Available online at www.sciencedirect.com
SCIENCE DIRECT
Fuzzy Sets and Systems 157 (2006) 2171–2204
FUZZY sets and systems
www.elsevier.com/locate/fss

Time-optimal, collision-free navigation of a car-like mobile robot using neuro-fuzzy approaches

Nirmal Baran Hui, V. Mahendar, Dilip Kumar Pratihar*

Department of Mechanical Engineering, Indian Institute of Technology, Kharagpur, Kharagpur - 721302, India

Received 27 June 2004; received in revised form 4 April 2006; accepted 5 April 2006
Available online 11 May 2006

Abstract

Neuro-fuzzy approaches are developed, in the present work, to determine time-optimal, collision-free path of a car-like mobile robot navigating in a dynamic environment. A fuzzy logic controller (FLC) is used to control the robot and the performance of the FLC is improved by using three different neuro-fuzzy (NN-FLC) approaches. The performances of these neuro-fuzzy approaches are compared among themselves and with those of three other approaches, such as default behavior, manually-constructed FLC and potential field method, through computer simulations. The neuro-fuzzy approaches are found to perform better than the other approaches, in most of the test scenarios. Moreover, the performances of both the genetic algorithm (GA)-optimized NN-FLC (Mamdani Approach) as well as GA-optimized NN-FLC (Takagi and Sugeno Approach) are seen to be comparable. It is also interesting to note that the CPU times of all these approaches are found to be low. Thus, they might be suitable for on-line implementations.

Keywords: Car-like robot; Navigation; Collision-free; Time-optimal; Neuro-fuzzy; Potential field method; Genetic algorithm

1. Introduction

Current research in robotics aims to build an autonomous and intelligent robot, which can plan its motion in a dynamic environment. A successful use of an autonomous mobile robot depends on its controller. The control action of the manipulator robots (a priori known, carefully engineered and highly predictable workspace) is comparatively easy and stable too. But, controlling of a car-like robot is difficult, because of the fact that an exact analytical model of such kind of robots is highly complicated due to its complex geometry. Car-like mobile robots are subjected to non-holonomic (non-integrable) kinematic constraints involving the time derivatives of configuration variables [21,3,14,8,25] and dynamic constraints. The path of the robot is also constrained by the partially-unknown movement of the moving obstacles [5], known as uncluttered environment. Thus, to generate collision-free path of a car-like robot during its navigation among several moving obstacles, it should have proper motion planning as well as obstacle avoidance schemes [35].

* Corresponding author. Tel.: +91 3222 282992; fax: +91 3222 282278.
E-mail addresses: nimu_2005@yahoo.com (N.B. Hui), mahi_swiss@yahoo.co.in (V. Mahendar), dkpra@mech.iitkgp.ernet.in (Di.K. Pratihar).

0165-0114/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2006.04.004

Nomenclature

α'	deviation of the robot (motion planner's output)
β, γ	constants used to define the pattern of membership function distributions
ΔT	time step, seconds
η	learning rate
ξ_{att}	positive scaling factor for attractive potential
ξ_{rep}	positive scaling factor for repulsive potential
μ_f	coefficient of sliding friction
μ_{ij}	membership function distribution of the jth neuron lying in ith layer
$\dot{\varphi}$	steering angle during turning
ρ	instantaneous radius of curvature of the CG of the robot during turning
θ	angle between the x-axis and the main axis of the robot
a	tangential acceleration at the present step
a_1	tangential acceleration at the predicted step
a_n	normal component of acceleration
A_k	area of the kth fired rules
B_{ij}^{sc}	target output at a distance step s, training case c, of the jth neuron lying in the ith layer
C	total number of training cases
d_{goal}	distance between the robot and goal
$d_{\min}$	minimum distance required to reach the goal with zero velocity of the robot
$d_{\text{obs}}(X)$	distance function between the robot and obstacles
E_{ij}^{ss}	mean squared error between actual output and target output at a distance step s, of a training case c, of the jth neuron lying in the ith layer
f	total number of fired rules
F_{R}	predicted position of the robot
F_{O}	predicted position of the most critical obstacle
$F(X)$	potential force function
g	acceleration due to gravity
I_{ij}	input to the jth neuron lying in ith layer
I_{R}	present position of the robot
I_{O}	present position of the most critical obstacle
I	total number of distance steps
l, m, n, p, k	neurons lying at different layers
M	mass of the robot
M_k	center of area of the kth fired rules
N	rotational speed of the motor
O_{ij}	output of the jth neuron lying in ith layer'
P	power of the motor
r	radius of the wheels
R	resultant force acting on the robot
R_b	normal reaction force
R_n	normal centrifugal force
R_t	tangential force
T	traveling time
T_{rem}	time required of the robot to travel the last distance step
$U(X)$	artificial potential energy function
v	tangential velocity at each location of the CG
V	connecting weight between 1st and 2nd layers
W	connecting weight between 4th and 5th layers
${}^{W}R_{\text{B}}$	relation of body coordinate frame with respect to world coordinate frame

(x, y)	world coordinate frame of reference
$(\dot{x}, \dot{y})$	components of tangential velocity
(X, Y)	body coordinate frame of reference

Abbreviations

AH	ahead
AL	ahead left
ANFIS	adaptive network-based fuzzy inference systems
AR	ahead right
BPNN	back propagation neural network
CG	center of gravity
CPU	central processing unit
FLC	fuzzy logic controller
FR	far
GA	genetic algorithm
GR	gear ratio
H	high
KB	knowledge base
L	low
LT	left
NN	neural network
NR	near
RT	right
VF	very far
VH	very high
VL	very low
VN	very near

Both analytical like potential field method [5,6,20] as well as graph-based techniques [32] have been used to solve the navigation problems of robots involving static obstacles. But, all such methods may not be suitable for on-line implementations due to their inherent computational complexity and limitations. Recently, Pratihar [33] have made an extensive survey on the navigational schemes of mobile robots moving among static and/or moving obstacles. In a partially-unknown environment, motion planning depends on the sensory information of the environment, which might be associated with imprecision and uncertainty. Thus, to have a suitable motion planning scheme in an uncluttered environment, the controller of such kind of robots must have to be adaptive in nature.

Soft computing includes fuzzy logic, genetic algorithm, neural network and their different combinations [34,2] and it can solve such complex real-world problems within a reasonable accuracy. The computational complexity of such methods is also expected to be low, due to their heuristic nature. Quite a few researchers [10,7,1] have used fuzzy logic technique based on Zadeh's fuzzy set theory [44], to develop an alternative controller to the existing highly complex conventional mathematical controllers. It was found that the fuzzy logic controller (FLC) is robust in presence of perturbations, easy to design and implement. Moreover, it relaxes the need of an accurate mathematical model of the system by replacing the mathematical knowledge by human (expert) knowledge and intuitions. Although, human knowledge-based FLCs can adequately control a given process to a reasonable accuracy, it may not necessarily be the optimal one. It is also necessary to mention that the process of knowledge acquisition for an FLC is a challenging task. A systematic approach for determining the knowledge base (consisting of membership function distributions of the variables, i.e., data base and rule base) of an FLC is yet to be developed.

Several methods had been suggested by various investigators for fuzzy rule generation. In this connection, work of Takagi and Sugeno [39], Wang and Mendel [42] are worth mentioning. Moreover, Nomura et al. [31] used a gradient descent method for fuzzy rule generation. More recently, Fukuda et al. [9] used reinforcement-learning technique for determining a good rule base of an FLC. Moreover, several researchers had attempted the problem of fuzzy rule generation by using neural networks [30,38,15,13,24,17].

Since artificial neural networks (ANN) [12] have the ability to learn the situations, many investigators have successfully applied the feed-forward neural network [29,43,22] to develop the model related to the navigation problem of a car-like robot. Abdessemed et al. [1] used fuzzy-Kohonen clustering network (FKCN) in combination with heuristic rule base of the fuzzy logic controller, to build the desired mapping between the perception of the human knowledge and the exact motion of the robot. But, the main drawback of neural network lies in the fact that the operations cannot be stated explicitly. A considerable amount of research is being carried out in Japan, Taiwan, China, Spain and France [28,23,36,26] on the possibilities of combining these two techniques. In some of the cases, an NN has been utilized to model the complex non-linear system and a fuzzy logic controller is used for controlling that process. Various kinds of NNs are tested to extract the rules of fuzzy logic controller. In such cases, two NNs are used [37,11], one for the model identification or the pattern recognition and the other to extract the rules for the fuzzy logic controller. Until or unless the knowledge base of the FLC is tuned, the adaptability of the system may not be good enough. Thus, there is a need for tuning the fuzzy variables, in order to train them to the desired situations.

Navigation problem of a car-like robot is tackled, in the present paper, by using three different neuro-fuzzy approaches. The performances of these approaches, to generate time-optimal, collision-free path of a robot, are compared among themselves and with those of other three approaches, namely default behavior, manually-constructed FLC and potential field method.

The rest of the paper is organized as follows: Section 2 explains the mathematical formulation of the problem and suggests a possible solution of the problem using an FLC. The developed approaches are described in Section 3. Simulation results of both the approaches are shown in Section 4. Some concluding remarks are made in Section 5 and the scope for future work is discussed in Section 6.

2. Mathematical formulation of the problem

A mobile car-like robot has to move from an initial position to a final position by avoiding collisions with a set of moving obstacles in minimum traveling time, after satisfying the kinematic and dynamic constraints. The robot's path is also constrained by the partially-unknown movement of moving obstacles [34]. To generate a collision-free path, the robot may have to move along a straight path or take a turn depending on the situations. The following assumptions are made to simplify the problem:

- All the moving obstacles are represented by their respective bounding circles,
- At a time, only one obstacle is considered to be critical and no two obstacles are allowed to overlap,
- The wheels of the robot are subjected to pure rolling action only,
- Non-holonomy and dynamic constraints of the vehicle are considered only,
- Coriolis component of the force is not taken into account.

