Intelligent Service Robotics (2019) 12:55–67
https://doi.org/10.1007/s11370-018-0260-2

ORIGINAL RESEARCH PAPER

Neural network-based approaches for mobile robot navigation in static and moving obstacles environments

Ngangbam Herojit Singh[1] · Khelchandra Thongam[1]

Received: 22 November 2017 / Accepted: 3 September 2018 / Published online: 10 September 2018

Abstract
Mobile robots can travel by acquiring the information using sensor-actuator control techniques from surrounding and perform several tasks. Due to the ability of traversing, mobile robots are used in different application for different places. In the field of robotic research, robot navigation is the fundamental problem and it is easier in static environment than dynamic environment. This paper presents a new method for generating a collision-free, near-optimal path and speed for a mobile robot in a dynamic environment containing moving and static obstacles using artificial neural network. For each robot motion, the workspace is divided into five equal segments. The multilayer perceptron (MLP) neural network is used to choose a collision-free segment and also controls the speed of the robot for each motion. Simulation results show that the method is efficient and gives near-optimal path reaching the target position of the mobile robot.

Keywords Mobile robot · Path planning · Dynamic environment · Artificial neural network · Obstacle avoidance · Collision-free path · Supervised learning · Multilayer perceptron

1 Introduction

In recent years, a lot of research is going on around the world to find a suitable navigation method for mobile robot without human interaction in static and dynamic environments. One of the major challenges of the autonomous navigation for mobile robots is the detection and obstacles avoidance during the robot navigation task. This problem can be solved by relating different methods or algorithms in order to attain best results.

A biologically inspired neural network in [34] is used to design for real-time navigation with obstacle avoidances of a mobile robot and a multi-joint robot manipulator in a non-stationary environment. Many researcher studies used soft computing algorithm to navigate the mobile robot in different environments. A collision-free path is constructed for moving robot among obstacles based on two neural networks [8]. The first neural network is used to determine the free space using sensor, and second is used to find safe direction for the next robot section of the path in the workspace while avoiding the nearest obstacles.

A reactive immune network (RIN) is designed in [12] for mobile robot navigation in unknown environments. In addition, an adaptive virtual target method is integrated to solve the local minima problem in navigation. In [25], an artificial neural network controller is designed for an autonomous mobile robot using a multilayer feed forward neural network, which enables the robot to navigate in a real-world dynamic environment. Using a technique based on utilization of neural networks and reinforcement learning in [27], a mobile robot learn to generate efficient navigation rules automatically without initial settings of rules by experts and constructed environments on its own.

In [7], three neural controllers are developed for robot navigation in urban environments. The first and second controllers of all the input units are fully connected with the hidden units. In the third controller, the hidden units are divided into three groups. The results show that neural controller with separated hidden neurons has a fast response to sensory input.

A neural network-based camera calibration method is presented in [4] for the global localization of mobile robots with monocular vision. The camera is calibrated using the neural network-based method, and monocular vision is used to initialize and recalibrate the robot position. In [2], they have

✉ Ngangbam Herojit Singh
herojitng@gmail.com

Khelchandra Thongam
thongam@gmail.com

1 National Institute of Technology Manipur, Imphal, India

surveyed the mobile robot navigation using neural networks and developments in the last few years of the neural networks with applications to mobile robot navigation. In [36], they have developed an algorithm called escaping algorithm (EA) which does not need any pre-knowledge information about dynamic obstacles. It used Kalman filter to predict the motion of dynamic objects and combines them with potential field approach to navigate safely in dynamic environment. An online navigation technique in [10] for a wheeled mobile robot (WMR) is developed in an unknown dynamic environment using fuzzy logic techniques. Tracking fuzzy logic control (TFLC) and obstacle avoidance fuzzy logic control (OAFLC) are combined to move the mobile robot to the target along a collision-free path. The algorithm starts with the former one, and if sensors detect any obstacles in the front of the robot, then the algorithm switches over to the later one. But time consumption is more, since it always used fuzzy computation if there is free path.

In [17], they have presented two navigation algorithms for wheeled mobile robot in unknown environments. The first algorithm based on geometrical-based technique determines the set of all possible collision-free steering angles and selects the steering angle that is nearest to the target with widest gap. The second algorithm based on neural network-based technique generates an optimized path by using the objective function which minimizes the traveled distance to the goal position while avoiding obstacles.

A hybrid path planning of mobile robot in cluttered environments is presented in [21] and [37]. An adaptive neuro-fuzzy inference system is constructed for path planning of mobile robot in [15,24,29] and [31] for different environments.

In [26,33] and [5], they have proposed the mobile robot navigation technique using fuzzy logic controllers in static and dynamic environments. The overall concepts are same, but the range of the memberships function is different.

An electrostatic potential filed methods are presented in [11,14,18] and [38] for navigation of a group of robot and single robot in structured and unstructured environments. They have compared with the other existing methods for navigation of mobile robot. But still an improvement is required for dynamic environments with respect to time and path length.

An hybrid optimal method of mobile robot navigation is developed by using fuzzy logic and genetic algorithm in [6] in order to optimize the path length using genetic algorithm which is generated by the fuzzy system.

Improved fuzzy logic techniques are developed in [3,13,16,19,22] and [30] for static and highly cluttered environments. But still there is a major drawback in dynamic environments.

A comparative survey is presented in [1] and [35] for different soft computing methods and heuristic methods in robot path planning. In different methods, fuzzy logic and neural network-based methods are more advantages than the other heuristic methods.

As the above existing methods have more computational time and path length, we introduce a method for generating a collision-free, near-optimal path and speed for a mobile robot in a dynamic environment containing moving and static obstacles using artificial neural network. Multilayer perceptron (MLP) is used to choose a collision-free segment from a set of five segments, and the network also controls the speed of the robot.

The paper is organized as follows: In Sect. 2, we provide a preliminaries to prepare for the rest of work. Section 3 gives the motion planning algorithm of a mobile robot. Experimental results, evaluation and discussion and comparison are presented in Sects. 4, 5 and 6, respectively. Finally, conclusions are given out in Sect. 7.

2 Preliminaries

2.1 Problem definition

The formal definition of our motion planning problem is as follows. Let R be a robot moving in a two-dimensional workspace W. The configuration of an arbitrary object is a specification of the position of every point on it relative to a fixed reference frame. Let FR and FW be Cartesian frame embedded in R and W, respectively. So, a configuration of R is a specification of the position (x, y) and orientation of FA with respect to FW. We use the notation $R(x)$ to refer to robot R configured at x in W. Let there be n moving obstacles $O_1, O_2, \ldots, O_n$ which state at time t is denoted by $O(t)$. All the obstacles have a maximal velocity given by $v_1, v_2, \ldots, v_n$. The robot has a maximal velocity V which is larger than each of the maximal velocities of the obstacles. We do not assume any knowledge of the velocities and directions of motion of the moving obstacles, other than that they have a maximal velocity. Let Ω be a list of adjacent points that will represent the path of the robot. Let $s, g \in \Omega$ be the start and goal configuration of the robot, respectively, and let t_0 be the start time. The task is to compute a path $\Pi : [t_0, T] \rightarrow \Omega$, such that $\Pi(t_0) = s$ and $\Pi(T) = g$, and there is no collision with the moving obstacles and the robot, i.e., $\forall(t \in [t_0, T]::R(\Pi(t)) \cap O(t) = \varphi)$. T is the arrival time of the robot R at its goal configuration.

2.2 Artificial neural networks

Our main objective is to find a collision-free path of a robot system from a given initial position to some goal position. The environment is a two-dimensional space with obstacles. The neural network that we used is the multilayer perceptron (MLP). The structure of the neural network is given in Fig. 1.

