distance of the obstacle in the front side visible direction.* Here, in this case, from the sensor readings, the minimum intersecting distance from the robot to the obstacle boundary is calculated in the front visible direction. In the present scenario, the minimum obstacle distance ($RDF_{min}$) is obtained by considering Obs1 according to Fig. 4 out of total four number of obstacles in the front visible range.

*3.1.3. Finding minimum obstacle distance in left side visible direction.* According to Fig. 5, the minimum distance of the obstacle present within the left side visible range of the robot is calculated based on the sensor inputs at each instance of time. Here, the minimum left side obstacle distance is denoted as $RDL_{min}$.

*3.1.4. Determination of average obstacle distance in left side visible direction.* In the present scenario, average intersecting distance in the left side visible direction of the robot is calculated according to the sensor inputs. The left side average obstacle distance is denoted as $RDL_{avg}$. If there are '$L$' numbers of intersecting obstacles in the left side of the robot, then the average obstacle distance can be written as follows:

$$RDL_{avg} = \left(RDL_1 + RDL_2 + RDL_3 + \ldots.. + RDL_L\right)/L \tag{4}$$

*3.1.5. Finding minimum obstacle distance in right side visible direction.* According to Fig. 5, the minimum distance of the obstacle present within the right side visible range of the robot is calculated based on the sensor inputs at each instance of time. Here, the minimum right side obstacle distance is denoted as $RDR_{min}$.

*3.1.6. Determination of average obstacle distance in right side visible direction.* In the present scenario, average intersecting distance in the right side visible direction of the robot is calculated according to the sensor inputs. The right side average obstacle distance is denoted as $RDR_{avg}$. If there are '$R$' numbers of intersecting obstacles in the right side of the robot, then the average obstacle distance can be written as follows:

$$RDR_{avg} = \left(RDR_1 + RDR_2 + RDR_3 + \ldots.. + RDR_L\right)/R \tag{5}$$

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

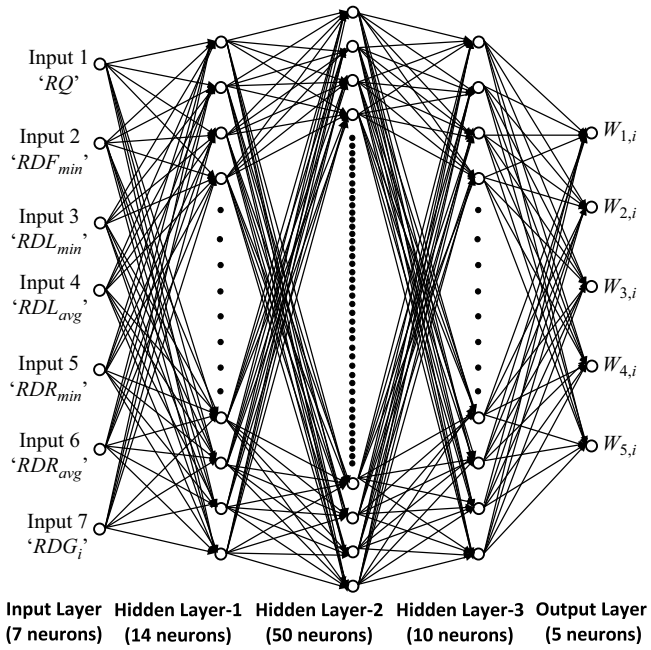


Fig. 6. Calculation of different weight factors from various environmental parameters using five-layered back-propagation neural network.

*3.1.7. Distance between robot's probable next position and goal.* There are total $n+1$ probable positions present in the arc 'ABC' to which the robot can proceed during its next step advancement. Therefore, here the distance from all those probable points to the goal point is calculated according to the following equation:

$$RDG_{i=1,n+1} = \left(\left(XR_G - X_{C+1,i}\right)^2 + \left(YR_G - Y_{C+1,i}\right)^2\right)^{0.5} \tag{6}$$

where $(X_{C+1,i},\ Y_{C+1,i})$ is the new probable '$i$th' position of the robot, where '$i$' varies from 1 to $n+1$ positions in the arc 'ABC'.

*3.1.8. Determining whether robot's next probable position is within the front obstacle or not.* During navigation, it is quite apparent that at some of the instances, the robot's probable next positions may fall within the front obstacle boundary itself. So, these positions are not to be considered further as the feasible next position for advancement. Therefore, the distance between the robot's current position and the possible '$n+1$' next positions should be always less than the minimum front obstacle distance, and the probability of existence of that probable next position can be represented according to Eq. (7).

$$\left\{\begin{array}{l} WF = 1 \text{ if } \left(\left(X_{C+1,i} - X_C\right)^2 + \left(Y_{C+1,i} - Y_C\right)^2\right)^{0.5} < RDF_{min} \\ \text{Else} \\ WF = 0 \end{array}\right\} \tag{7}$$

Several influential weight factors are calculated according to above eight cases and they are termed as $W_{1,i}$, $W_{2,i}$, $W_{3,i}$, $W_{4,I}$, $W_{5,I}$ and $W_{6,i}$. These weight factors depend upon the following input parameters: (i) intensity of obstacles in the robot motion direction; (ii) minimum obstacle distance in the front visible range; (iii) minimum and average obstacle distances in left visible range of robot; (iv) minimum and average obstacle distances in right visible range of robot; (v) distance between robot's probable next position and goal and (vi) probability of existence of the robot next step of advancement as described before. The influential weights $W_{1,i}$, $W_{2,i}$, $W_{3,i}$, $W_{4,i}$ and $W_{5,i}$ are calculated from neural network as depicted in Fig. 6. Whereas, $W_{6,i}$ = WF as stated in Eq. (7).