Fig. 1 shows a typical problem scenario, in which a car-like robot is moving among five moving obstacles, in the same workspace. The robot has to find its time-optimal and collision-free path. S and G are the starting and goal positions of the robot, respectively. The relative velocity of the robot plays an important role for determining the most critical obstacle. The obstacle physically closest to the robot, may not be treated as the most critical one always. If any obstacle lies within an angle of 120° (within ±60° from the robot's main axis and inside the imaginary extended bounding circle of the robot) and is directed towards the robot, then it might be considered as the critical one. Among all such obstacles lying within the angle of search, the physically closest one is taken as the most critical obstacle. Thus, although the obstacle O_4 is the physically closest to the robot, it is not being treated as the most critical one. Rather the obstacle O_3 is considered to be the critical one, because it lies within the angle of search and is directed towards the robot also. The radius of the imaginary extended boundary circle of the robot is taken equal to the distance that the robot can travel in one distance step. The angle of search is decided based on the fundamentals of human vision. Moreover, there is a possibility that the obstacles may hit the boundary of the environment and in such incidences, they are supposed to be bouncing back from the boundary by following the laws of reflection. The aim of the present work is to develop a suitable adaptive and robust controller for the robot.

Two reference frames, namely world coordinate frame {W} and body coordinate frame {B} have been considered for the purpose of analysis. The origin of the body coordinate frame is fixed at the CG of the robot. These two coordinate frames are related to each other (refer to Fig. 2(a)) in the following manner.

$\{X'\} = \{x\} - {}^{\mathrm{W}}R_{\mathrm{B}}\{X\}$, where $\{X\}$ denotes the column vector of the coordinate matrix with respect to the body-coordinate frame, $\{x\}$ denotes column vector of the coordinate matrix with respect to the world-coordinate frame and

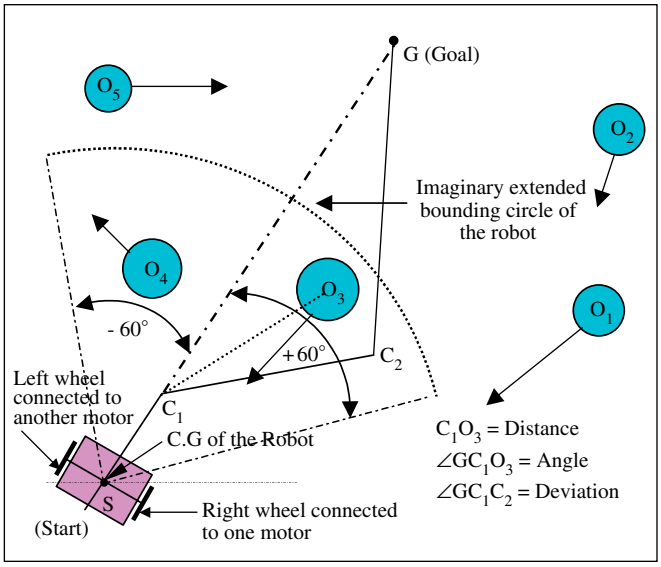

Fig. 1. Robot motion among moving obstacles.

${}^{W}R_{B}$ is the rotation matrix.

$$ {}^{W}R_{B} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \{X\} = \begin{Bmatrix} X \\ Y \\ \theta \end{Bmatrix}, \quad \{x\} = \begin{Bmatrix} x \\ y \\ \theta \end{Bmatrix}, $$

where $\{X'\}$ indicates the coordinates of the origin of body-coordinate frame $\{X\}$ with respect to the world-coordinate frame $\{x\}$.

The turning at a particular time step is obtained in ten steps as shown in Fig. 2(b), to get the accuracy in motion and at the same time, for easy steering of the robot.

In the next two sub-sections, the kinematic and dynamic constraints of the car-like robot have been explained, in detail.

2.1. Kinematics of the car-like robot

The car-like robot is modeled as a rigid body, which is moving on a plane surface and is supported by two wheels making point contact with the ground. A rigid body moving on a plane has only one center of rotation under the perfect rolling condition, and a wheel must move along the normal direction to its axle. Let us consider that in a particular small time duration ΔT, the CG of the robot moves from C to C_1 by following the curved path (refer to Fig. 3), with the tangential velocity v and O is the instantaneous center of rotation.

The configuration of the robot moving on a plane surface at every time instant is defined by a triple (x, y, θ). The following equations are to be satisfied by the car-like robot:

$$ -\dot{x}\sin\theta + \dot{y}\cos\theta = 0, \tag{1} $$

$$ (\dot{x})^2 + (\dot{y})^2 - (\rho_{\min}\dot{\varphi})^2 \geqslant 0, \tag{2} $$

where $\dot{x}$ and $\dot{y}$ are the component of tangential velocity along + ve x-axis and + ve y-axis, respectively,

θ: Angle between the x-axis and the main axis of the robot,

ρ: Instantaneous radius of curvature of the CG of the robot during turning,

φ: Steering angle during turning,

For any real value of the velocity, ρ must be real, to get a real curved path.

Thus, $\Delta\varphi$ has to lie within 0° and 90°, i.e., $0° < \Delta\varphi < 90°$.

Eqs. (1) and (2) are non-holonomic [21] and θ can be derived in a unique and straightforward manner from $(\dot{x}, \dot{y})$.

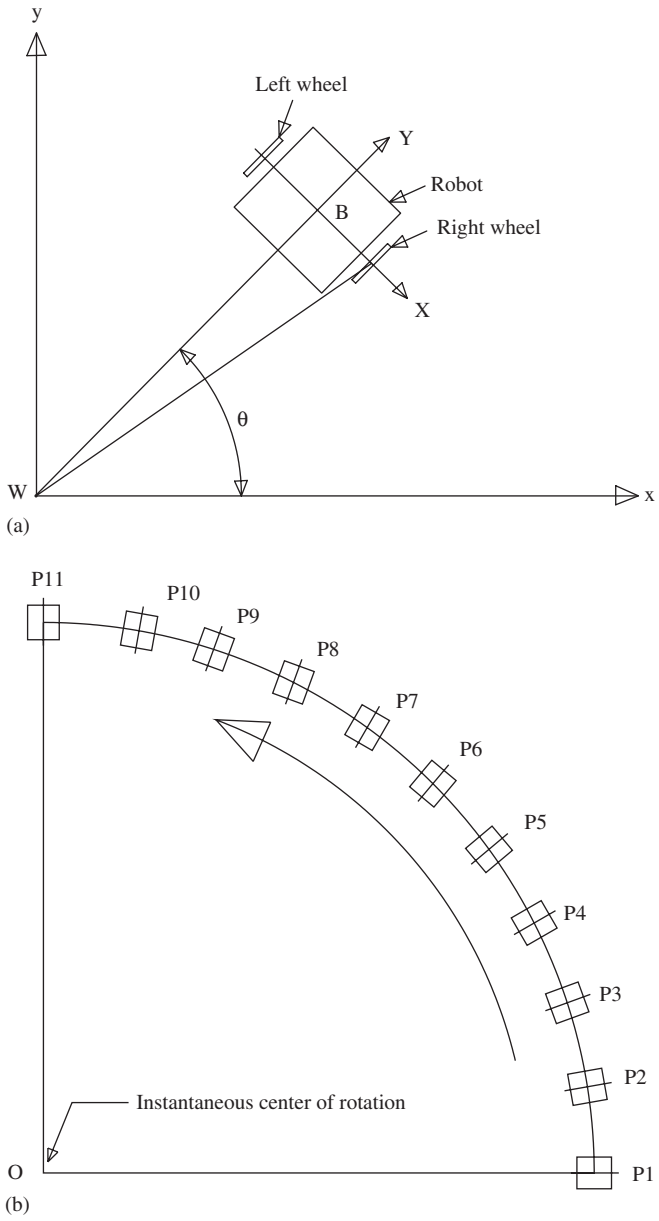

Fig. 2. (a) A schematic diagram to show the relation between two coordinate frames. (b) Rotation of the robot.

2.2. Dynamics of the car-like robot

The robot is subjected to various dynamic constraints [25,5,6] (i.e., sliding constraint, motor torque constraint, curvature constraint, etc.). Each constraint can be transformed into the constraints related to the velocity and acceleration of the robot as explained below:

2.2.1. Sliding constraints

The wheels of the robot should not have any sliding movement, i.e., they have rolling motions only. The forces acting on the center of gravity (CG) of the robot are as follows:

- Tangential force along the direction $\hat{t}$ to create tangential acceleration is given by $R_t = Ma_t = Ma$, where M is the mass of the robot, a is the tangential acceleration.

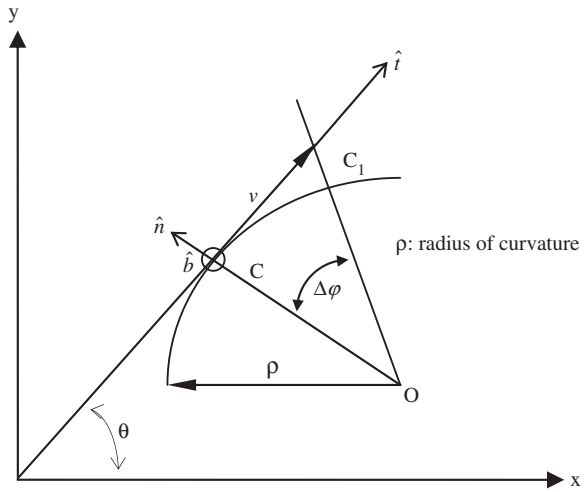

Fig. 3. Car-like robot taking a turn.

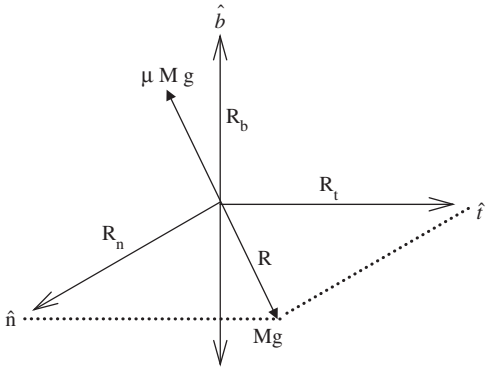

Fig. 4. Free body diagram of the robot in 3-D space.