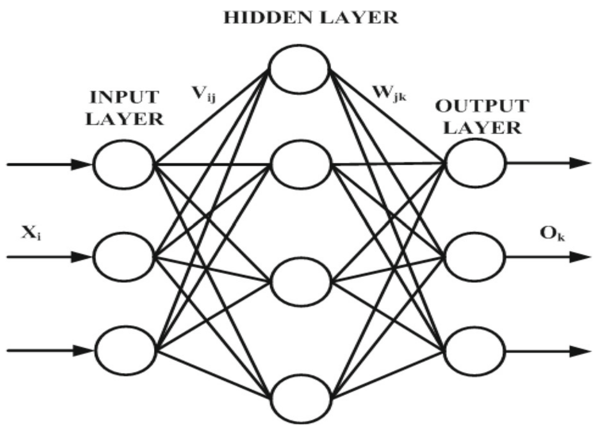

Fig. 1 Structure of MLP network

The network contains three layers: input, hidden and output. The layers are connected by synaptic weights. The learning of the network is realized by back-propagation (BP) algorithm. The BP algorithm is based on the error-correction principle. The parameters that are used during the learning process are given:

i. x_i : The ith input
ii. y_j : The output of the jth hidden neuron
iii. O_k : The output of the kth output neuron
iv. d_k : The desired output
v. V_{ji} : The weight from the ith input to the jth hidden neuron
vi. W_{kj} : The weight from the jth hidden neuron to the kth output neuron
vii. η : The learning rate

There are I inputs, J hidden neurons and K output neurons. The weights of the hidden layer are updated using the equation:

$$V_{ji}(n+1) = V_{ji}(n) + \eta \times \delta_j(n) \times x_i(n) \tag{1}$$

δ_j is the error signal produced by the jth hidden neuron.

The weights of the output layer are updated using the equation:

$$W_{kj}(n+1) = W_{kj}(n) + \eta \times \delta_k(n) \times y_j(n) \tag{2}$$

δ_k is the error signal produced by the kth output neuron.

$$\delta_k(n+1) = (d_k - O_k) \times (1 - O_k) \times O_k \tag{3}$$

Using δ_k, we can calculate δ_j as follows:

$$\delta_j = (1 - y_j) \times y_j \times \Sigma_{k=0}^{k} \delta_k \times W_{jk} \tag{4}$$

So, for any input, we find the output of the hidden neurons and then the output of the output layer neurons. The outputs in each layer are computed using the sigmoid function. The weights of each hidden and output layer neurons are updated using the above equations. The error signals of the hidden neurons are back-propagated from the output layer to the hidden layer. This process is repeated for the next input–output pattern and so on until the error is below a pre-specified threshold. We used the minimization of the squared error cost function:

$$E = \frac{1}{2} \Sigma_{k=0}^{k} (d_k - O_k)^2 \tag{5}$$

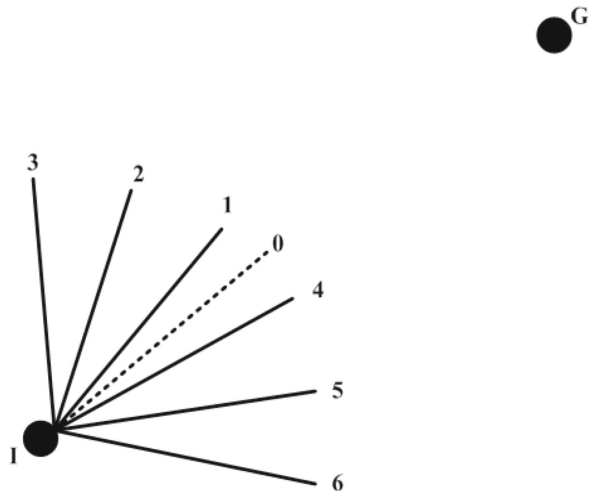

Fig. 2 Robot line of sight segment to destination

3 The motion planning algorithm

The main idea of the proposed model is to generate a collision-free, near-optimal path and speed for a mobile robot in a dynamic environment containing moving and static obstacles using artificial neural network. For each robot motion, the workspace is divided into 5 segments each of 30 degree. As in Fig. 2, the robot at positions I and G is the global destination position. 2I3, 1I2, 4I1, 5I4 and 6I5 are the five segments. I0 is perpendicular to global destination position G. Robot will choose a collision-free segment and will move ahead a distance of D=20 with a speed controlled by MLP neural network. The collision-free segment will also be computed by MLP.

As in Fig. 3, the robot is at position I. O1, O2 and O4 are static obstacles. O3 and O5 are moving obstacles.

Definition 1 (*Critical obstacle*)
The obstacle reaching a segment taking minimum time than other moving obstacles is chosen as the critical obstacle for the segment for that motion.

Input to the MLP network is the time taken by critical obstacle to reach its segment if the segment is not blocked by static or moving obstacles at that time t1. If blocked, then input value of that segment is 0.

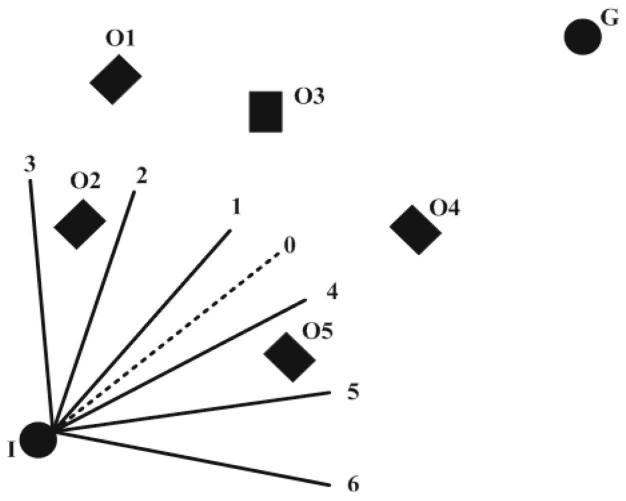

Fig. 3 Robot segment with obstacles

Output of the network is the time taken by the robot to reach its local destination if the segment is not blocked by static or moving obstacles at that time t1. If blocked, then output value of that segment is 0.

3.1 Input to MLP

The size of each input samples/patterns is 5 because of 5 numbers of segments. Each five value from left to right in the input relates to the 5 segments starting from left to right according to current robot position. The input is coded as follows:

i. A value of 0 will mean that the particular segment is occupied by static or moving obstacles at time t1 when the robot is at a particular position.
ii. Any value greater than 0 in the input will be the time taken by the critical obstacle to reach its segment.
iii. If the segment is not blocked by static or moving obstacles and there is no critical obstacle for that segment, then the input value will be 1.5.

3.2 Output from MLP

The size of each output patterns is 5 because of 5 numbers of segments and input size. The input–output mapping is as follows:

i. If the input value is 0 which means block by obstacle, then the corresponding output value is 0 which means robot motion is not possible in that segment.
ii. If the input value is from 0.1 to 0.5, then the corresponding output value is 0 which means robot motion is not possible in that segment because of moving obstacle.
iii. If the input value is from 0.6 to 1.0, then the corresponding output value is 0.5 which means the robot will take time = 0.5 to reach the local destination in that particular segment taking the middle path of the segment to avoid the moving obstacle.
iv. If the input value is greater than 1.0, then the corresponding output value will be 1 which means the robot will take time = 1.0 to reach the local destination in that particular segment taking the middle path of the segment to avoid the moving obstacle is exist.

The neural network will learn a set of target outputs (desired outputs) for a given set of input patterns. The output at each layer *i.e.* hidden and output layers is determined using the sigmoid function.