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

### *3.2. Description of five-layered back-propagation neural network used in modified DAYANI-neural technique*

Figure 6 describes the architecture of the five-layered back-propagation neural network used for calculation of several weight factors. The first layer of the neural network consists of seven neurons. Second, third and fourth layers are the hidden layers and consist of 14, 50 and 10 neurons, respectively. The last layer, that is, the output layer consists of five neurons as shown in Fig. 6. Several hundred training patterns are prepared to train the neural network. The inputs to the neural network are (i) number of intersecting obstacles '$RQ$' in the front direction of the robot within visible range; (ii) minimum distance '$RDF_{min}$' to the intersecting obstacle in within front visible range; (iii) minimum distance '$RDL_{min}$' to the intersecting obstacle in left side visible range; (iv) average obstacle distance '$RDL_{avg}$' in left direction of the robot within visible range; (v) minimum distance '$RDR_{min}$' to the intersecting obstacle in right side visible range; (vi) average obstacle distance '$RDR_{avg}$' in right direction of the robot within visible range and (vii) goal distance '$RDG_i$' from probable robot position for calculation of global weight factors. The outputs from the neural network are different weight factors (i.e., $W_{1,i}$, $W_{2,i}$, $W_{3,i}$, $W_{4,i}$ and $W_{5,i}$). The network learning is achieved through back-propagation algorithm which is based on the error correction methodology.[41] During training, the error signals are back-propagated until the desired output is realized. The error is calculated as a performance measure of the network as presented by[23,29]

$$E_{rr} = \frac{1}{2} \sum_{\substack{\text{all training} \\ \text{patterns}}} \left(W_{p,i_{\text{desired}}} - W_{p,i_{\text{actual}}}\right)^2 \tag{8}$$

where $W_{p,i}$ is the output from the neural network. The error is back-propagated from the outer layers to the hidden layers to train the network. A local error gradient is computed for the neurons of the hidden layers for suitable correction of weights.[42]

$$\delta_r^{\{lay\}} = f'\left(V_r^{\{lay\}}\right)\left(\sum_k \delta_k^{\{lay+1\}} N_{sr}^{\{lay+1\}}\right) \tag{9}$$

The synaptic weights are corrected according to the following equations:[23]

$$N_{qr}(t+1) = N_{qr}(t) + \Delta N_{qr}(t+1) \tag{10}$$

where

$$\Delta N_{qr}(t+1) = \alpha \Delta N_{qr}(t) + \Delta \eta \delta_r^{\{lay\}} y_q^{\{lay-1\}} \tag{11}$$

Here, $t$ = iteration number, $\alpha$ is the momentum coefficient and $\eta$ is the rate of learning.

Then, the final output from the back-propagation neural network will be

$$W_{p,i} = f\left(V_{p=1,5}^{[5]}\right) \tag{12}$$

where

$$V_p^5 = \sum_q N_{pq}^{\{5\}} y_q^{\{4\}} \tag{13}$$

The overall weight ($W_{G,i}$) for the robot's $i$th ($i = 1 - n + 1$) position out of robot probable positions from the arc 'ABC' is calculated as follows:

$$W_{G,i} = W_{1,i} \cdot W_{2,i} \cdot W_{3,i} \cdot W_{4,i} \cdot W_{5,i} \cdot W_{6,i} \tag{14}$$

From the obtained values of '$W_{G,i}$', as calculated from Eq. (14) for total '$n + 1$' conditions of the robot's possible positions, the position with maximum value of '$W_{G,i}$' is considered. The robot travels to that best next possible position according to values obtained from Eq. (14). This entire process continues in a loop till the robot achieves its target position. MATLAB code is written to implement

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

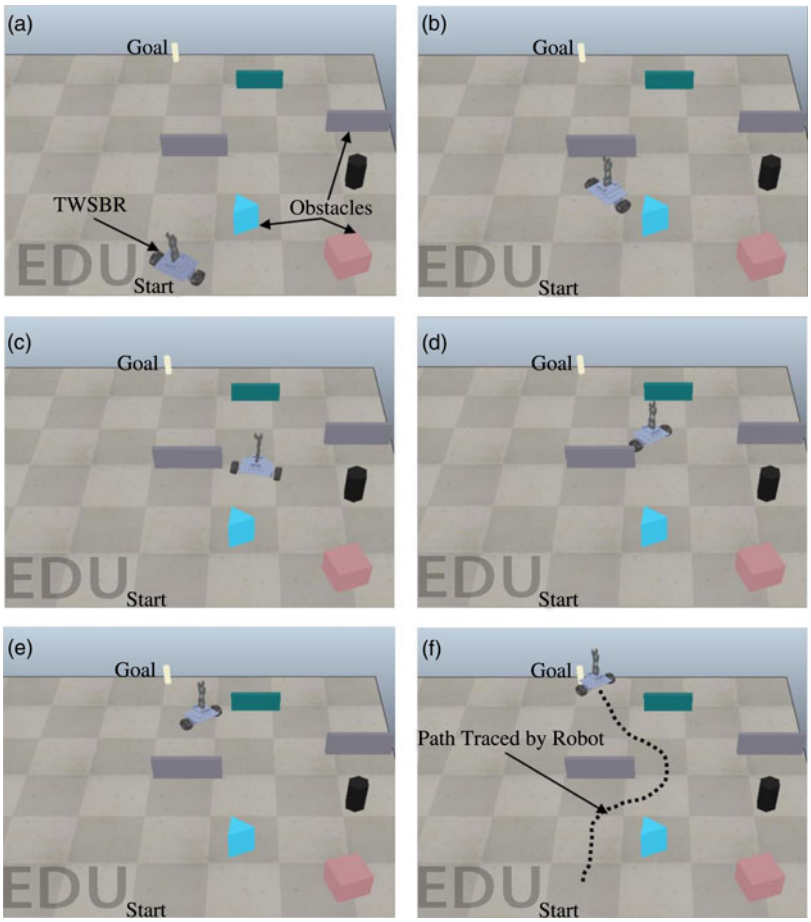


Fig. 7. Simulation results showing (a) initial position, (b-e) intermediate steps of robot navigation from start to goal position and (f ) final position during robot motion in a maze-like environment.

the neural network integrated modified DAYANI approach for path control of TWSBR in unknown terrain.