- Normal centrifugal force along the normal direction ($\hat{n}$) to prevent the centripetal action is expressed by $R_n = Ma_n = M(v^2/\rho)$, where v is the tangential velocity of the robot
- Normal reaction force along $\hat{b}$ to prevent the gravitational action is given by $R_b = Mg$, where g is the acceleration due to gravity.

Resultant force in the plane $(\hat{t}, \hat{n})$ (refer to Fig. 4) will be

$$R = \sqrt{R_t^2 + R_n^2} = \sqrt{(M \times a)^2 + \left(\frac{Mv^2}{\rho}\right)^2}.$$

Now, to prevent the slippage during turning, sliding frictional force should be greater than the resultant force of the tangential force and normal centrifugal force acting in the plane $(\hat{t}, \hat{n})$, i.e., $\sqrt{(M \times a)^2 + (\frac{Mv^2}{\rho})^2} \leqslant \mu_f Mg$, where μ_f is the coefficient of sliding friction.

The expression for acceleration a can be written as

$$-\sqrt{(\mu_f \mathrm{g})^2 + \left(\frac{v^2}{\rho}\right)^2} \leqslant a \leqslant \sqrt{(\mu_f \mathrm{g})^2 - \left(\frac{v^2}{\rho}\right)^2}. \tag{3}$$

Substituting $\rho = \frac{v}{\dot{\varphi}}$, in Eq. (3), we get

$$-\sqrt{(\mu_f g)^2 - (v\dot{\varphi})^2} \leqslant a \leqslant \sqrt{(\mu_f g)^2 - (v\dot{\varphi})^2}. \tag{4}$$

Now, a will be real only when discriminate of the equation (4) is positive,

$$\text{i.e., } -\frac{\mu_f g}{\dot{\varphi}} \leqslant v \leqslant \frac{\mu_f g}{\dot{\varphi}}. \tag{5}$$

Thus, a feasible range of velocity v and acceleration a can be determined to prevent sliding movement.

2.2.2. Motor torque constraint

Each wheel of the robot is driven by an individual motor connected through one gearbox and it rotates along the normal direction to its axle. DC motors are used to control the wheels. Thus, tangential velocity of the car body is restricted, due to the motor constraint condition related to angular speed. Let us assume that P is the power of the motor, $N_{\max}$ is the maximum rotational speed of the motor in r.p.m, r is the radius of the wheels and GR indicates the gear ratio. Thus, tangential acceleration may follow the following approximate relation:

$$a \geqslant \frac{60P}{2\pi r \times \mathrm{GR} \times M \times N_{\max}}. \tag{6}$$

2.2.3. Curvature constraint

The turning movement of the robot is planned along a circular path. The radius of the curvature is lower bounded by the robot's geometry. Thus, the tangential velocity of the robot must maintain the equation given below.

$$v \geqslant \rho_{\min}\dot{\varphi}, \tag{7}$$

where $\rho_{\min}$ is the minimum radius of curvature.

2.3. A possible solution of the problem using FLC

The aim of this research is to develop a suitable adaptive controller, which can be implemented on-line. In actual navigation, information of the input variables collected by using the camera or sensor might be imprecise in nature. Thus, fuzzy logic controller could be a potential candidate for solving this problem. Two condition variables—*distance* and *angle*s are fed as inputs to the controller. *Distance* (C_1O_3, refer to Fig. 1) is the Euclidian distance between the robot and the most critical-obstacle forward to it. *Angle* ($\angle GC_1O_3$ refer to Fig. 1) is the relative angle between the path joining the robot and the goal and the path to the nearest obstacle forward. The relative velocity between the robot and obstacle is not explicitly considered as fuzzy variable. Instead, an incremental approach is adopted to eliminate its explicit consideration. There are two outputs of the FLC, i.e., action variables, such as *deviation* and *acceleration*. *Deviation* is the angle through which the robot has to move to avoid the collision and is measured with respect to the line joining the robot and its goal point. *Acceleration* is the acceleration with which the robot will move along a distance step. Two major approaches of developing FLC, namely Mamdani Approach [27] and Takagi and Sugeno Approach [40], have been tried in the present work.

In Mamdani Approach [27], the condition and action variables of the FLC are expressed in terms of membership function distributions. Fig. 5 shows the author-defined membership function distributions of both the input as well as output variables. For simplicity, the shape of the membership function is assumed to be triangular in nature. Four grades of *distance* are considered: very near (VN), near (NR), far (FR), very far (VF). The membership function distributions of both *angle* and *deviation* are assumed to be similar and their total range is divided into five terms: left (LT), ahead left (AL), ahead (AH), ahead right (AR) and right (RT). The range of *acceleration* is divided into four linguistic terms, namely very low (VL), low (L), high (H) and very high (VH).

The rule base is set manually based on intuition. With four choices for *distance* and five choices for *angle*, there could be 4×5 or 20 possible combinations of two different condition variables. For each of these 20 combinations, there could be two values of the action variables, one for *deviation* and another for *acceleration*. Thus, there is a maximum of 20 rules present in the rule base. All 20 rules that are used in this study, are shown in Table 1.

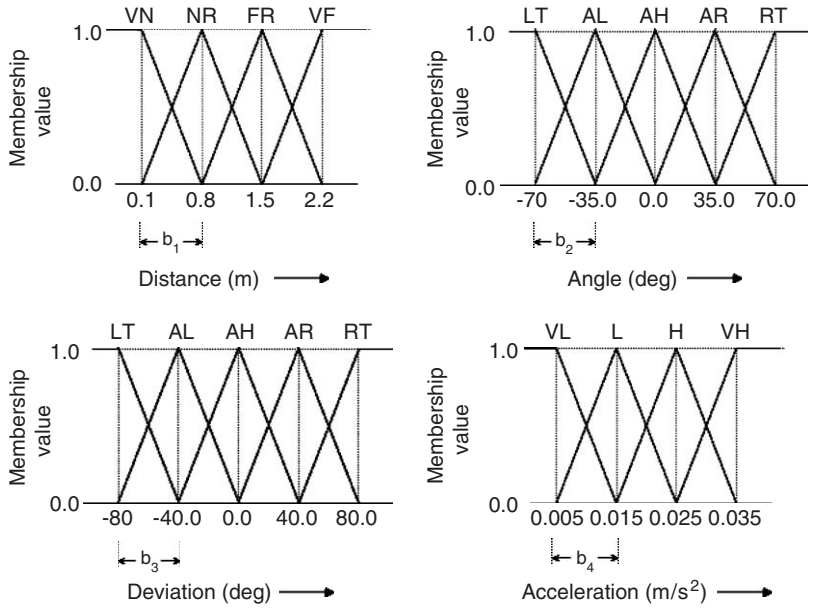

Fig. 5. Membership function distributions for input and output variables of the FLC.

Table 1
Rule base of the FLC for determining *deviation* and *acceleration*

Deviation

Distance \ Angle	LT	AL	AH	AR	RT
VN	AH	AR	LT	AL	AH
NR	AH	AR	RT	AL	AH
FR	AH	AH	AR	AH	AH
VF	AH	AH	AL	AH	AH

Acceleration

Distance \ Angle	LT	AL	AH	AR	RT
VN	H	H	VH	H	H
NR	L	H	VH	H	L
FR	VL	L	H	L	VL
VF	VH	VH	VH	VH	VH

A robot uses all these rules during its navigation. The fuzzy rules for determining the acceleration of the robot have been designed based on the principle that when it finds the most critical obstacle at a distance very far (VF) irrespective of the angular position of the obstacle, it will try to move faster to cover maximum distance, i.e., with maximum acceleration and minimum deviation. Moreover when the most critical obstacles is seen to move towards the robot from a distance very near (VN), the robot will again try to move faster but with greater deviation to avoid collision. Thus, a typical rule of the FLC (refer to 1st rule of Table 1) will look as follows:

IF distance is VN AND angle is LT, *THEN deviation is AH*, *acceleration is H*.

In Takagi and Sugeno Approach [40], the membership function distributions of the input variables have been assumed to be the same as shown in Fig. 5, whereas the outputs (i.e., consequents) are determined by using the first-order model (i.e., each output is expressed as a linear function of the input variables). Thus, a particular rule may be expressed like the following.

$$\textit{IF distance } (d) \textit{ is VN AND angle } (\theta) \textit{ is LT, THEN deviation } = a_1 d + b_1 \theta + c_1, \textit{acceleration } = a_2 d + b_2 \theta + c_2,$$

where a_1, b_1, a_2, b_2 are the coefficients of the input variables and c_1, c_2 are the constants.

The performance of an FLC is influenced by its knowledge base (KB). Thus, it is essential to tune the KB of the fuzzy logic controller to get a better performance. Since the tuning can be viewed as an optimization process, either a neural network (NN) or a genetic algorithm (GA) offers a possibility to solve this problem.