Table 1 Connection weights in W_{kj}

– 4.798507	4.436496	– 3.170834	– 3.566869	– 0.000353	– 3.089767	2.781051	1.639151	0.660623
0.003215	13.061621	– 11.303281	0.565211	– 3.783132	– 6.271729	– 4.758735	– 5.344402	0.004594
– 0.005463	– 0.632282	11.273642	– 12.966937	– 0.840588	0.561634	– 2.499261		
3.718968	4.708266	15.655665	– 4.023282	0.000099	– 7.165201	2.522545	15.124515	2.827022
0.005205	– 1.004524	– 5.756383	– 0.819283	– 4.526511	– 0.622096	– 0.786652	– 3.877044	– 0.005993
– 0.000804	– 0.448951	– 1.044669	0.468633	0.616444	– 2.127601	6.549060		
5.812205	8.867767	– 2.487811	– 11.666871	– 7.223574	– 7.415298	1.248051	5.276962	– 5.648685
18.699631	– 3.713130	– 5.959953	– 4.249684	– 5.626977	– 8.564949	– 14.123363	– 9.243060	14.527835
6.335071	1.625626	– 2.629568	– 1.981676	9.021267	2.865631	– 2.546922		
0.921611	5.783584	5.528798	– 5.809571	0.003279	– 3.439252	0.070215	5.000390	12.560915
0.004049	5.838228	– 6.355222	7.171010	– 9.108553	– 4.465524	4.444562	– 8.159965	– 0.005252
14.631222	– 15.389951	0.601364	– 17.913308	– 3.874958	12.390359	4.878840		
– 0.206452	– 0.624876	0.692462	– 3.696295	0.001123	– 1.940337	18.717489	1.699700	2.092247
0.001760	3.457083	3.180747	– 6.133540	– 17.228736	– 1.997152	4.660716	– 0.168702	– 0.006454
0.001754	2.021208	0.163855	0.470052	– 1.542114	– 4.015380	– 1.113645		

Table 2 Connection weights in V_{ji}

– 2.960963	9.309544	– 0.000328	1.871326	– 1.371737	9.932809
1.897830	0.365330	– 0.004168	20.358647	– 1.661227	20.658612
0.674744	18.862764	0.002098	– 0.594260	0.181480	20.424767
– 0.144433	0.202799	0.002985	– 4.859000	9.627889	6.220749
4.885847	3.969082	5.154472	– 8.934990	– 8.864817	3.252217
– 1.111842	14.606621	0.002298	0.725560	0.497880	11.981329
0.013023	0.468516	0.001723	0.375066	24.986440	27.986166
0.460605	22.393354	– 0.000807	– 0.867219	– 1.171882	11.859664
0.373515	– 7.807040	0.002372	7.672275	– 7.867049	0.201146
– 1.717224	– 0.713717	17.646525	– 1.015787	– 2.067355	8.365713
21.922102	0.722236	0.002551	– 0.015588	0.176944	23.909542
1.595880	– 3.999070	0.000406	1.021887	2.929893	8.550035
– 2.528985	1.538253	0.000461	4.045580	2.752693	0.478713
0.758141	0.232205	– 0.001458	– 0.885308	– 23.740708	– 12.263458
14.823890	0.348736	0.004977	– 0.255074	0.588553	11.561733
– 11.572954	– 1.115629	– 0.003023	– 0.880008	9.679857	– 13.851378
– 0.731401	– 0.563298	0.002357	20.416877	1.945459	16.723965
0.751574	1.074143	23.568633	– 1.029733	0.513208	26.027098
– 3.022009	– 1.131386	8.276764	10.290194	– 1.758658	– 3.721038
10.268509	– 0.588880	0.000626	– 17.497793	1.603037	– 9.104362
9.008677	– 1.072264	0.008501	0.677365	– 0.343723	4.414005
– 22.514168	– 0.318998	0.004138	1.789523	1.689173	– 10.820963
5.091151	– 4.301495	0.001929	3.704013	– 2.491043	1.356041
– 0.880454	0.232218	0.002788	16.417970	– 0.256866	17.892988

During training, the difference between the actual output and the desired output is minimized by adjusting the connecting weights using Eqs. (1) and (2).

Finally in testing/real operation, the robot chooses a segment and a speed to move ahead toward the goal according to the output of the network. The MLP will be operated for each robot motion to move a distance of $D = 20$ to reach its local destination.

If at least one of the network outputs more than 0.5 or 1.0 or both (meaning more than one segment is safe to move ahead), then the robot will choose the segment which is nearest to the global destination position.

4 Simulation results

In our application, the size of the robot is taken as 4×3 square unit (pixel). The results were obtained from a machine with Intel(R) Core(TM) i5 CPU with 3.20 GHz–3.33 GHz, 4 GB RAM, running Microsoft Visual C++ 2005 with OpenGL under Windows 7 Ultimate, 32-bit operating system.

Tables 1 and 2 show the connection weights in W_{kj} and V_{ji}, respectively. Figures 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18 and 19 show the robot motion in the same dynamic environment (first environment), where *I* is the initial position of the robot and *G* is the goal position. The black rectangular boxes represent the obstacles. Obstacles labeled with *O1*, *O2*, *O3* and *O4* are moving obstacles, and unlabeled ones are static obstacles. The *x*- and *y*-axes of the robot workspace range from – 120 to 120. The horizontal axis is the *x*-axis, and the vertical one is the *y*-axis. Similarly, Figs. 20–26 show robot motion in another dynamic environment (second environment).