## 4. Simulation and Experimental Results

### *4.1. Navigation of TWSBR in simulation framework*

Virtual Robot Experimentation Platform (V-REP) has been selected as the simulation framework for verifying the authenticity of the proposed algorithm. V-REP is an easily coded and powerful simulation platform which can incorporate various control algorithms through its versatile programming approaches. It is a scalable simulator with multitude in-built tools having ready to use capabilities for simultaneous sensing, actuation and control of different robotic systems with reduced complexities. Also, the actual CAD model of the robot considering its system dynamics along with the obstacles of exact dimensions analogous to the experimental scenario can be integrated in this simulation framework. Here, the Solid-works model of the TWSBR is directly imported to the simulation platform and the control algorithm is implemented through Lua script. A platform of size $160 \times 240$ units is created for performing the simulation analysis. Several simulation exercises have been performed by selecting various random environments and a few of them are presented here. Figure 7 depicts the path traced by the robot during simulation from start point to goal point while avoiding the haphazardly located obstacles in a maze-like environment. Similarly in Fig. 8 also another different kind of complex cluttered environment with random static obstacles is considered for analyzing the feasibility of the neural network integrated modified DAYANI approach. In both the cases, it is observed that the robot successfully navigates among the obstacles from the start to the goal point in the shortest route.

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

Table I. Technical specification of the TWSBR used for experimental analysis.

| **Components** | **Technical descriptions** |
|---|---|
| Microcontroller | One ATmega2560 and MK20DX256VLH7, Cortex-M4, 72 MHz |
| Flash memory | 256 KB |
| RAM | 64 KB |
| Sensor | One MPU6050 and eight ultrasonic sensors |
| Actuator | Two 300 RPM DC motors with 30 kg.cm torque each |
| Encoder | 1800 counts/rotation |
| Speed | Max: 0.3 m/s |
| Power source | Two 12 V, 5000 mAh lithium-polymer batteries |
| Voltage regulator | LM2596 buck converter |
| Wireless communication | HC-05 bluetooth with UART interface and USB port |
| $L \times W \times H$ | 40 cm × 22 cm × 50 cm |
| Weight | Approx. 2732 g |
| Payload | Approx. 2000 g |
| Simulator | V-REP, MATLAB |
| Development environment | Arduino IDE and Teensyduino |

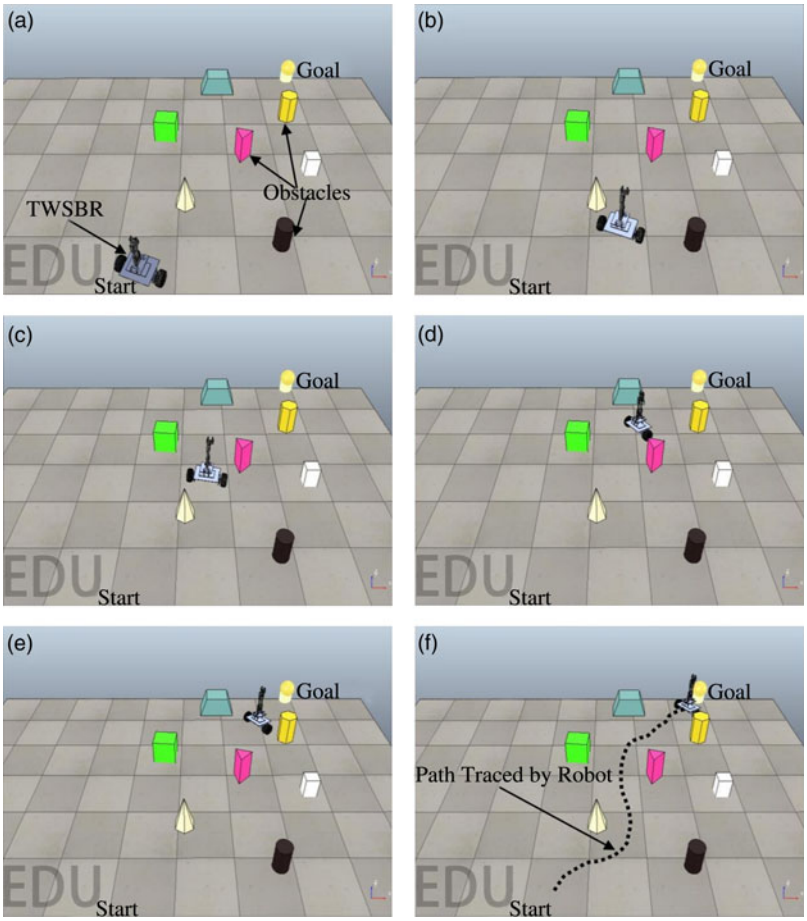


Fig. 8. Simulation results showing (a) initial position, (b-e) intermediate steps of robot navigation from start to goal position and (f ) final position during robot motion in a complex cluttered environment.