To control the robot, an incremental approach is adopted, where the robot's movement is a collection of small steps combining both straight as well as curved paths. The axis of the robot, at the starting position, may not be aligned towards the direction of the goal. Thus, to solve this problem, the wheels are rotated to align the robot's main axis towards the goal direction in $\Delta T/4$ seconds by maintaining all the constraints as discussed in Section 2 and making the assumption that the CG of the robot does not move forward or backward. The curved path is considered to be a circular path and its instantaneous radius of curvature is determined using the above-mentioned both kinematics as well as dynamic constraints. The velocity of the CG of the robot is kept constant during this turning movement. Each small distance step is traveled during a fixed time interval ΔT. Fig. 6 shows the velocity and acceleration distributions of the CG and wheels of the robot. For the first quarter of time cycle (A–B), when the robot starts from the predefined starting position, it travels with an acceleration (a) and then maintains a constant linear velocity $a\Delta T/4$ for half of the time cycle (B–C). At the point (C), the robot determines its path for the next time step, i.e., the robot will find *deviation* and *acceleration* (a_1) for the next cycle. If the robot has to change its path in the next distance step, then it travels with a deceleration (a) for the next 1/8th of cycle time (C–D) and then takes the turn with a constant velocity for the 1/4th of cycle time (D–F). Otherwise, the robot does not decelerate and continues in the same direction with the same velocity a $\Delta T/4$ from point C. During straight movement, the angular speeds of both the wheels are the same but during the turning movement of the robot, they alter according to the necessary conditions. It is possible to maintain two different speeds at the two wheels, if and only if individual motor separately controls each wheel. If the robot has to take a left turn, then the speed of the right wheel must be higher compared to that of the left wheel, and vice versa. It is important to mention that tangential speed of the CG of the robot is kept fixed, during the turning movement. The robot is assumed to have acceleration or deceleration lying within a fixed range. The range of the acceleration is set after a careful study considering the physical dimensions of the robot and its constraints.

2.3.1. Determination of the traveling time

The aim of this work is to generate a time-optimal/near-optimal collision-free path of a robot navigating among various moving obstacles present in the workspace. At the end of a constant velocity straight travel in a distance step, as discussed in the velocity diagram (refer to point C of Fig. 6), the robot senses the position of all obstacles and determine the most critical obstacle based on the distance and relative angle. *Acceleration* (a_1) and *deviation* of the robot for the next time step are determined using the motion planner. If the robot needs to deviate from its current direction of movement in the next step, then it starts decelerating for $\Delta T/8$ seconds to achieve the turning velocity of the CG of the robot. Otherwise, it will continue moving in the same direction (following a straight path) with the velocity $(a + a_1)\Delta T/8$ (the maximum velocity of the robot), where a symbolizes the acceleration of the present step and a_1 indicates the acceleration of the next distance step. It is important to mention that when the robot does not find any change in the direction of movement in two successive distance steps, there is a saving of time by $(7a - a_1)\Delta T/(32a)$ seconds (obtained by a trivial calculation) and the present step acceleration (a_1) is assumed to be equal to the previous step acceleration (a). Continuing in this fashion, when the robot comes closer to its destination and does not find any critical obstacles ahead of its movement, it starts decelerating from a distance ($d_{\min}$) of $(a + a_1)^2\Delta T^2/(128a_1)$, so as to reach the goal with zero velocity. It is important to mention that the last time step (T_{rem}) may not be a complete one; it depends on the distance left uncovered (d_{goal}) by the robot. If distance between the robot and goal (d_{goal}) comes to be less than the predefined minimum distance ($d_{\min}$), then it starts decelerating and stops at the goal. Moreover, at the start of navigation, the robot's main axis may not be directed towards the goal, thus, the robot requires an additional $\Delta T/4$ seconds time to align it's main axis towards the goal. Total traveling time (T) is then calculated by summing up the time steps needed to reach its predefined destination, starting from a fixed initial position as

$$T = \sum \Delta T + \frac{\Delta T}{4} - \sum \frac{7a - a_1}{32a}\,\Delta T + T_{\text{rem}}.$$

2.3.2. Collision avoidance

The robot's path is planned based on the predicted positions of the obstacles in the environment. Thus, the predicted position of the robot at the end of a time step may be absolutely collision-free but it may not be so, during its movement

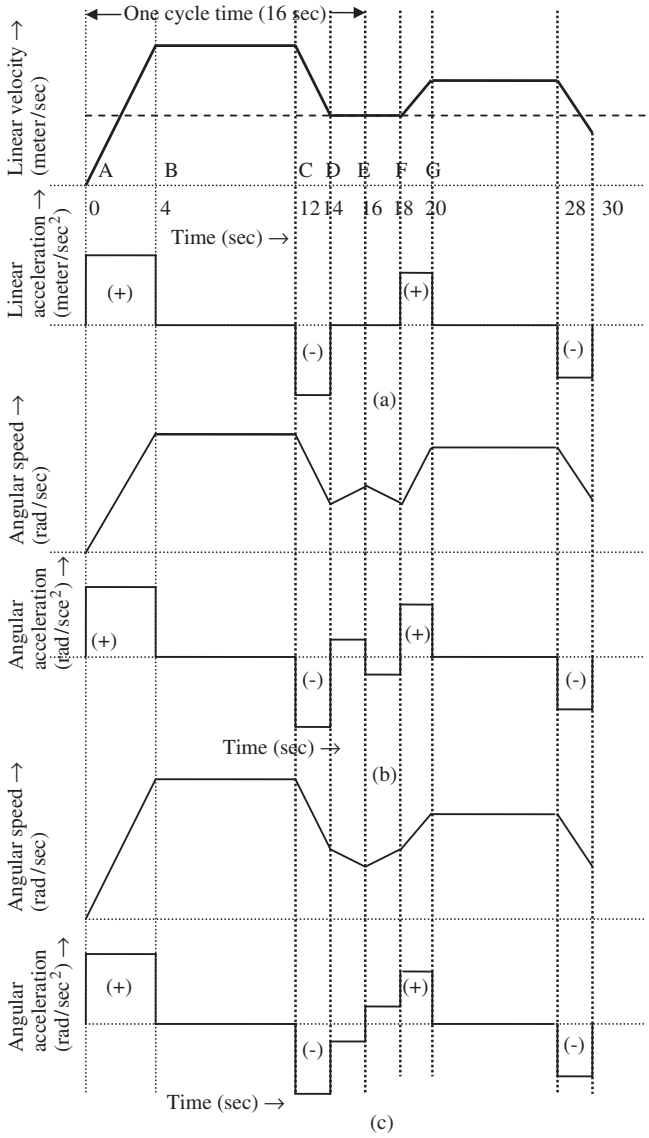

Fig. 6. Velocity and acceleration distributions (for taking the left turn) of: (a) CG of the robot; (b) right wheel of the robot; (c) left wheel of the robot.

from the current position to the predicted position. It depends on the duration of a time step. If the duration is more, the probability of collision between the planning robot and the moving obstacles inside the distance step is more, and vice-versa. It is to be noted that the back hitting of the robot by the obstacles is neglected, in the present work. This critical problem is solved with the help of relative velocity principle. Let us assume that the present and predicted positions of the robot are I_R and F_R (refer to Fig. 7) and those for the obstacle are I_O and F_O, respectively. During their navigation, it may happen that both the robot and the obstacle have reached the same point (C) at the same instant of time. Then, there will be a collision between the robot and the obstacle. In that case, one correction is to be made to the robot's future direction of movement to avoid the collision, otherwise the robot continues to move with the same angle as predicted by the motion planner. In Fig. 7, α is the angle made by the direction of movement of the robot with respect to the x-axis, in the previous step, whereas the angles α' and α'' indicate the directions of movement of the robot with respect to the x-axis, as planned by the motion planner and as corrected by the obstacle avoidance scheme, respectively. Thus, it can be treated as a traveling time minimization problem, subjected to several kinematic and dynamic constraints and unknown, uncertain and a little predictable environment.

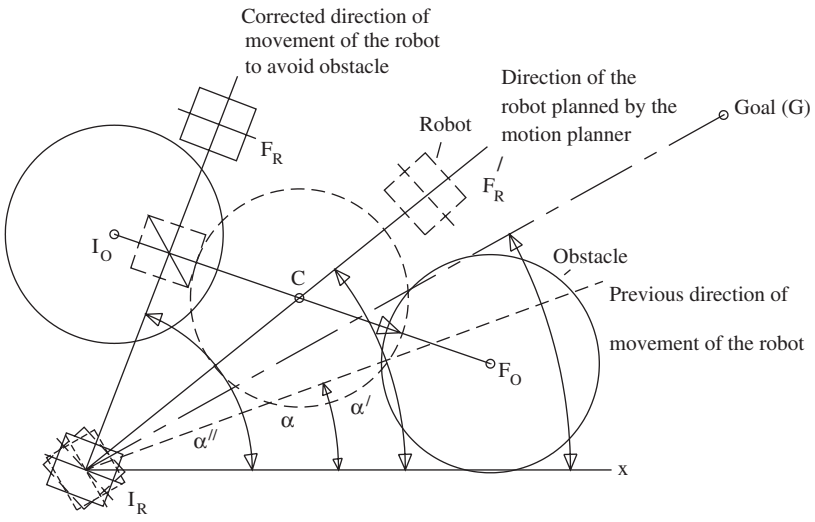

Fig. 7. Collision avoidance of the robot.

The objective function can be expressed as follows:

Minimize,

$$\text{Total traveling time, } T = \sum \Delta T + \frac{\Delta T}{4} - \sum \frac{7a - a_1}{32a}\,\Delta T + T_{\text{rem}}, \tag{8}$$

Subjected to:

- path is collision-free;
- kinematic and dynamic constraints are not violated.

The developed motion planning scheme of the robot is explained with the help of Figs. 8 and 9 as follows. At the beginning of navigation, the axis correction module is to be activated, if the robot's main axis does not coincide with the goal direction. After that, the robot finds whether there exists any critical obstacle ahead of it, in the predicted distance step. If it finds any critical obstacle, the motion planner is to be activated. Otherwise, the robot moves with the maximum possible acceleration and zero deviation in the next step. The outputs of the motion planner are nothing but the acceleration of the robot and the deviation necessary to avoid collision with the most critical obstacle. Moreover, if required, the motion planner's deviation output is to be corrected by using the collision avoidance scheme. If the robot's future direction of movement differs from the present one, then all the constraints are to be satisfied. It is to be noted that if there is no change of direction in two consecutive time steps, there will be a saving in traveling time, as discussed earlier. This process will continue, until the robot reaches its destination. The total traveling time is then calculated by summing up all the small time steps.