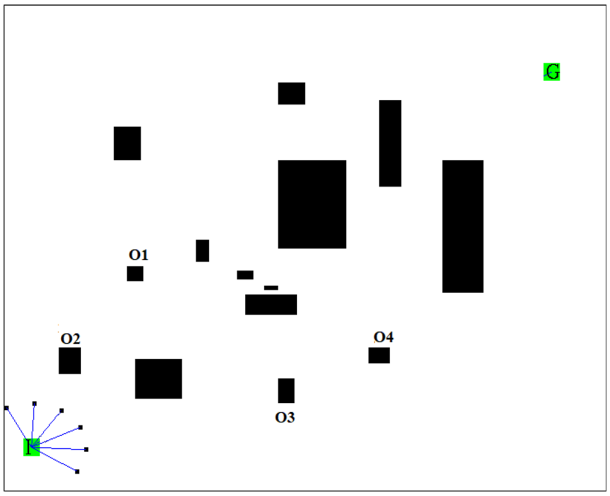

Fig. 4 Initial positions and segments for the first motion

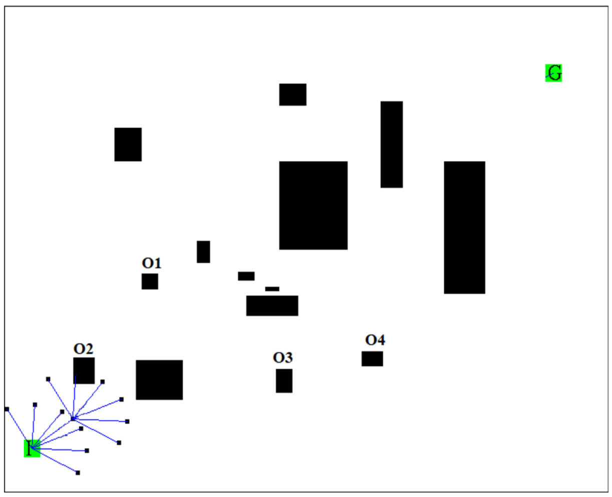

Fig. 5 First motion and segments for second motion

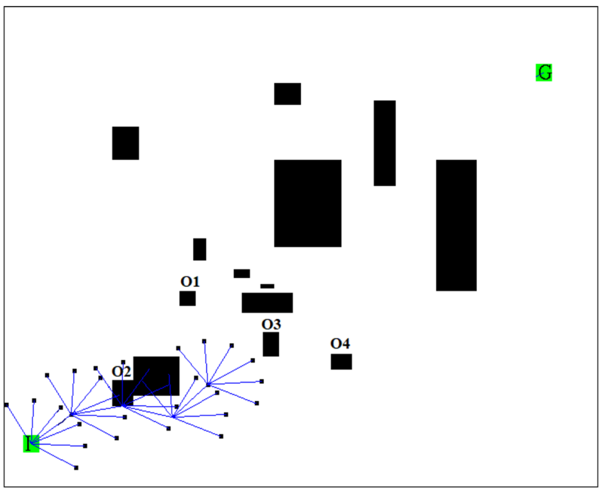

Fig. 8 Fourth motion and segments for fifth motion

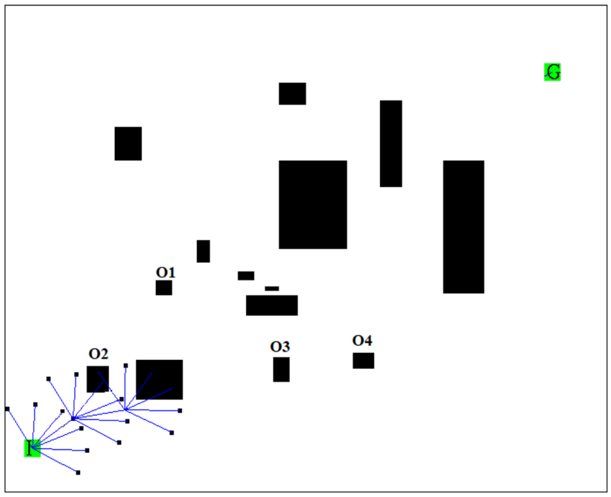

Fig. 6 Second motion and segments for third motion

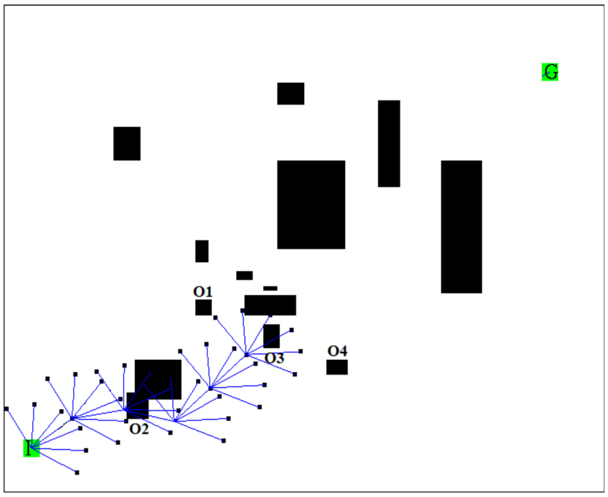

Fig. 9 Fifth motion and segments for sixth motion

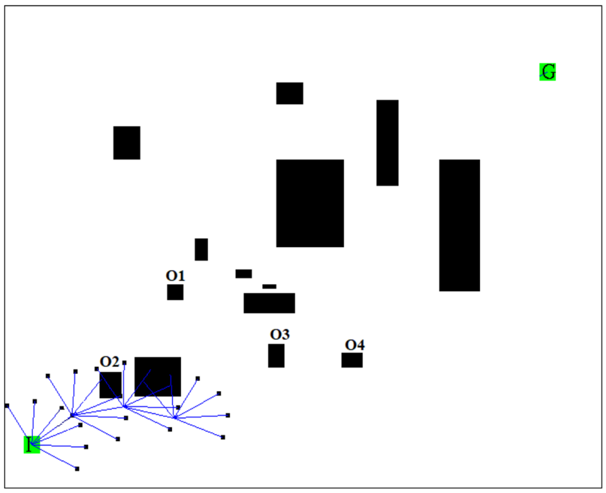

Fig. 7 Third motion and segments for fourth motion

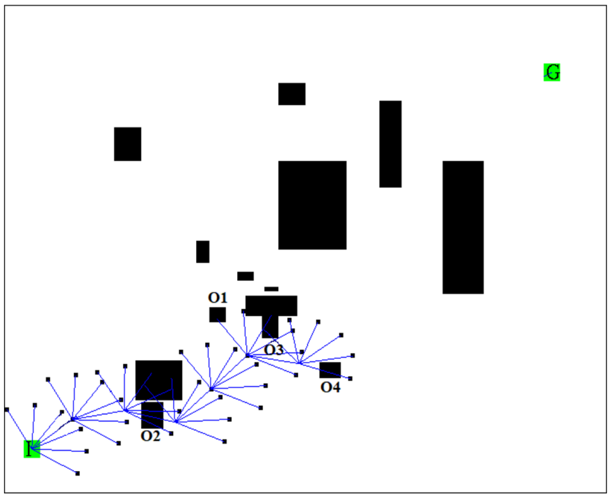

Fig. 10 Sixth motion and segments for seventh motion

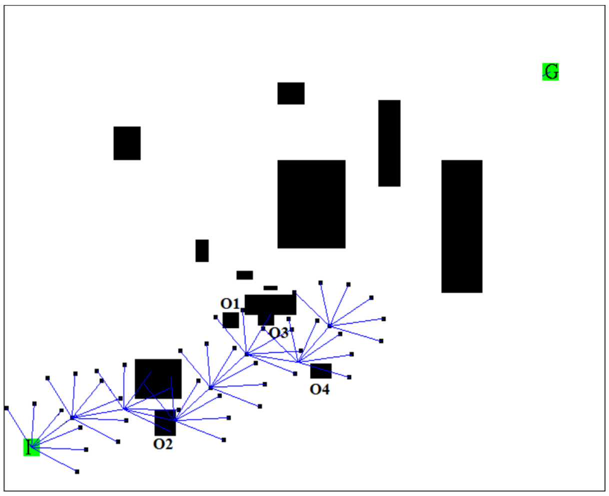

Fig. 11 Seventh motion and segments for eighth motion

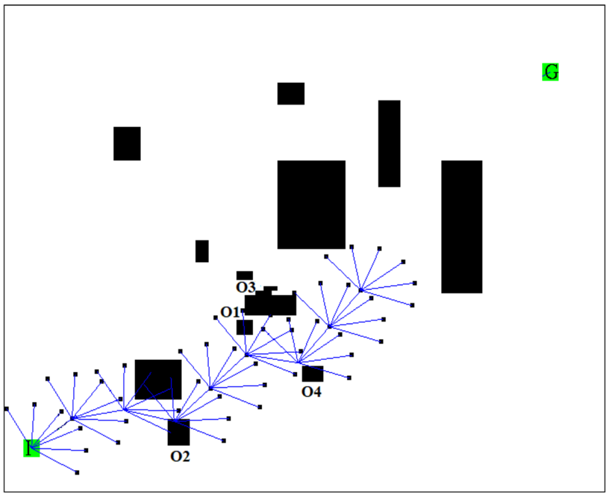

Fig. 12 Eighth motion and segments for ninth motion

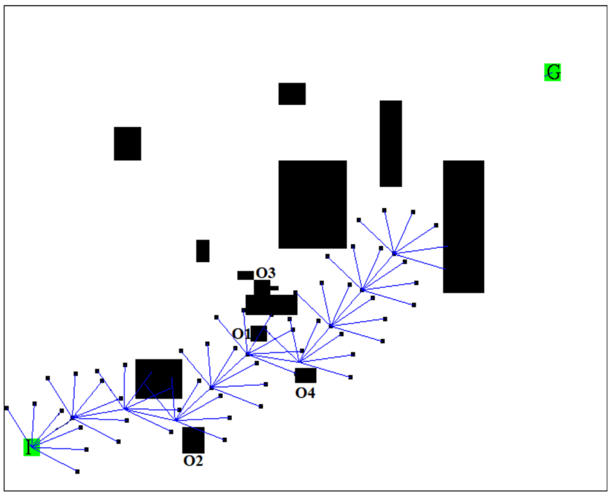

Fig. 13 Ninth motion and segments for tenth motion

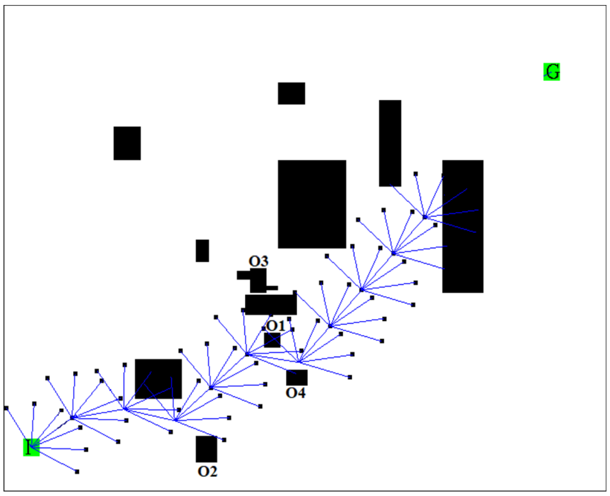

Fig. 14 Tenth motion and segments for eleventh motion

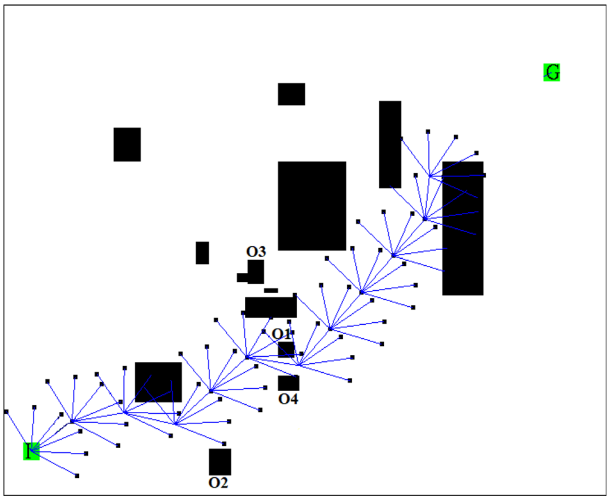

Fig. 15 Eleventh motion and segments for twelfth motion

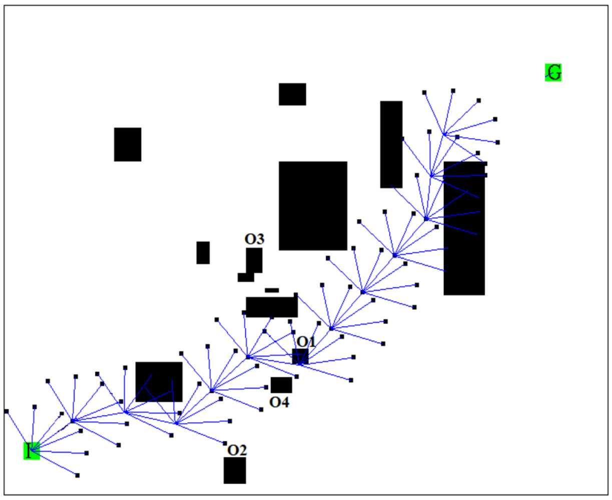

Fig. 16 Twelfth motion and segments for thirteenth motion

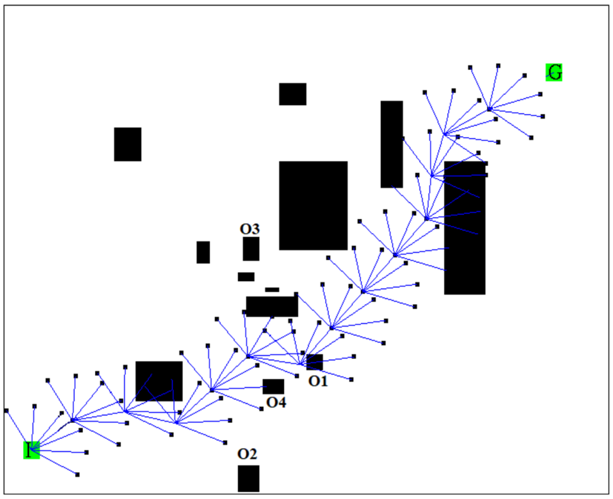

Fig. 17 Thirteenth motion and segments for fourteenth motion

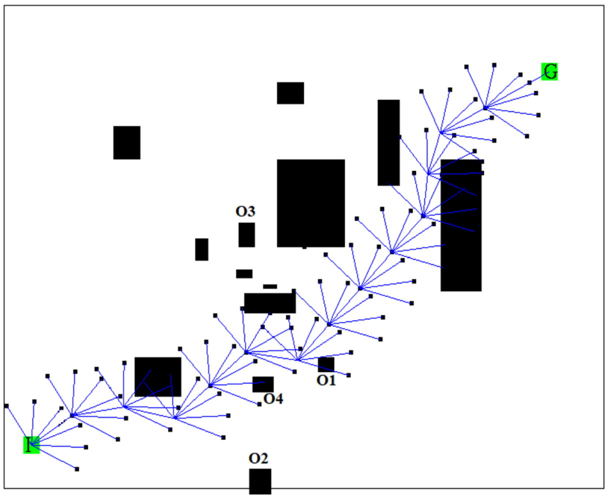

Fig. 18 Robot reaching the goal position

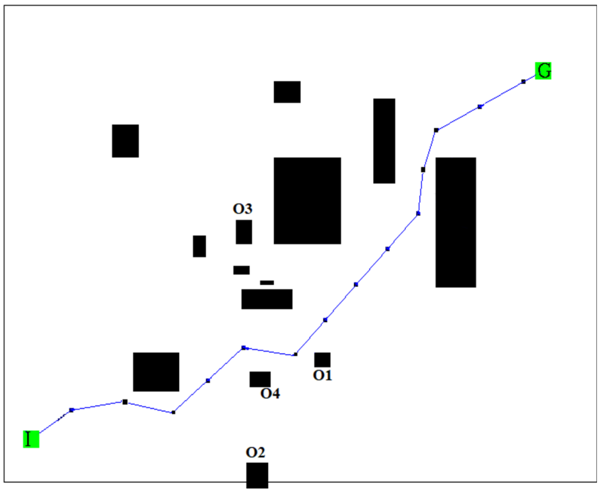

Fig. 19 Robot reaching the goal position without unused paths

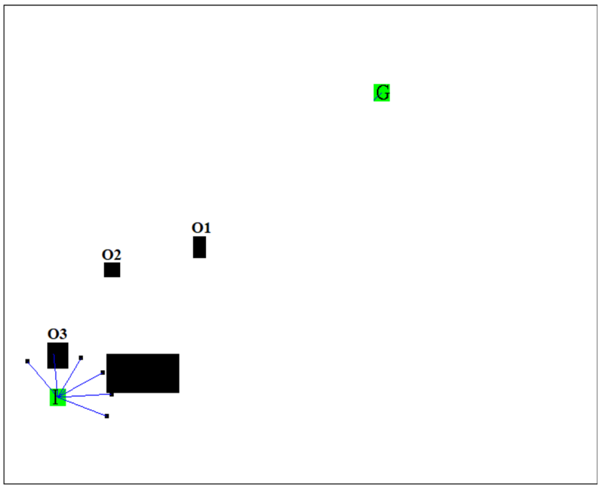

Fig. 20 Initial positions and segments for the first motion