### *4.2. Navigation of TWSBR in experimental setup*

The efficiency of the proposed controller is validated through a series of real-time experiments. Here, the practical implementation of the simulation results is carried out through creation of exactly similar environment under laboratory conditions. The arena size is taken as 1.60 × 2.40 m and comprised of obstacles of exactly similar shape and size as that of the simulation window. The complete

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

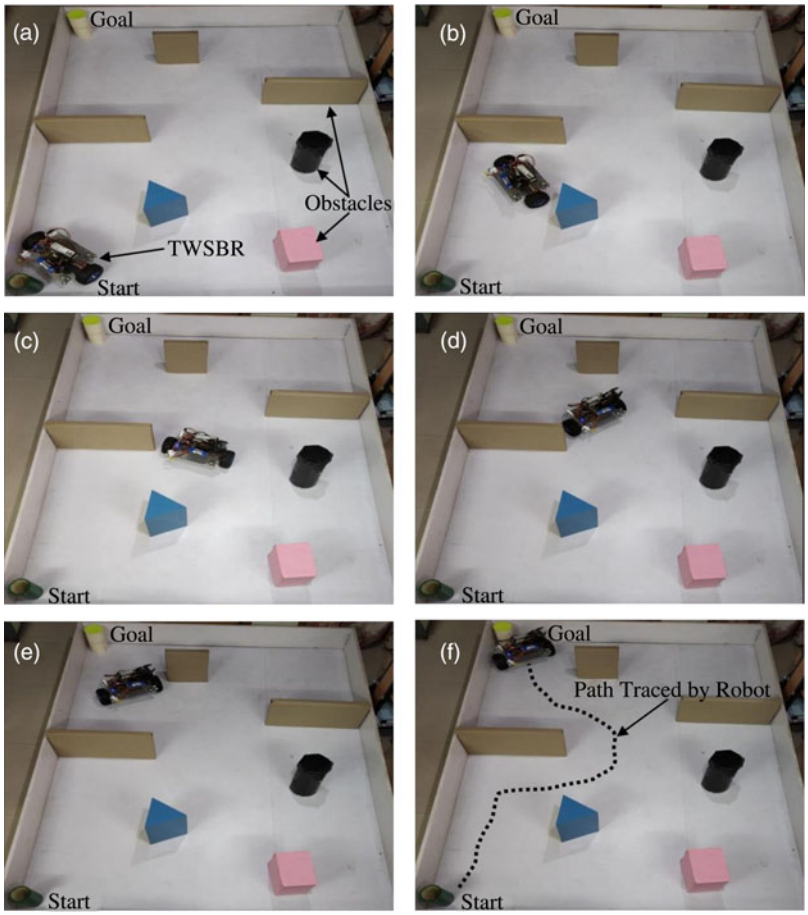


Fig. 9. Snapshots of experimental results showing (a) initial position, (b-e) intermediate steps of robot navigation from start to goal position and (f ) final position during robot motion in a maze-like environment.

specification of the TWSBR used in the experimental analysis is represented in Table I. The TWSBR is programmed through Arduino programming language due to its supportive nature to multiple programming paradigms and has a standard library. The threshold distance for the ultrasonic range finding sensors used in the current study is taken as 25 cm. The robot starts moving towards the target by searching the best next feasible point according to the neural network integrated modified DAYANI algorithm and avoids the obstacles in between. Figures 9 and 10 represent the snapshots of intermediate steps involved during the navigation.

### *4.3. Comparative analysis of simulation and experimental results*

The path length and the time taken in different environmental scenarios involved during simulation analysis are noted from the V-REP window. Also, the path length from the start point to goal point is measured with the help of measuring tape during equivalent experimental setup. Simultaneously, the respective time elapsed for traversing that path is recorded through stopwatch. As the authenticity of the proposed navigational controller is verified through a comparative analysis of the simulation and respective experimental results, a series of 30 different environmental scenarios are considered here for the investigation. Table II shows the path length covered by the robot from start point to goal point while avoiding the obstacles for 30 scenarios. The average deviation for path length between simulation and experimental results is found to be within 6%. Table III represents the respective time taken to cover the path by the robot for the same 30 scenarios as stated in Table II. Here, the average deviation for time elapsed between simulation and experimental results is also found to be within 6% which is well within the acceptable limit. This deviation in results arises due to the frictional resistance between wheel and floor, presence of backlash in gear system as well as any kind of slippage condition during real-time experimental analysis. But, in identical simulation exercises, there is least chance of frictional and data transmission losses due to its ideal condition.

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

Table II. Path length traversed by robot from start point to goal point while avoiding obstacles.