3. Developed algorithms

Navigation problem of the mobile robot has been solved by using six different approaches. Initially, it has been tackled by utilizing an approach, which works based on default behavior. A fuzzy logic controller has been designed in Approach 2, to solve the said problem. Moreover, three neuro-fuzzy approaches (i.e., Approaches 3, 4, 5) and potential field method (i.e., Approach 6) have been implemented to solve the above problem. All the six approaches are discussed below, in detail.

3.1. Approach 1: based on default behavior

In this approach, no motion planner is used for the robot. The robot maintains a default rule (i.e., move with the maximum acceleration and zero deviation) at each time step and the collision between the robot and the most critical obstacle is avoided with the help of a collision avoidance scheme explained in Section 2.3.2.

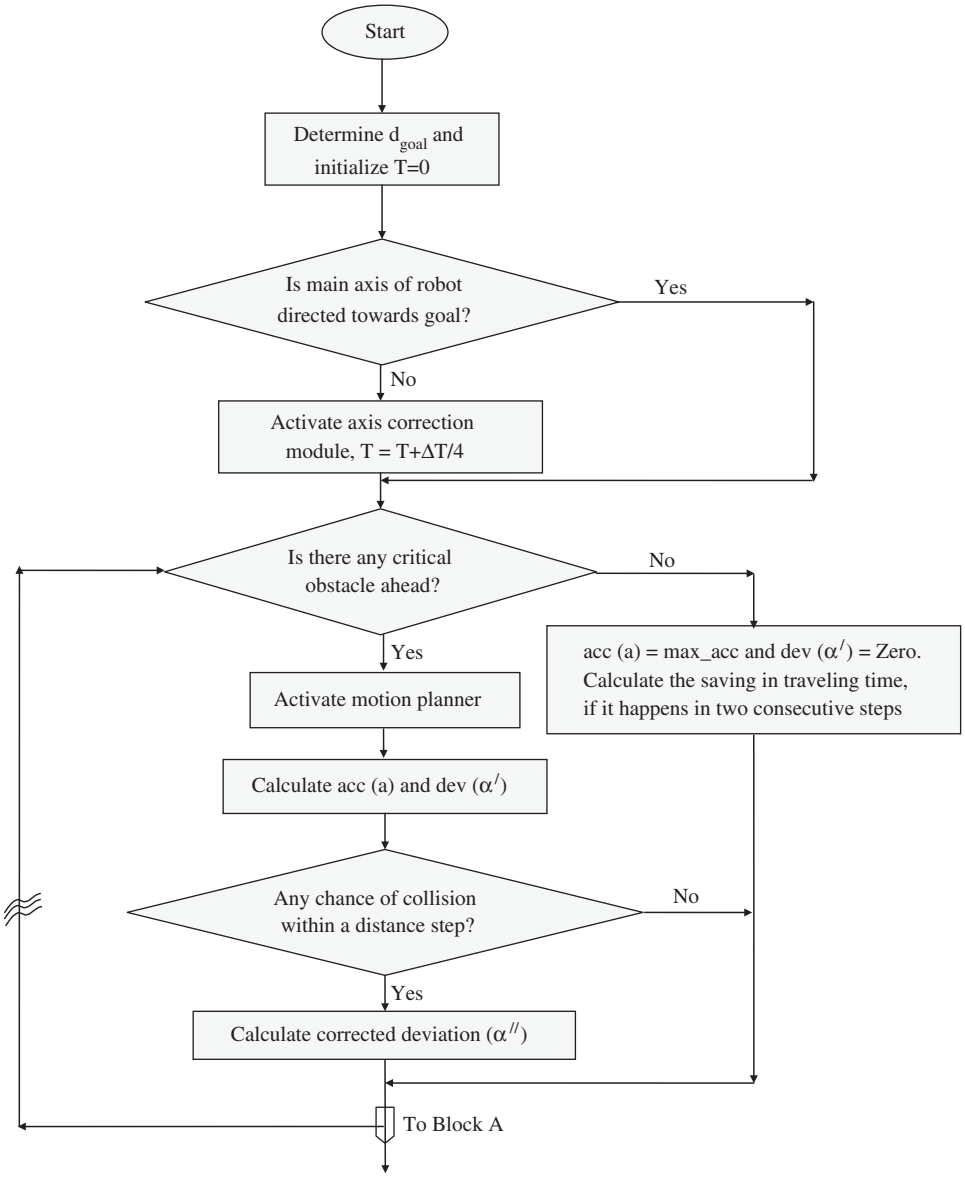

Fig. 8. A schematic diagram showing flowchart of the motion planning scheme.

3.2. Approach 2: manually-constructed fuzzy logic controller (FLC)

A feed-forward NN consisting of five layers is developed, in this approach, to design a fuzzy logic controller. The schematic diagram of this approach is shown in Fig. 10. The first layer transmits input values to the next layer using linear transfer function. The next layer is the fuzzification layer, in which the membership function values of the input variables are determined corresponding to input conditions. The third layer indicates all the possible rules and performs the logical 'AND' operation and the next layer identify all the fired rules. The inputs of the neurons lying on the fifth layer are calculated by using logical 'OR' operation. The fifth or last layer is the de-fuzzification layer, which converts the fuzzified output to its corresponding crisp value. There are two neurons in the first layer corresponding to two different input variables of the FLC (*distance, angle*). In the second layer, corresponding to the distance input, there exist four different neurons, which represent four different grades of the distance, namely very near (VN), near (NR),

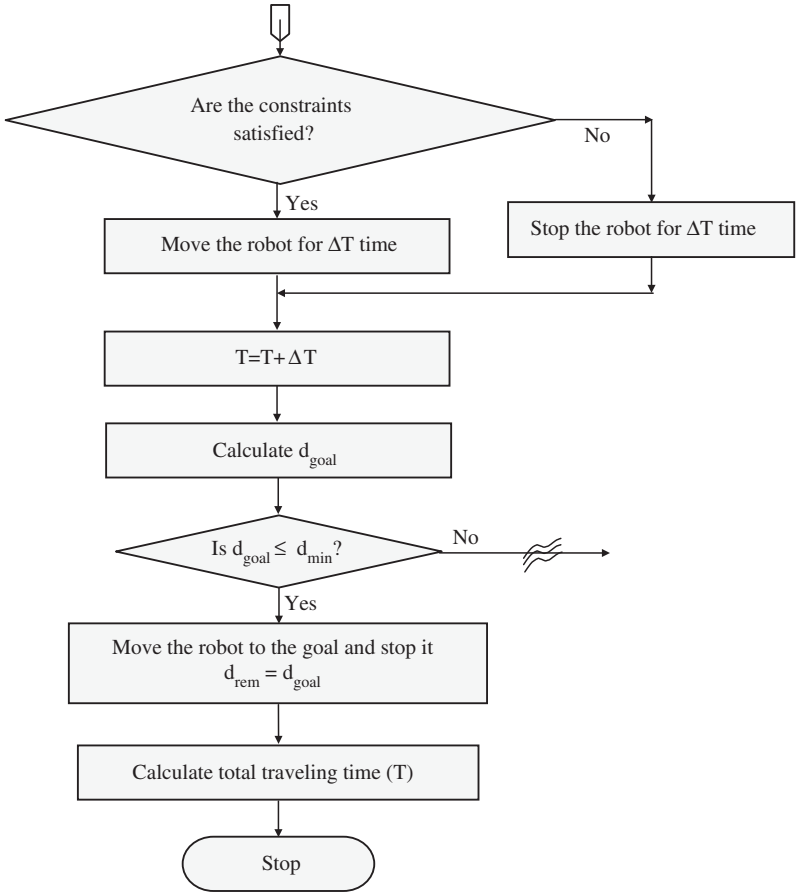

Fig. 9. A schematic diagram showing Block-A of Fig. 8.

far (FR) and very far (VF). Similarly, corresponding to the angle input, five neurons are considered in the second layer, which represent the five different grades of the angle input of the FLC. It is important to mention that in both these two cases, the weights $[V]$ of the second layer are taken equal to the half base-width of the fuzzy membership function distributions. Since there are twenty possible rules for these two input variables, twenty different neurons are considered in the third layer. Every neuron, in this layer, represents a corresponding fuzzy rule. In the fourth layer, nine neurons are considered, out of which five are used for representing the deviation output and the rest four neurons are utilized for indicating four different grades of the acceleration output. The connectivity between a neuron lying in the third layer and a neuron lying in the fourth layer represents the output of a particular rule. In the last layer, there exist only two neurons representing the two different outputs. The weights between the fourth and fifth layers, i.e., $[W]$ are also taken to be equal to the half base-width of the membership function distributions of the output.

The formulation of the developed neuro-fuzzy system is explained layer-by-layer as shown in Fig. 10. The following notations are used in this system: I_{ij} and O_{ij} represent the input and output, respectively of the jth neuron lying in ith layer, μ_{ij} indicates membership function value of the jth neuron lying in the ith layer. Let us assume that lth, mth, nth, pth and kth neurons are lying in the first, second, third, fourth and fifth layers, respectively, of the network.

Layer 1. Two variables, namely distance and angle are fed as inputs to the network. The output will be the same as the input, as a linear transfer function has been considered in this layer.