5 Evaluation and discussion

In our MLP network, the size of each input neuron $I = 6$ input (including -1 the dummy input), number of hidden neurons $J = 25$ and the number of output neurons $K = 5$.

5.1 Computation of path in the first environment

In Figs. 4–19, the initial position I of the robot is $x = -100$ and $y = -90$ and the goal position G of the robot is $x = 90$, $y = 80$.

O1 is at the initial position with x-axis parameter $x1 = -59$, $x2 = -65$ and y-axis parameter $y1 = -15$, $y2 = -8$. O2 is at the initial position with x-axis parameter $x1 = -90$, $x2 = -82$ and y-axis parameter $y1 = -57$, $y2 = -45$. O3 is at the initial position with x-axis parameter $x1 = -10$, $x2 = -4$ and y-axis parameter $y1 = -70$, $y2 = -59$. O4 is at the initial position with x-axis parameter $x1 = 23$, $x2 = 31$ and y-axis parameter $y1 = -52$, $y2 = -45$.

O1 covers a distance of 5 units in the positive x direction and 3 units in negative y direction in 1 s. O2 moves a distance of 5 units in positive x direction and 4 units in negative y direction in 1 s. O3 moves a distance of 1 unit in negative x direction and 5 units in positive y direction in 1 s. O4 moves a distance of 3 units in negative x direction and 1 unit in negative y direction in 1 s.

Left most segment, we call it Segment1, and next as Segment2 till the right most one as Segment5.

i. Initial position, I of the mobile robot and moving obstacles is shown in Fig. 4. The segments for the first motion are also shown in the figure.
ii. In Fig. 5, the robot chooses Segment3 for the **first motion**. There is no critical obstacle for all the segments.