| No. of exercise | Path length (m) (simulation) | Path length (m) (experiment) | Deviation (%) | Average deviation (%) |
|---|---|---|---|---|
| 1 | 2.97 | 3.06 | 2.95 | 5.59 |
| 2 | 1.78 | 1.88 | 5.32 | |
| 3 | 3.39 | 3.69 | 8.14 | |
| 4 | 2.29 | 2.45 | 6.54 | |
| 5 | 2.31 | 2.54 | 9.06 | |
| 6 | 2.99 | 3.06 | 2.29 | |
| 7 | 3.08 | 3.18 | 3.15 | |
| 8 | 1.65 | 1.82 | 9.35 | |
| 9 | 2.41 | 2.65 | 9.06 | |
| 10 | 1.8 | 1.9 | 5.27 | |
| 11 | 1.6 | 1.73 | 7.52 | |
| 12 | 3.5 | 3.64 | 3.85 | |
| 13 | 3.28 | 3.59 | 8.64 | |
| 14 | 3.13 | 3.29 | 4.87 | |
| 15 | 2.85 | 3.12 | 8.66 | |
| 16 | 3.06 | 3.18 | 3.78 | |
| 17 | 2.09 | 2.17 | 3.69 | |
| 18 | 1.76 | 1.85 | 4.87 | |
| 19 | 2.67 | 2.79 | 4.31 | |
| 20 | 3.23 | 3.43 | 5.84 | |
| 21 | 2.49 | 2.56 | 2.74 | |
| 22 | 3.3 | 3.39 | 2.66 | |
| 23 | 1.9 | 2.08 | 8.66 | |
| 24 | 1.74 | 1.82 | 4.4 | |
| 25 | 3.01 | 3.11 | 3.22 | |
| 26 | 3.1 | 3.23 | 4.03 | |
| 27 | 2.51 | 2.71 | 7.39 | |
| 28 | 1.72 | 1.79 | 3.92 | |
| 29 | 2.41 | 2.65 | 9.06 | |
| 30 | 1.75 | 1.83 | 4.38 | |

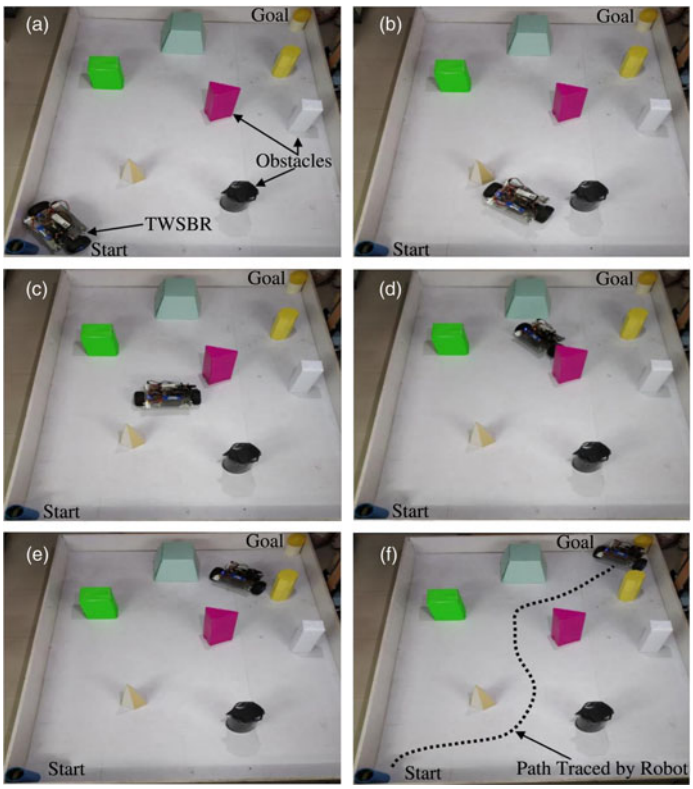


Fig. 10. Snapshots of experimental results showing (a) initial position, (b-e) intermediate steps of robot navigation from start to goal position and (f ) final position during robot motion in a complex cluttered environment.

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

Table III. Time taken by robot from start point to goal point while avoiding obstacles.

| No. of exercise | Time taken (ms) (simulation) | Time taken (ms) (experiment) | Deviation (%) | Average deviation (%) |
|---|---|---|---|---|
| 1 | 44,329 | 48,080 | 7.81 | 5.69 |
| 2 | 26,568 | 28,400 | 6.46 | |
| 3 | 50,598 | 55,597 | 9 | |
| 4 | 34,180 | 36,320 | 5.9 | |
| 5 | 34,478 | 36,879 | 6.52 | |
| 6 | 44,627 | 45,655 | 2.26 | |
| 7 | 45,971 | 47,006 | 2.21 | |
| 8 | 24,627 | 25,998 | 5.28 | |
| 9 | 35,971 | 37,552 | 4.22 | |
| 10 | 26,866 | 29,200 | 8 | |
| 11 | 23,881 | 25,295 | 5.6 | |
| 12 | 52,239 | 53,706 | 2.74 | |
| 13 | 48,956 | 50,954 | 3.93 | |
| 14 | 46,717 | 51,074 | 8.54 | |
| 15 | 42,538 | 45,095 | 5.68 | |
| 16 | 45,672 | 49,131 | 7.05 | |
| 17 | 31,195 | 32,448 | 3.87 | |
| 18 | 26,269 | 28,513 | 7.88 | |
| 19 | 39,851 | 41,160 | 3.19 | |
| 20 | 48,209 | 51,894 | 7.11 | |
| 21 | 37,165 | 38,779 | 4.17 | |
| 22 | 49,254 | 52,387 | 5.99 | |
| 23 | 28,359 | 30,759 | 7.81 | |
| 24 | 25,971 | 26,825 | 3.19 | |
| 25 | 44,926 | 48,759 | 7.87 | |
| 26 | 46,269 | 47,953 | 3.52 | |
| 27 | 37,463 | 38,867 | 3.62 | |
| 28 | 25,672 | 27,218 | 5.69 | |
| 29 | 35,971 | 38,825 | 7.36 | |
| 30 | 26,120 | 28,383 | 7.98 | |

## 5. Comparison with Existing Navigational Controllers

From the above simulation and experimental analysis, it is quite evident that by implementing the neural network integrated modified DAYANI algorithm, TWSBR is perfectly able to navigate avoiding obstacles in unknown cluttered environments. However, the efficacy of the proposed controller should be verified by comparing it with other existing navigational techniques. Therefore, path length traced by some existing authors implementing optimized fuzzy controller using genetic algorithm method and TLBO-based ANFIS technique is considered for comparison purpose. Figure 11(a) and (b) shows the comparison of the path obtained by Zhao et al.[43] using an optimized fuzzy controller through genetic algorithm method and our proposed modified Neural DAYANI method. Similarly, the path comparison of the proposed technique with TLBO-based ANFIS technique as obtained by Aouf et al.[44] is represented in Figure 12(a) and (b). The path lengths generated from both the existing techniques and the proposed technique are calculated and presented for comparison in Table IV. From the comparative analysis, it is clearly visible that the neural network integrated modified DAYANI approach can be chosen as a better alternative than the existing techniques with around 11% reduction in path length.