Layer 2. The inputs of this layer are taken to be equal to the outputs of the first layer. Thereafter, these crisp values of the inputs are converted into the fuzzy membership function values, with the help of membership function distributions. For both the inputs, the membership function distributions are taken to be triangular, for simplicity. Fig. 11 shows the triangular membership function distributions for different grades of input. I_{2m} and O_{2m} denote the input and output,

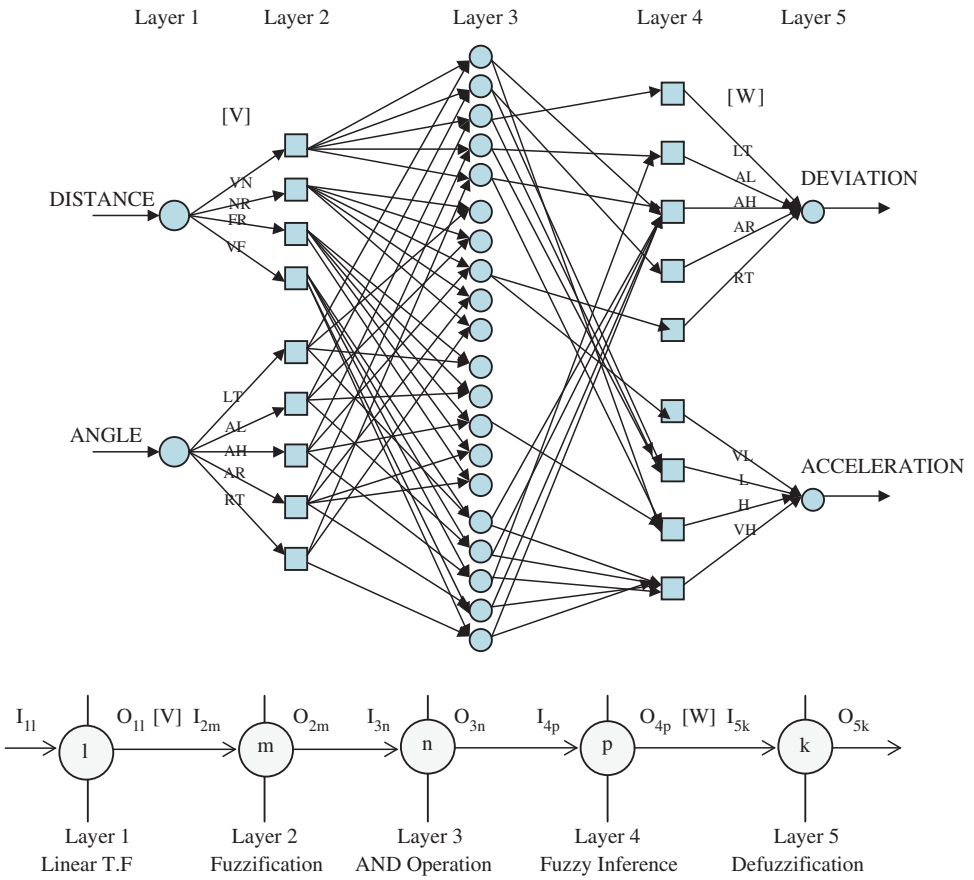

Fig. 10. A schematic diagram of the neuro-fuzzy approach.

respectively, of the mth neuron lying in the 2nd layer. S_{2m} indicates the starting point of the crisp value corresponding to a membership function distribution of mth neuron lying in the 2nd layer, V_{lm} represents the link weight between lth neuron of 1st layer and mth neuron of the 2nd layer. Three different possibilities may occur, for which the input–output relationships can be expressed as follows.

First possibility: the membership function distribution is a right-angled triangle (representing the left-most triangle for each variable of Fig. 5), as shown in Fig. 11(a).

$$\begin{aligned} O_{2m} &= \mu_{2m}(I_{2m}, S_{2m}, V_{lm}) \\ &= 1.0, \quad \text{if } I_{2m} \leqslant S_{2m}; \\ &= \frac{(S_{2m} - I_{2m})}{V_{lm}} + 1, \quad \text{if } 2m \leqslant I_{2m} \leqslant (S_{2m} + V_{lm}); \\ &= 0.0, \quad \text{if } I_{2m} \geqslant (S_{2m} + V_{lm}). \end{aligned} \tag{9}$$

Second possibility: the membership function distribution is a right-angled triangle (representing the right-most triangle for each variable of Fig. 5) as shown in Fig. 11(b).

$$\begin{aligned} O_{2m} &= \mu_{2m}(I_{2m}, S_{2m}, V_{lm}) \\ &= 0.0, \quad \text{if } I_{2m} \leqslant (S_{2m} + \gamma V_{lm}); \\ &= \frac{(I_{2m} - S_{2m})}{V_{lm}} - \gamma, \quad \text{if } (S_{2m} + \gamma V_{lm}) \leqslant I_{2m} \leqslant (S_{2m} + (\gamma + 1) V_{lm}); \\ &= 1.0, \quad \text{if } I_{2m} \geqslant (S_{2m} + (\gamma + 1) V_{lm}), \end{aligned} \tag{10}$$

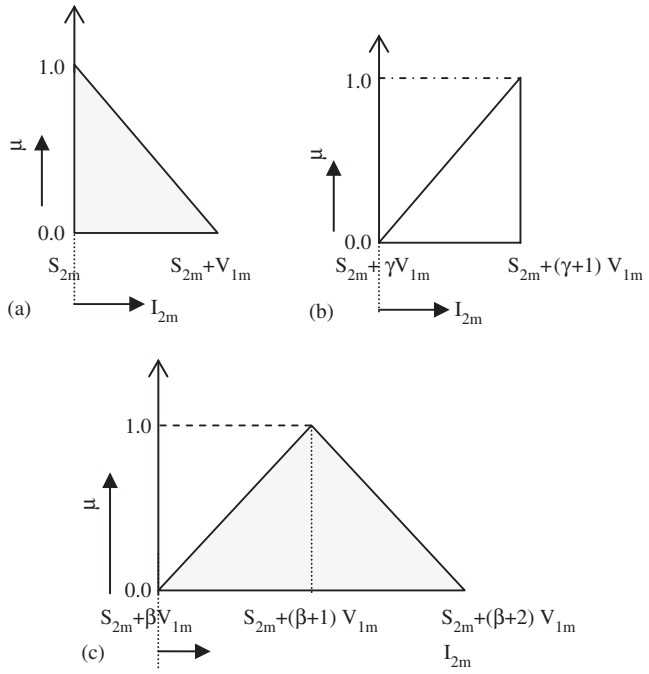

Fig. 11. Triangular membership function distributions for different grades of input: (a) first grade; (b) last grade; (c) intermediate grades.

Third possibility: the membership function distribution is a triangle as shown in Fig. 11(c).

$$\begin{aligned} O_{2m} &= \mu_{2m}(I_{2m}, S_{2m}, V_{lm}) \\ &= 0.0, \quad \text{if } I_{2m} \leqslant (S_{2m} + \beta V_{lm}) \text{ or } I_{2m} \geqslant (S_{2m} + (\beta+2)V_{lm}); \\ &= \frac{(I_{2m} - S_{2m})}{V_{lm}} - \beta, \quad \text{if } (S_{2m} + \beta V_{lm}) \leqslant I_{2m} \leqslant (S_{2m} + (\beta+1)V_{lm}); \\ &= \frac{(S_{2m} - I_{2m})}{V_{lm}} + (\beta+2), \quad \text{if } (S_{2m} + (\beta+1)V_{lm}) \leqslant I_{2m} \leqslant (S_{2m} + (\beta+2)V_{lm}), \end{aligned} \tag{11}$$

where β and γ are two constant quantities that define the pattern of fuzzy membership function distribution for the inputs. For the distance input, β takes the value of either 0 (i.e., near) or 1 (i.e., far) and γ is set equal to 2 (i.e., very far). Similarly, for the angle input, β varies from 0 to 2 (0 for ahead left, 1 for ahead, 2 for ahead right) and γ takes the value of 3 (i.e., right).

Let us take an example, in which one of the inputs, say *distance* has come out to be equal to 0.7m. It may be considered either VN or NR with different membership function values (refer to Fig. 5). Now, we will have to determine those membership function values. Using equation (9) (corresponding to VN *distance*), the output of 2nd layer, o_{2m} can be obtained as follows:

$$o_{2m} = \frac{0.1 - 0.7}{0.7} + 1.0 = \frac{1}{7}.$$

Similarly, corresponding to NR *distance*, o_{2m} can be calculated using Eq. (11) as follows:

$$o_{2m} = \frac{0.7 - 0.1}{0.7} - 0.0 = \frac{6}{7}.$$

Layer 3. This layer performs the task of logical AND operation. Each neuron lying in this layer is connected to two neurons of the previous layer, as shown in Fig. 10. Membership function values calculated in the previous layer are considered as the inputs of a particular neuron (say nth) lying in this layer. These two membership function values are compared and the minimum of these two values is taken as the output of that nth neuron [27].

Layer 4. This layer is the fuzzy inference layer, which identifies the fired rules for a set of inputs.

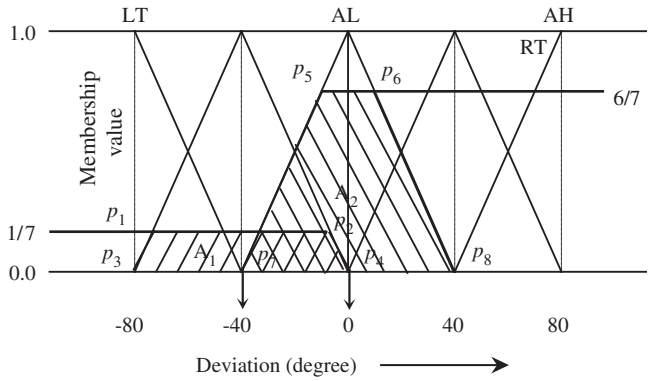

Fig. 12. Defuzzification using center of sums method.

Layer 5. The connecting weights between the 4th and 5th layers indicate the membership function distributions of the output variables. Once the membership function distributions are known, this layer calculates the output of all the fired rules (in terms of area of membership function distributions). After the outputs of all the fired rules have been determined, they are superimposed to get the fuzzified output, considering all the fired rules together. As the fuzzified output (nothing but an area) is not suitable for implementation as a control action, a crisp value corresponding to this fuzzified output is calculated. This process is known as de-fuzzification. A center of sums method is adopted for de-fuzzification. The final output O_{5k} of the kth neuron lying in this layer can be expressed as follows.

$$O_{5k} = \frac{\sum_{k=1}^{f} A_k M_k}{\sum_{k=1}^{f} A_k}, \tag{12}$$

where A_k and M_k are the area and center of area, respectively, for kth fired rule and f is the total number of fired rules. Thus, in the present work, an FLC has been developed based on Mamdani Approach [27].