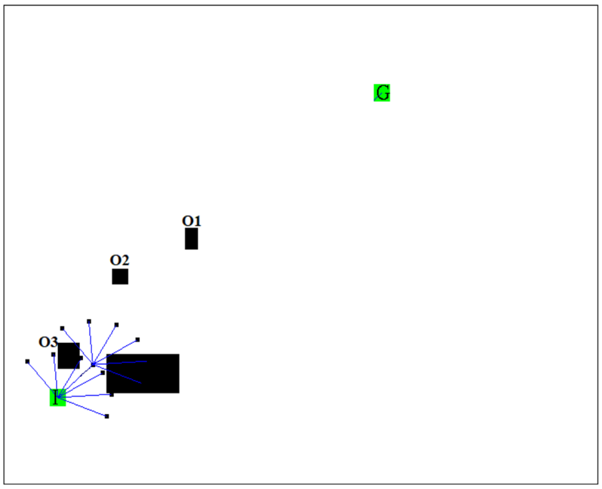

Fig. 21 First motion and segments for second motion

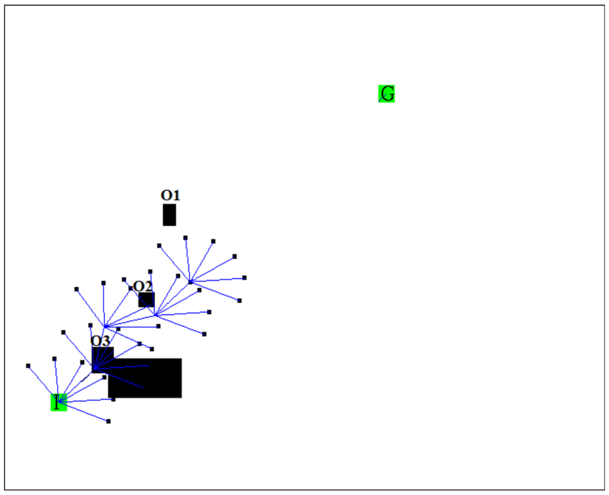

Fig. 24 Fourth motion and segments for fifth motion

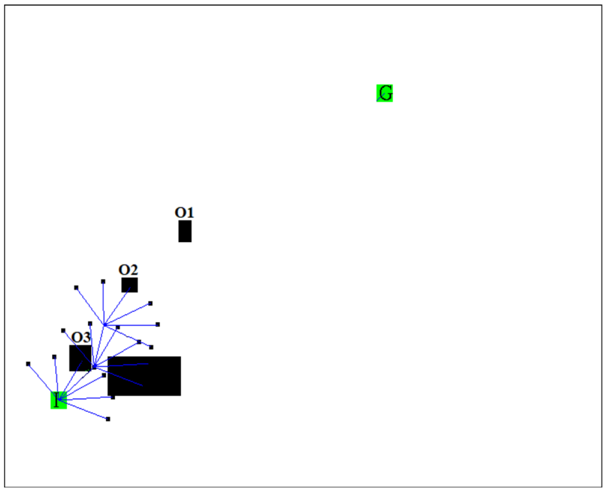

Fig. 22 Second motion and segments for third motion

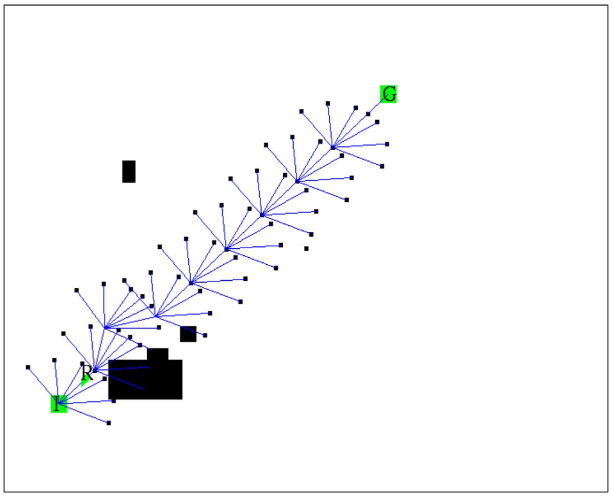

Fig. 25 Robot reaching the goal position

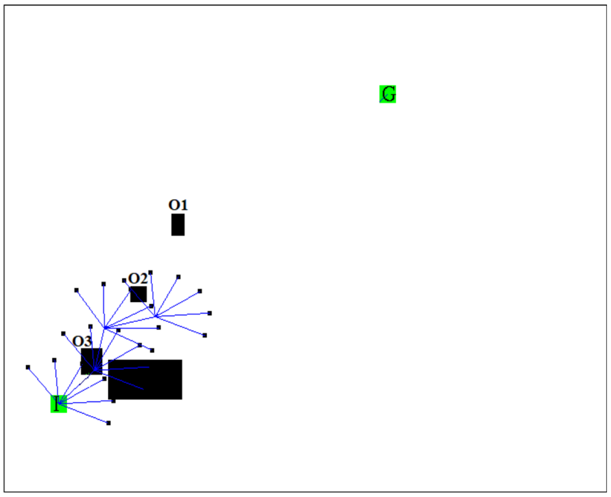

Fig. 23 Third motion and segments for fourth motion

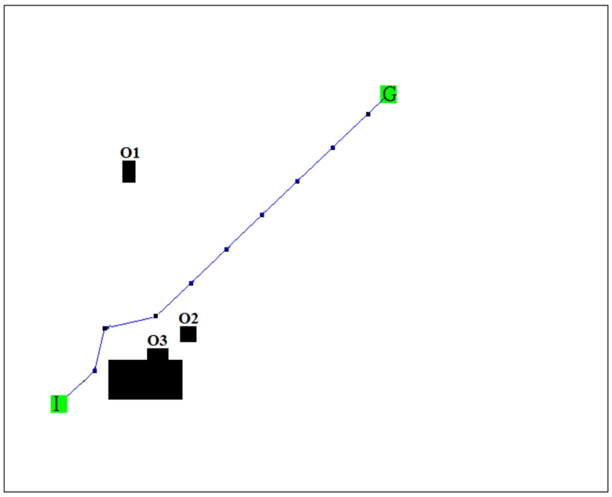

Fig. 26 Robot reaching the goal position without unused paths

The input to the MLP network for this motion is

$$\{1.5, 1.5, 1.5, 1.5, 1.5\}$$

The output is

$$\{0.999567, 0.999999, 0.999984, 1.000000, 0.999996\}$$

The segments for second motion are also shown in the figure.

iii. In Fig. 6, the robot chooses Segment4 for the **second motion**. O2 is the critical obstacle for Segment3, and it reaches the segment in 0.4 s for the second motion. There is no critical obstacle for other segments. Segment1 and Segment2 are blocked by O2 just before the second motion. The input to the MLP network for this motion is

$$\{0, 0, 0.4, 1.5, 1.5\}$$

The output is

$$\{0.000000, 0.000247, 0.000002, 0.999988, 0.999989\}$$

The segments for third motion are also shown in the figure.

iv. In Fig. 7, the robot chooses Segment5 for the **third motion**. There is no critical obstacle for all the segments. Segment1 is blocked by O2 just before the third motion. Segment2, Segment3 and Segment4 are blocked by static obstacle. The input to the MLP network for this motion is

$$\{0, 0, 0, 0, 1.5\}$$

The output is

$$\{0.000000, 0.000061, 0.000000, 0.000000, 1.000000\}$$

The segments for fourth motion are also shown in the figure.

v. In Fig. 8, the robot chooses Segment3 for the **fourth motion**. There is no critical obstacle for all the segments. Segment1 and Segment2 are blocked by static obstacle. The input to the MLP network for this motion is

$$\{0, 0, 1.5, 1.5, 1.5\}$$

The output is

$$\{0.000000, 0.000246, 1.000000, 0.999988, 0.999989\}$$

The segments for fifth motion are also shown in the figure.

vi. In Fig. 9, the robot chooses Segment3 for the **fifth motion**. There is no critical obstacle for all the segments, and no segment is blocked by obstacle. The input to the MLP network for this motion is

$$\{1.5, 1.5, 1.5, 1.5, 1.5\}$$

The output is

$$\{0.999567, 0.999999, 0.999984, 1.000000, 0.999996\}$$

The segments for sixth motion are also shown in the figure.