## 6. Conclusion

In the current analysis, a novel neural network integrated modified DAYANI approach has been analyzed and implemented for navigational path control of TWSBR in unknown environment. Here, a back-propagated neural network method has been deployed for determination of the intensity of various weight factors required to be used in the modified DAYANI method. These intensity weight

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

Table IV. Comparison of path length traced by Zhao et al.[43] and Aouf et al.[44] with path obtained by neural network integrated modified DAYANI approach by considering 1 cm = 1 unit.

| **Technique used** | **Path length (unit)** | **Deviation (%)** |
|---|---|---|
| Optimized fuzzy controller by genetic algorithm method, Zhao et al.[43] and Fig. 11(a) | 16.8 | 11.3 |
| Neural network integrated modified DAYANI approach, Fig. 11(b) | 14.9 | |
| TLBO-based ANFIS technique by Aouf et al.[44] and Fig. 12(a) | 9.5 | 12.76 |
| Neural network integrated modified DAYANI approach, Fig. 12(b) | 8.3 | |

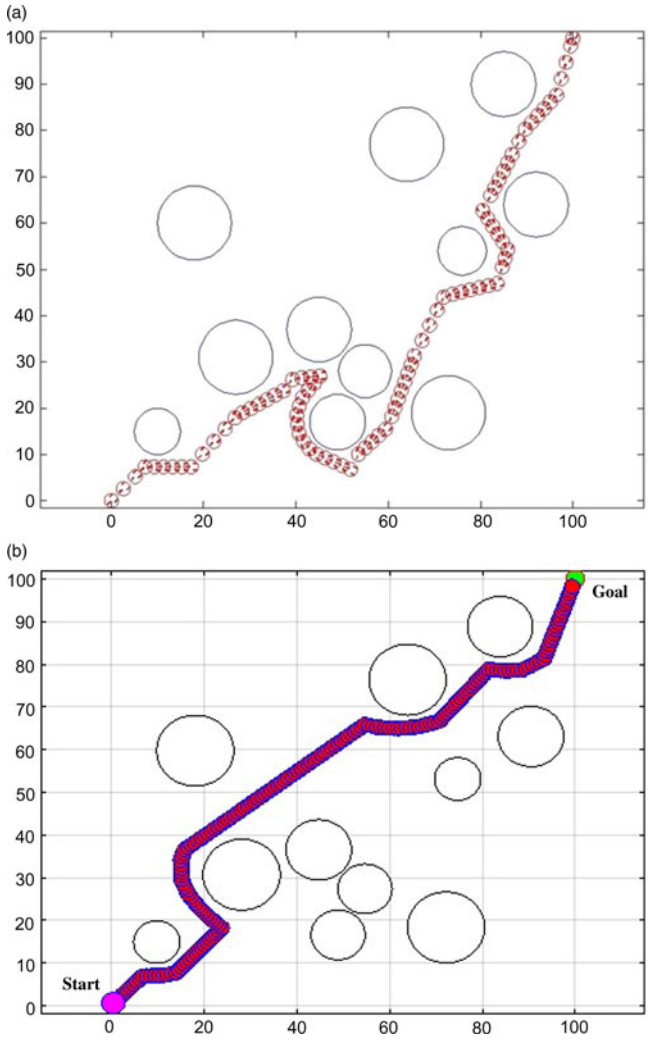


Fig. 11. (a) Path obtained by optimized fuzzy controller by genetic algorithm method by Zhao et al.[43] and (b) path obtained by neural network integrated modified DAYANI approach.

factors are calculated using the five-layered neural network considering number of visible intersecting obstacles along the goal direction, minimum visible front intersecting distances as obtained from the sensors, average of left side distances, minimum left side distance within the visible left side range of the robot, average of right side distances, minimum right side distance within the visible right side range of the robot and goal distance from the robot's probable position as input parameters.