Let us take an example, in which we will have to determine the outputs (such as deviation and acceleration) of the FLC, for a set of inputs—distance $= 0.05$ m and angle $= -40.0°$. Let us also assume that the above distance can be expressed by using only one linguistic term, say VN and the said angle may be called either LT or AL. Under these circumstances, only two rules will be getting fired as given below.

Rule 1: IF distance is *VN AND* angle is *LT*, *THEN* deviation is *AH*, acceleration is *H*,

Rule 2: IF distance is *VN AND* angle is *AL*, *THEN* deviation is *AR*, acceleration is *H*.

Let us try to calculate the crisp value of only one output, say deviation, corresponding to the above two fired rules. We assume that the membership value corresponding to VN distance is coming out to be equal to 1.0 and that corresponding to LT and AL angle is 1/7 and 6/7, respectively. Thus, the firing strength of Rules 1 and 2 will be equal to $\min(1.0, 1/7) = 1/7$ and $\min(1.0, 6/7) = 6/7$, respectively, as per Mamdani Approach [27]. Let us also assume that after considering the connecting weights between Layers 4 and 5, the membership function distributions of the output—deviation will take the form as shown in Fig. 12. Now, we calculate the area of the fuzzified output (say A_1) corresponding to Rule 1, like the following:

$$A_1 = \frac{1}{2}(p_1 p_2 + p_3 p_4) \times \frac{1}{7} = \frac{1}{2}\left(2 \times 40 \times \frac{6}{7} + 2 \times 40\right) \times \frac{1}{7} = \frac{520}{49}.$$

Similarly, the fuzzified output of Rule 2 (i.e., A_2) is found to be equal to $\frac{1920}{49}$. Now, we determine the centers (M_1 and M_2) of the areas—A_1 and A_2, respectively, which are coming out to be as follows: $M_1 = 0.0°$ and $M_2 = 40.0°$. Thus, the output—deviation can be obtained by using Eq. (12) like the following.

$$\text{Deviation} = \frac{\left(\frac{520}{49} \times 0.0\right) + \left(\frac{1920}{49} \times 40.0\right)}{\left(\frac{520}{49} + \frac{1920}{49}\right)} = 31.475°.$$

3.3. Approach 3: neuro-fuzzy approach

The performance of Approach 2 depends on both the rule base as well as database (i.e., $[V]$ and $[W]$ values). In the present approach, a back-propagation algorithm is adopted to find the optimal set of membership function distributions. During training, the weights between the 1st and 2nd layers, i.e., $[V]$ and those between the 4th and 5th layers, i.e., $[W]$ are to be updated to reduce the error in prediction. The batch mode of training is used to update the weights. In robot motion planning, a robot has to follow a few steps of movement consisting of both linear as well as curved paths, to reach the goal after starting from an initial position. Let us consider that there are C training scenarios and in a particular training scenario, there is a maximum of J distance steps. In a particular distance step, the mean squared error is given by $E_{5k}^{sc} = \frac{1}{2}(B_{5k}^{sc} - O_{5k}^{sc})^2$, where O_{5k}^{sc} indicates actual output from the controller at a distance step s of the cth training scenario and B_{5k}^{sc} indicates the target output at a distance step s. The target output has been set as follows. For acceleration output, the target is kept fixed to the maximum allowable acceleration and for deviation output, it is the zero deviation with respect to the new reference line joining the present position of the robot and its goal. Then for the cth training scenario, the average error can be expressed as follows:

$$\begin{aligned}\bar{E}_{5k}^{c} &= \frac{1}{2J}[(B_{5k}^{1c} - O_{5k}^{1c})^2 + \cdots + (B_{5k}^{sc} - O_{5k}^{sc})^2 + \cdots + (B_{5k}^{Jc} - O_{5k}^{Jc})^2] \\ &= \frac{1}{2J}\sum_{s=1}^{J}(B_{5k}^{sc} - O_{5k}^{sc})^2. \end{aligned} \tag{13}$$

Thus, average error during the training is calculated as follows:

$$\bar{E}_{5k} = \frac{1}{C}\sum_{c=1}^{C}\bar{E}_{5k}^{c} = \frac{1}{C}[\bar{E}_{5k}^{1} + \bar{E}_{5k}^{2} + \cdots + \bar{E}_{k}^{c} + \cdots + \bar{E}_{5k}^{C}]. \tag{14}$$

Therefore, change in $[W]$ weights is determined as follows.

$$\Delta W_{pk} = -\eta \times \frac{\partial \bar{E}_{5k}}{\partial W_{pk}}. \tag{15}$$

Since the inputs of the fifth layer is independent of the $[W]$ weights at that layer, $\partial \bar{E}_{5k}/\partial W_{pk}$ can be expressed as follows:

$$\frac{\partial \bar{E}_{5k}}{\partial W_{pk}} = \frac{\partial \bar{E}_{5k}}{\partial \bar{E}_{5k}^{c}} \times \frac{\partial \bar{E}_{5k}^{c}}{\partial O_{5k}^{sc}} \times \frac{\partial O_{5k}^{sc}}{\partial W_{pk}}. \tag{16}$$

The updated weight is then calculated by using the following expression:

$$W_{pk}(t+1) = W_{pk}(t) - \eta \times \frac{\partial \bar{E}_{5k}}{\partial W_{pk}}, \tag{17}$$

where η is the learning rate.

To calculate the change in $[V]$ weights, the contributions of both the outputs are taken into account, i.e., errors due to the individual outputs are combined and the average value is considered. Thus, it is essential to find out the effect of individual output on the change in error and to do so, four possible combinations are to be dealt with, as discussed below.

- Change in distance weight due to deviation output,
- Change in distance weight due to acceleration output,
- Change in angle weight due to deviation output,
- Change in angle weight due to acceleration output.

Then, the average error after the whole training can be expressed as follows:

$$\bar{\bar{E}} = \frac{1}{2}(\bar{E}_{51} + \bar{E}_{52}) = \frac{1}{2}\sum_{k=1}^{2}\bar{E}_{5k}. \tag{18}$$

The change in $[V]$ values can be determined as given below.

$$\Delta V_{lm} = -\eta \times \frac{\partial \bar{\bar{E}}}{\partial V_{lm}},$$

where

$$\frac{\partial \bar{\bar{E}}}{\partial V_{lm}} = \frac{1}{2}\left(\frac{\partial \bar{E}_{51}}{\partial V_{lm}} + \frac{\partial \bar{E}_{52}}{\partial V_{lm}}\right),$$

where

$$\frac{\partial \bar{E}_{5k}}{\partial V_{lm}} = \frac{\partial \bar{E}_{5k}}{\partial \bar{E}^{c}_{5k}} \times \frac{\partial \bar{E}^{c}_{5k}}{\partial \bar{E}^{sc}_{5k}} \times \frac{\partial \bar{E}^{sc}_{5k}}{\partial O^{sc}_{5k}} \times \frac{\partial O^{sc}_{5k}}{\partial O^{sc}_{4p}} \times \frac{\partial O^{sc}_{4p}}{\partial I^{sc}_{4p}} \times \frac{\partial I^{sc}_{4p}}{\partial O^{sc}_{3n}} \times \frac{\partial O^{sc}_{3n}}{\partial I^{sc}_{3n}} \times \frac{\partial I^{sc}_{3n}}{\partial O^{sc}_{2m}} \times \frac{\partial O^{sc}_{2m}}{\partial V_{lm}},$$

where $k = 1$ or 2.

Then the weights $[V]$ will be updated as per the following expression:

$$V_{lm}(t+1) = V_{lm}(t) - \eta \times \frac{\partial \bar{\bar{E}}}{\partial V_{lm}}, \tag{19}$$

where η is the learning rate.

3.4. Approach 4: genetic-neuro-fuzzy approach

In the previous approach (i.e., Approach 3), a back-propagation algorithm is used to optimize the membership function distributions (i.e., the data base) of the FLC. However, the solutions provided by the back-propagation algorithm may be trapped into the local minima. To overcome this difficulty, the performance of the NN-structured FLC is improved by using a GA, in the present approach. For this purpose, a binary-coded GA with 60-bits long string (the first 20 bits indicate the rule base—1 for presence and 0 for absence and 10 bits are utilized to represent the membership function distributions for each of the four variables) has been used to represent the knowledge base of the FLC. Thus, a particular GA-string will look as follows.

$$\underbrace{10\cdots01\cdots10}_{rule\ base}\ \underbrace{10\cdots01}_{V_1}\ \underbrace{01\cdots10}_{V_2}\ \underbrace{10\cdots10}_{W_1}\ \underbrace{11\cdots10}_{W_2}.$$

The GA begins its search by randomly creating a number of solutions (equals to the population size) represented in binary-coded strings. Each solution in the population is then evaluated to assign a fitness value. In this study, the fitness of a GA-string is considered to be the cumulative average of acceleration and deviation error considering all the training scenarios. Since the objective is to minimize the average error, the GA will find a string that corresponds to the minimum fitness value. It is important to mention that the error in deviation output may be either positive or negative. Thus, absolute value of the error has been considered in the fitness calculation. Moreover, if the output of the FLC in the predicted distance step is such, that the robot may collide with the most critical obstacle during its movement from the present position to predicted position, a fixed penalty equal to 200 is added to the fitness. Again, sometimes the motion of the robot as suggested by the FLC, is not possible to implement, due to its kinematic and/or dynamic constraints. In such a situation, the robot is stopped for that particular time step and a fixed penalty equal to 2000 is added to the fitness, to avoid such incidences. Therefore, the fitness of a GA-string is calculated as follows:

$$Fitness = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{1}{U}\sum_{u=1}^{U}\left\{\sum_{i=1}^{2}(T_{nui} - O_{nui})\right\}\right] + \text{Penalty}, \tag{20}$$

where U denotes the total number of completed time steps and N indicates the maximum number of training scenarios. T_{nui} and O_{nui} represent the target and calculated outputs of the FLC, respectively.