vii. In Fig. 10, the robot chooses Segment5 for the **sixth motion**. O1 is the critical obstacle for Segment1, and it reaches the segment in 0.3 s for the second motion. There is no critical obstacle for other segments. Segment3 and Segment4 are blocked by O2 just before sixth motion, and Segment2 is blocked by static obstacle. The input to the MLP network for this motion is

$$\{0.3, 0, 0, 0, 1.5\}$$

The output is

$$\{0.000000, 0.000054, 0.000000, 0.000000, 1.000000\}$$

The segments for seventh motion are also shown in the figure.

viii. In Fig. 11, the robot chooses Segment3 for the **seventh motion**. There is no critical obstacle for other segments. Segment1 and Segment5 are blocked by O3 and O4, respectively, just before seventh motion. The input to the MLP network for this motion is

$$\{0, 1.5, 1.5, 1.5, 0\}$$

The output is

$$\{0.000000, 1.000000, 1.000000, 1.000000, 0.000000\}$$

The segments for seventh motion are also shown in the figure.

ix. In Fig. 17, the robot chooses Segment3 for the **thirteenth motion**. There is no critical obstacle for all the segments. The input to the MLP network for this motion is

$$\{1.5, 1.5, 1.5, 1.5, 1.5\}$$

The output is

$$\{0.999567, 0.999999, 0.999984, 1.000000, 0.999996\}$$

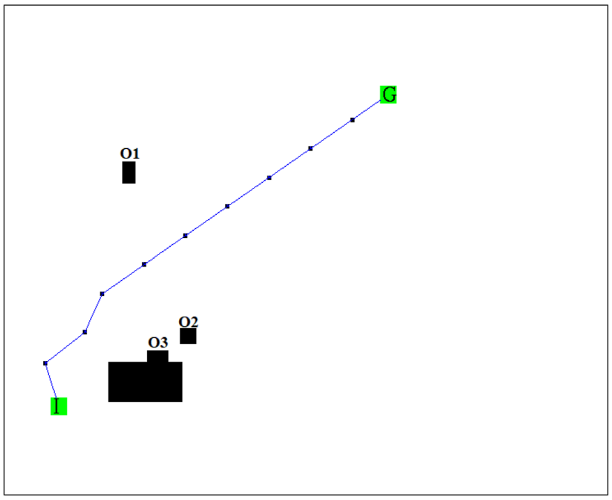

Fig. 27 Robot reaching the goal position for GA-Fuzzy in [32]

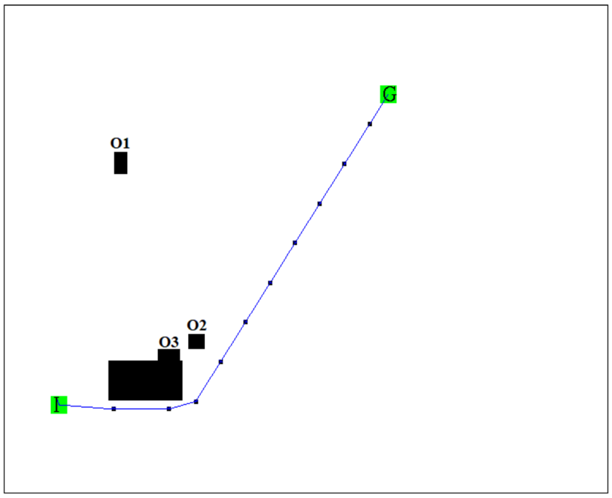

Fig. 28 Robot reaching the goal position for Fuzzy-WDO in [28]

The segments for the fourteenth motion are also shown in the figure.

x. In Fig. 18, the robot reaches the global destination.

5.2 Computation of path in the second environment

In Figs. 20–26, the initial position I of the robot is $x = -90$, $y = -70$, and the goal position G of the robot is $x = 30$, $y = 70$.

O1 is at the initial position with x-axis parameter $x1 = -35$, $x2 = -40$ and y-axis parameter $y1 = 4$, $y2 = -6$. O2 is at the initial position with x-axis parameter $x1 = -67$, $x2 = -73$ and y-axis parameter $y1 = -15$, $y2 = -8$. O3 is at the initial position with x-axis parameter $x1 = -94$, $x2 = -86$ and y-axis parameter $y1 = -57$, $y2 = -45$.

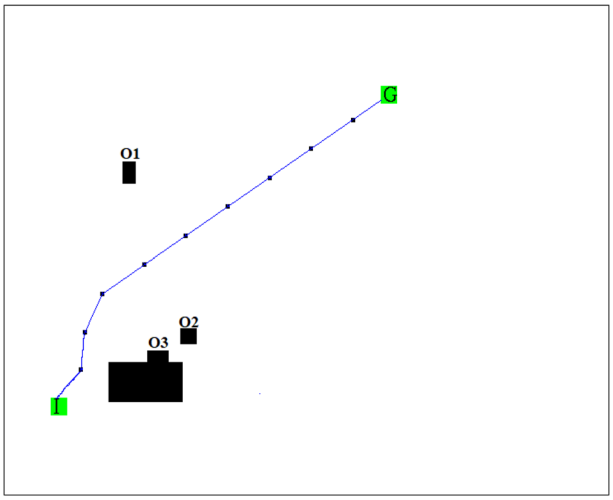

Fig. 29 Robot reaching the goal position for potential field method in [38]

O1 covers a distance of 3 units in the negative x direction and 4 units in positive y direction in 1 s. O2 moves a distance of 3 units in positive x direction and 3 units in negative y direction in 1 s. O3 moves a distance of 4 unit in positive x direction in 1 s.

i. Initial position, I of the mobile robot and moving obstacles is shown in Fig. 20. The segments for the first motion are also shown in the figure.

ii. In Fig. 21, the robot chooses Segment3 for the **first motion**. O3 is the critical obstacle for Segment3, and it reaches the segment in 0.4 s for the first motion. There is no critical obstacle for other segments. Segment4 is blocked by static obstacle.
The input to the MLP network for this motion is

$$\{0, 0, 0.7, 0, 1.5\}$$

The output is

$$\{0.000000, 0.000061, 0.460933, 0.000000, 1.000000\}$$

The segments for second motion are also shown in the figure.

iii. In Fig. 22, the robot chooses Segment2 for the **second motion**. O3 is the critical obstacle for Segment2, and it reaches the segment in 0.4 s for the second motion. There is no critical obstacle for other segments. Segment1 is blocked by O3 just before the second motion, and Segment3, Segment4 and Segment5 are blocked by static obstacle. The input to the MLP network for this motion is

$$\{0, 1.2, 0, 0, 0\}$$

Table 3 Comparison of path length and computational time for the second environment

Methods	Start position	Goal position	Path length (pixels)	Computational time (s)
GA-Fuzzy in [32]	(− 90,− 70)	(30,70)	195.75	0.0640
Fuzzy-WDO in [28]	(− 90,− 70)	(30,70)	205.73	0.5048
Potential field method in [38]	(− 90,− 70)	(30,70)	197.63	0.2063
Proposed method	(− 90,− 70)	(30,70)	**190.23**	**0.0050**

Bold values indicate that in the proposed method the path length and computational time is less

The output is

$$\{0.000000, 0.999996, 0.000000, 0.000000, 0.000001\}$$

The segments for third motion are also shown in the figure.

iv. In Fig. 23, the robot chooses Segment4 for the **third motion**. There is no critical obstacle for all the segments. Segment2 and Segment3 are blocked by O2 just before the third motion. The input to the MLP network for this motion is

$$\{1.5, 0, 0, 1.5, 1.5\}$$

The output is

$$\{0.999972, 0.000002, 0.000000, 1.000000, 1.000000\}$$

The segments for fourth motion are also shown in the figure.

v. In Fig. 24, the robot chooses Segment3 for the **fourth motion**. O2 is the critical obstacle for Segment2, and it reaches the segment in 0.3 s for the fourth motion. Segment1 is blocked by O2 just before fourth motion. The input to the MLP network for this motion is

$$\{0, 0.9, 1.5, 1.5, 1.5\}$$

The output is

$$\{0.000000, 0.534637, 1.000000, 0.999985, 0.999983\}$$

The segments for fifth motion are also shown in the figure.

vi. In Fig. 25, the robot reaches the global destination.