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

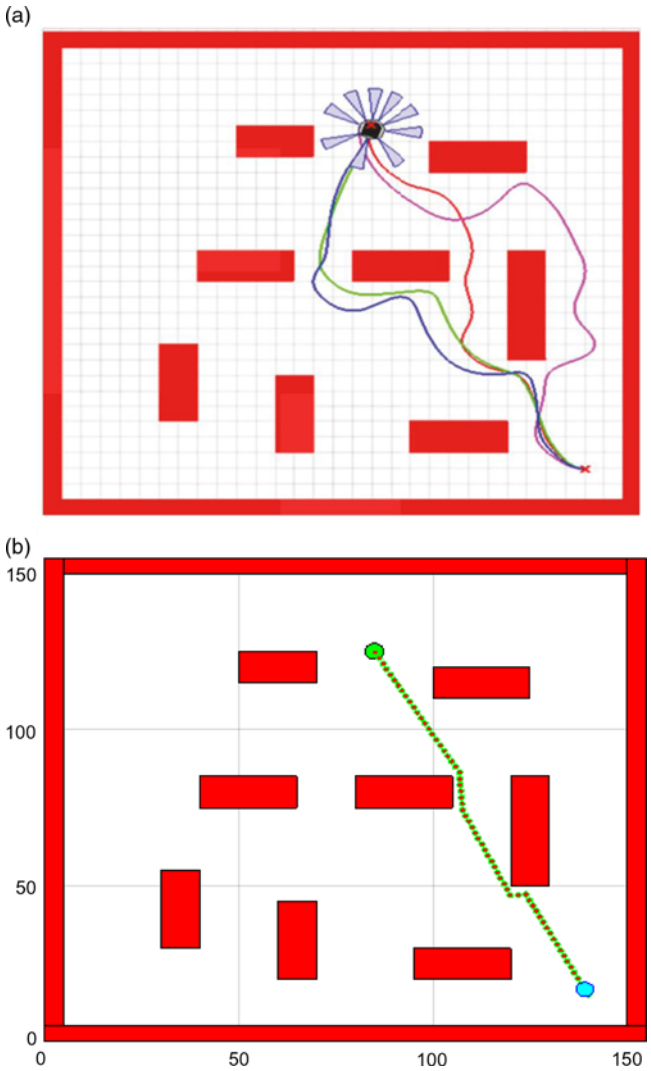


Fig. 12. (a) Path obtained by TLBO-based ANFIS technique by Aouf et al.[44] and (b) path obtained by neural network integrated modified DAYANI approach.

Comparisons are carried out for verifying the robustness of the proposed controller between simulation and experimental exercises and the deviation is found to be within 6%. In order to comprehend the effectiveness of the developed method, the results obtained from the proposed approach have been compared with the results acquired through both optimized fuzzy controller by genetic algorithm method and TLBO-based ANFIS technique in identical environmental scenario. This witnesses a better enhancement of results through our suggested controller than the existing methods by other authors. Thus, using the proposed neural network integrated modified DAYANI method, the TWSBR is able to find the optimal path from start point to goal point in a lesser time without colliding with the obstacles in between.

## References

1. P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *Int. J. Phy. Sci*. **7**(9), 1314–1320 (2012).
2. H. Zhang, J. Butzke and M. Likhachev, "Combining Global and Local Planning with Guarantees on Completeness," *IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, MN, USA (2012) pp. 4500–4506.
3. E. Masehian and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning-a chronological review," *World Acad. Sci. Eng. Technol*. **29**(1), 101–106 (2007).
4. D. R. Parhi and M. K. Singh, "Navigational strategies of mobile robots: A review," *Int. J. Autom. Control* **3**(2–3), 114–134 (2009).
5. J. C. Latombe, *Robot Motion Planning*, vol. 124 (Springer Science & Business Media, New York, 2012).