6 Comparison

The diagrammatic path length of our method is shown in Fig. 26, and for other methods, namely GA-Fuzzy in [32], Fuzzy-WDO in [28] and potential field method in [38] are shown in Figs. 27, 28 and 29, respectively. Table 3 shows the data of comparison of path length and computational time of our proposed method with other two standard methods in the same second environment. It shows that the proposed method gives optimal path with less computational time as compared to GA-Fuzzy in [32], Fuzzy-WDO in [28] and potential field method in [38].

7 Conclusion

In this paper, we have described a new method for mobile robot navigation in dynamic environment containing static and moving obstacles using artificial neural network. Multilayer perceptron (MLP) neural network is used to choose a collision-free segment from a set of five segments, and the network also controls the speed of the robot for each motion of robot. For each motion, the MLP computes the time taken by the robot to reach its local destination for each collision-free segment using the time taken by the critical obstacles. Finally, the robot chooses a segment and path which is nearest to the global destination. Simulation results show that the neural network gives optimal path with less computational time as compared to other standard approaches. In the future work, we will consider for automated driving system studying its performance and development of autonomous systems like autonomous soldier robot, UAV (unmanned aerial vehicle) and hill-climbing robot.

References

1. Algabri M, Mathkour H, Ramdane H, Alsulaiman M (2015) Comparative study of soft computing techniques for mobile robot navigation in an unknown environment. Comput Hum Behav 50:42–56
2. An MZ, Zeng GH, Si YF, Min T (2006) Neural networks for mobile robot navigation: a survey. In: Proceedings of the third international conference on advances in neural networks 2:1218–1226
3. Andurkar AG, Rupali T (2017) Fuzzy logic based path navigation for robot using Matlab. Int Res J Eng Technol 04:3165–3170
4. Anmin Z, Zengguang H, Lejie Z, Min T (2005) A neural network-based camera calibration method for mobile robot localization problems. In: Proceedings of the second international conference on advances in neural networks 3:277–284

5. Asita KR, Dayal RP, Harish CD, Manoj KM, Priyadarshi BK (2018) Analysis and use of fuzzy intelligent technique for navigation of humanoid robot in obstacle prone zone. Def Technol 17:1–6
6. Azzeddine B, Abdelfetah H, Hakim B, Abderraouf M, Ouarda H, Brahim B (2017) Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. Robot Auton Syst 89:95–109
7. Capi G, Kaneko S, Hua B (2015) Neural network based guide robot navigation: an evolutionary approach. Proc Comput Sci 76:74–79
8. Danica J (2004) Neural networks in mobile robot motion. Int J Adv Robot Syst 1(1):15–22
9. David LC, Wen Y (2017) Path planning of multi-agent systems in unknown environment with neural kernel smoothing and reinforcement learning. Neurocomputing 233:34–42
10. Faisal M, Hedjar R, Al-Sulaiman M, Al-Mutib K (2013) Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment. Int J Adv Robot Syst 10(37):1–7
11. Farhad B, Sepideh NN, Morteza A (2018) Mobile robots path planning: electrostatic potential field approach. Expert Syst Appl 100:68–78
12. Guan CL, Wei WL (2008) An immunological approach to mobile robot reactive navigation. Appl Soft Comput 8:30–45
13. Hajer O, Mohamed S, Mohamed M (2016) Fuzzy logic based control for autonomous mobile robot navigation. Comput Intell Neurosci, Volume 2016, Article ID 9548482, 10 pages
14. Julien M, Pierre M, Stephane V, Francois A, Franck G (2017) Path planning with fractional potential fields for autonomous vehicles. IFAC Pap 50:11433–11438
15. Karray A, Njah M, Feki M, Jallouli M (2016) Intelligent mobile manipulator navigation using hybrid adaptive-fuzzy controller. Comput Electr Eng 56:773–783
16. Mahdi F, Amirreza K, Mohsen J (2016) Humanoid robot path planning with fuzzy Markov decision processes. J Appl Res Technol 14:300–310
17. Mariam Al S, Rached D (2016) Neural based autonomous navigation of wheeled mobile robots. J Autom Mobile Robot Intell Syst 2(2):64–72
18. Matoui F, Boussaid B, Metoui B, Frej GB, Abdelkrim MN (2017) Path planning of a group of robots with potential field approach: decentralized architecture. IFAC Pap 50:11473–11478
19. Mauricio B, Douglas WB, Nardnio AM (2017) A robust adaptive fuzzy variable structure tracking control for the wheeled mobile robot: simulation and experimental results. Control Eng Pract 64:27–43
20. Meyesa R, Tercana H, Roggendorf S, Thiele T, Buscher C, Obdenbusch M, Brecher C, Jeschke S, Meisen T (2017) Motion planning for industrial robots using reinforcement learning. Proc CIRP 63:107–112
21. Ming-Chih L, Chen-Chien H, Yuan-Jun C, Shih-An L (2012) Hybrid path planning incorporating global and local search for mobile robot. In: Advances in autonomous robotics, Lecture notes in computer science 7429:441–443
22. Mohamed SM, Najla K, Mohamed M, Nabil D (2016) Fuzzy logic controllers design for omnidirectional mobile robot navigation. Appl Soft Comput 49:901–919
23. Mohamed B, Abdelkrim B, Mohammed C (2017) Robust adaptive neural network-based trajectory tracking control approach for nonholonomic electrically driven mobile robots. Robot Auton Syst 92:30–40
24. Mohanty PK, Parhi DR (2014) A new intelligent motion planning for mobile robot navigation using multiple adaptive neuro-fuzzy inference system. Appl Math Inf Sci 8(5):2527–2535
25. Mukesh KS, Dayal RP (2011) Path optimisation of a mobile robot using an artificial neural network controller. Int J Syst Sci 42:107–120
26. Ngangbam HS, Khelchandra T (2018) Mobile robot navigation using fuzzy logic in static environments. Proc Comput Sci 125:11–17
27. Omid M, Danial N, Sai HT, Babak K, Weria K (2013) Automatic navigation of mobile robots in unknown environments. Neural Comput Appl 24:15691581
28. Pandey A, Parhi DR (2017) Optimum path planning of mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm. Def Technol 13:47–58
29. Pandey A, Saroj K, Pandey KK, Parhi DR (2016) Mobile robot navigation in unknown static environments using ANFIS controller. Perspect Sci 8:421–423
30. Patle BK, Parhi DR, Jagadeesh A, Sunil KK (2016) Probabilistic fuzzy controller based robotics path decision theory. World J Eng 13:181–192
31. Pothal J, Parhi DR (2015) Navigation of multiple mobile robots in a highly clutter terrains using adaptive neuro-fuzzy inference system. Robot Auton Syst 72:48–58
32. Pratihar DK, Kalyanmoy D, Amitabha G (1999) A genetic-fuzzy approach for mobile robot navigation among moving obstacles. Int J Approx Reason 20:145–172
33. Rahib HA, Irfan SG, Nurullah A, Ersin A, Ahmet C, Sanan A (2017) Fuzzy control of omnidirectional robot. Proc Comput Sci 120:608–616
34. Simon XY, Max M (2000) An efficient neural network approach to dynamic robot motion planning. Neural Netw 13:143–148
35. Thi TM, Cosmin C, Duc TT, Robin DK (2016) Heuristic approaches in robot path planning: a survey. Robot Auton Syst 86:13–28
36. Yaghmaie F Adib, Mobarhani A, Taghirad HD (2013) A new method for mobile robot navigation in dynamic environment: escaping algorithm. In: First RSI/ISM international conference on robotics and mechatronics (ICRoM), February 13–15, Tehran, Iran
37. Yan Z, Yuliang S, Wei W (2012) Mobile robot hybrid path planning in an obstacle-cluttered environment based on steering control and improved distance propagating. Int J Innov Comput Inf Control 8(6):4095–4109
38. Zhiyu Z, Junjie W, Zefei Z, Donghe Y, Jiang W (2018) Tangent navigated robot path planning strategy using particle swarm optimized artificial potential field. Optik 158:639–651