https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

6. S. Garrido, L. Moreno, M. Abderrahim and F. Martin, "Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China (2006) pp. 2376–2381.
7. H. Kaluđer, M. Brezak and I. Petroviæ, "A Visibility Graph Based Method for Path Planning in Dynamic Environments," *MIPRO, Proceedings of the 34th International Convention*, Opatija, Croatia (2011) pp. 717–721.
8. B. Dugarjav, S. G. Lee, T. B. Quang, K. W. Gwak and B. Lee, "Adaptive online cell decomposition with a laser range finder in unknown non-rectilinear environments," *Int. J. Precis. Eng. Manuf.* **18**(4), 487–495 (2017).
9. O. Montiel, R. Sepúlveda and U. Orozco-Rosas, "Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field," *J. Intell. Rob. Syst.* **79**(2), 237–257 (2015).
10. T. T. Mac, C. Copot, D. T. Tran and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Rob. Auton. Syst.* **86**, 13–28 (2016).
11. D. H. Kim, "Escaping route method for a trap situation in local path planning," *Int. J. Control Autom. Syst.* **7**(3), 495–500 (2009).
12. A. T. Ismail, A. Sheta and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *J. Comput. Sci.* **4**(4), 341–344 (2008).
13. R. Kala, A. Shukla and R. Tiwari, "Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness," *Neurocomputing* **74**(14–15), 2314–2335 (2011).
14. S. K. Pradhan, D. R. Parhi and A. K. Panda, "Fuzzy logic techniques for navigation of several mobile robots," *Appl. Soft Comput.* **9**(1), 290–304 (2009).
15. D. C. Rao, M. R. Kabat, P. K. Das and P. K. Jena, "Cooperative navigation planning of multiple mobile robots using improved krill herd," *Arabian J. Sci. Eng.* **43**(12), 7869–7891 (2018).
16. B. B. V. L. Deepak, D. R. Parhi and B. M. V. A. Raju, "Advance particle swarm optimization-based navigational controller for mobile robot," *Arabian J. Sci. Eng.* **39**(8), 6477–6487 (2014).
17. A. V. Savkin and H. Li, "A safe area search and map building algorithm for a wheeled mobile robot in complex unknown cluttered environments," *Robotica* **36**(1), 96–118 (2018).
18. A. Pandey and D. R. Parhi, "Optimum path planning of mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm," *Defence Technol.* **13**(1), 47–58 (2017).
19. P. K. Mohanty and D. R. Parhi, "A new hybrid optimization algorithm for multiple mobile robots navigation based on the CS-ANFIS approach," *Memetic Comput.* **7**(4), 255–273 (2015).
20. D. R. Parhi and S. Kundu, "A hybrid fuzzy controller for navigation of real mobile robot," *Int. J. Appl. Artif. Intell. Eng. Syst.* **3**(1), 169 (2011).
21. A. Kumar, P. B. Kumar and D. R. Parhi, "Intelligent navigation of humanoids in cluttered environments using regression analysis and genetic algorithm," *Arabian J. Sci. Eng.* **43**(12), 7655–7678 (2018).
22. P. B. Kumar, K. K. Pandey, C. Sahu, A. Chhotray and D. R. Parhi, " A Hybridized RA-APSO Approach for Humanoid Navigation," *IEEE Nirma University International Conference on Engineering (NUiCONE)*, Ahmedabad, India (2017) pp. 1–6.
23. D. R. Parhi and M. K. Singh, "Heuristic-rule-based hybrid neural network for navigation of a mobile robot," *Proc. Inst. Mech. Eng. Part B: J. Eng. Manuf.* **224**(7), 1103–1118 (2010).
24. Y. Bassil, "Neural network model for path-planning of robotic rover systems," arXiv preprint arXiv:1204.0183 (2012).
25. M. Al-Sagban and R. Dhaouadi, "Neural-Based Navigation of a Differential-Drive Mobile Robot," *12th IEEE International Conference on Control Automation Robotics & Vision (ICARCV)*, Guangzhou, China (2012) pp. 353–358.
26. C. Luo, J. Gao, X. Li, H. Mo and Q. Jiang, "Sensor-Based Autonomous Robot Navigation Under Unknown Environments with Grid Map Representation," *IEEE Symposium on Swarm Intelligence (SIS)*, Orlando, FL, USA (2014) pp. 1–7.
27. M. Boukens and A. Boukabou, "Design of an intelligent optimal neural network-based tracking controller for nonholonomic mobile robot systems," *Neurocomputing* 226, 46–57 (2017).
28. G. Farias, E. Fabregas, E. Peralta, H. Vargas, G. Hermosilla, G. Garcia and S. Dormido, "A neural network approach for building an obstacle detection model by fusion of proximity sensors data," *Sensors* 18(3), 683 (2018).
29. M. K. Singh and D. R. Parhi, "Path optimisation of a mobile robot using an artificial neural network controller," *Int. J. Syst. Sci.* 42(1), 107–120 (2011).
30. N. H. Singh and K. Thongam, "Mobile robot navigation using MLP-BP approaches in dynamic environments," *Arabian J. Sci. Eng.* **43**(12), 8013–8028 (2018).
31. D. R. Parhi and A. Chhotray, "Development and analysis of DAYANI arc contour intelligent technique for navigation of two-wheeled mobile robot," *Ind. Rob. Int. J.* **45**(5), 688–702 (2018).
32. R. P. M. Chan, K. A. Stol and C. R. Halkyard, "Review of modelling and control of two-wheeled robots," *Annu. Rev. Control* **37**(1), 89–103 (2013).
33. A. Maddahi, A. H. Shamekhi and A. Ghaffari, "A Lyapunov controller for self-balancing two-wheeled vehicles," *Robotica* **33**(1), 225–239 (2015).
34. H. W. Kim and S. Jung, "Control of a two-wheel robotic vehicle for personal transportation," *Robotica* **34**(5), 1186–1208 (2016).
35. T. Takei and R. Imamura, "Baggage transportation and navigation by a wheeled inverted pendulum mobile robot," *IEEE Trans. Ind. Electron.* **56**(10), 3985–3994 (2009).


https://doi.org/10.1017/S0263574718001558 Published online by Cambridge University Press

36. P. Shu-Hua, H. Cui and L. Deng-Hua, "Fuzzy Path Planning of Two-Wheeled Robot Optimized by Gold Mean," **In**: *Informatics in Control, Automation and Robotics* (Springer, Berlin, Heidelberg, 2011) pp. 477–484.
37. H. Mirzaeinejad and A. M. Shafei, "Modeling and trajectory tracking control of a two-wheeled mobile robot: Gibbs–Appell and prediction-based approaches," *Robotica* **36**(10), 1551–1570 (2018).
38. J. Lee and W. Park, "Probability-Based Optimal Path Planning for Two-Wheeled Mobile Robots," *ASME 2015 Dynamic Systems and Control Conference, 2015, American Society of Mechanical Engineers*, Columbus, OH, USA (2015) pp. V003T40A006-V003T40A006.
39. N. A. Ghani, L. Haur, T. Yon and F. Naim, "Dual mode navigation for two-wheeled robot," *World Acad. Sci. Eng. Technol*. **58**, 278–283 (2011).
40. A. Chhotray, M. K. Pradhan, K. K. Pandey and D. R. Parhi, "Kinematic Analysis of a Two-Wheeled Self-balancing Mobile Robot," *Proceedings of the International Conference on Signal, Networks, Computing, and Systems*, New Delhi, India (2016) pp. 87–93.
41. D. T. Pham and D. R. Parhi, "Navigation of multiple mobile robots using a neural network and a Petri Net model," *Robotica* **21**(1), 79–93 (2003).
42. S. Haykin, *Neural Networks: A Comprehensive Foundation* (Prentice Hall PTR, Upper Saddle River, NJ, USA, 1994).
43. R. Zhao, D. H. Lee and H. K. Lee, "Mobile robot navigation using optimized fuzzy controller by genetic algorithm," *Int. J. Fuzzy Logic Intell. Syst*. **15**(1), 12–19 (2015).
44. A. Aouf, L. Boussaid and A. Sakly, "TLBO-based adaptive neurofuzzy controller for mobile robot navigation in a strange environment," *Comput. Intell. Neurosci*. (2018). doi:10.1155/2018/3145436.