After each solution in the population is evaluated and fitness is assigned, the population is modified by using three operators-reproduction, single-point crossover and bit-wise mutation. The tournament selection compares two solutions

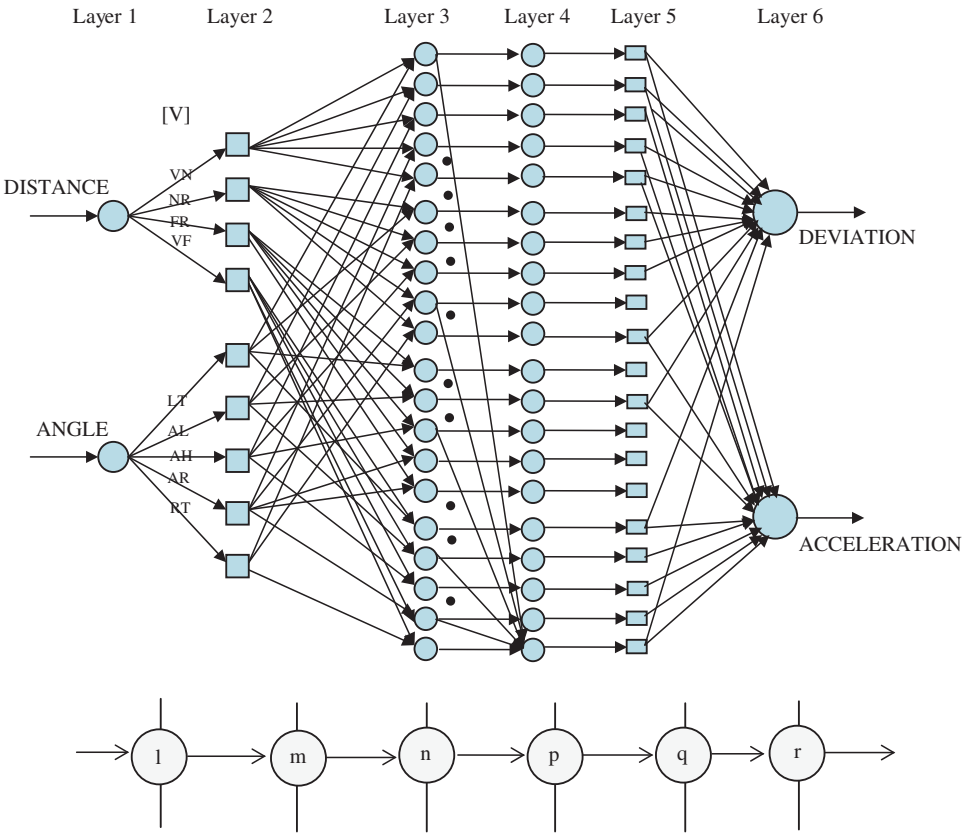

Fig. 13. A schematic diagram of the ANFIS structure.

at a time from the population and chooses the solution having the smaller fitness value. Crossover operator exchanges bits information between two such strings forming a mating pair and creates two new strings. The mutation operator alters a string locally to create a new string. After a new population of solutions is created, each of them is evaluated again to find its fitness value and all three operators are applied again. One iteration of these three operators followed by the evaluation procedure is called a generation. Generations proceed until a termination criterion is satisfied. In this study, the GA is allowed to run for a pre-specified number of generations. For a more detailed study of this approach, interested readers may refer to [16].

3.5. Approach 5: GA-tuned ANFIS

In this approach, a binary-coded GA is used to design an adaptive network-based fuzzy inference system (ANFIS) [18,19,41] for solving the navigation problems of a car-like robot among several moving obstacles. An ANFIS is a multilayer feed-forward network, in which each layer performs a particular function. The layers are characterized by the fuzzy operations they perform. Fig. 13 shows the schematic diagram of the ANFIS structure, which consists of six layers, such as Layer 1 (Input layer), Layer 2 (condition layer), Layer 3 (rule base layer), Layer 4 (normalization layer), Layer 5 (consequence layer) and Layer 6 (output layer). The first two layers perform the similar task as done by Layers 1 and 2 of Approach 2. The functions of rest of the layers are explained below.

Layer 3: Rule base layer.This layer defines the fuzzy rules. Each neuron in this layer represents a fuzzy rule and is termed as a rule node. The output of every neuron lying in this layer is the multiplication of their two incoming signals. For example, the output of a neuron, say n lying in this layer can be calculated as follows:

$$O_{3n} = O_{2i} \times O_{2j}, \tag{21}$$

where O_{2i} and O_{2j} are the outputs of the neurons i and j lying on Layer 2. It is to be noted that each node output represents the firing strength of a rule.

Layer 4: Normalization layer. This layer has the same number of nodes as in the previous layer. It calculates the normalized firing strength of each node, as shown below.

$$O_{4p} = \frac{O_{3n}}{\sum_{n=1}^{20} O_{3n}}, \tag{22}$$

where O_{4p} indicates the output of pth node lying on this layer.

Layer 5: Consequence layer. The output of a particular neuron (say, qth) lying on this layer is determined as given below.

$$O_{5q} = O_{4p}(a_q I_{11} + b_q I_{12} + c_q), \tag{23}$$

where I_{11} and I_{12} represent the two inputs of Layer 1, namely distance and angle, respectively, and $\{a_q, b_q, c_q\}$ represents one set of consequent parameters associated with the qth node.

Layer 6: Output layer. A node (say, r) corresponding to this layer computes the overall output as the sum of all incoming signals, as indicated below.

$$O_{6r} = \sum_{q=1}^{20} O_{5q}. \tag{24}$$

It is important to mention that the performance of an ANFIS depends on optimal selection of the consequence parameters and the premise parameters (i.e., half base-widths of the input membership function distributions). For this purpose, a binary-coded GA with 610-bits long string is used. The first ten bits (five bits per variable) will carry information about the half base-widths of the two input variables (i.e., V_1 and V_2). Out of the remaining 600-bits, every 15-bits are used to represent a set of consequent parameters (i.e., a_q, b_q, c_q), corresponding to a node lying in the fifth layer, for a particular output. Thus, a GA-string will look as follows.

$$\underbrace{1\cdots0}_{V_1}\ \underbrace{0\cdots1}_{V_2}\ \cdots\cdots\ \underbrace{\underbrace{1\cdots1}_{a_q}\ \underbrace{0\cdots0}_{b_q}\ \underbrace{1\cdots0}_{c_q}}_{\text{consequent parameter of } q\text{th node}}\ \cdots\cdots$$

The fitness of the GA-string is calculated following Eq. (20) used in Approach 4.

3.6. Approach 6: potential field method

Potential field method is one of the most popular conventional techniques for solving the motion planning problems of mobile robot [5,32,34,1]. In this approach, the robot in the configuration space is represented as a particle under the influence of an artificial potential field U. The potential field function can be defined over free surface as the sum of an attractive potential, pulling the robot towards the goal configuration and a repulsive potential pushing the robot away from the obstacle [6]. The artificial potential field function defined at the robot position X is of the form,

$$U(X) = U_{\text{att}}(X) + U_{\text{rep}}(X), \tag{25}$$

where $U_{\text{att}}(X)$ is the attractive potential produced by the goal at X and $U_{\text{rep}}(X)$ indicates the repulsive potential exerted by the obstacle at X.

The performance of the potential field method depends on the chosen artificial potential function. Several potential functions, such as parabolic well, conic well, hyperbolic function, rotational field functions, quadratic potential field function, exponential potential field function, are tried by various investigators [4]. Parabolic and hyperbolic functions have been used for attractive and repulsive potential fields, respectively, as those are found to perform in the optimal sense for solving the similar problem [4]. The attractive potential field $U_{\text{att}}(X)$ can be defined as a parabolic-well

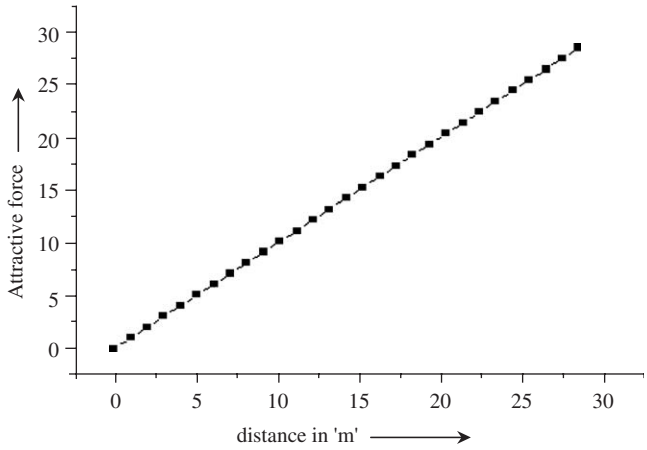

Fig. 14. $F_{\text{att}}(X)$ vs. $d_{\text{goal}}(X)$.

as follows.

$$U_{\text{att}}(X) = \tfrac{1}{2} \times \xi_{\text{att}} \times d^2_{\text{goal}}(X), \tag{26}$$

where ξ_{att} is a positive scaling factor of attractive potential and $d_{\text{goal}}(X)$ denotes the Euclidean distance of the robot from its current position to the goal.

The repulsive potential field $U_{\text{rep}}(X)$ can be expressed as follows:

$$U_{\text{rep}}(X) = -\tfrac{1}{2} \times \xi_{\text{rep}} \times \left[\frac{1}{d_{\text{obs}}(X)} - \frac{1}{d_{\text{obs}}(0)}\right]^2, \tag{27}$$

where ξ_{rep} is a positive scaling factor of repulsive potential and $d_{\text{obs}}(X)$ is the distance of the robot from the obstacle and $d_{\text{obs}}(0)$ is the distance of influence of the obstacle, which is made equal to the center distance between the robot's bounding circle to that of the obstacle.

Attractive potential force can be determined by differentiating the attractive potential with respect to $d_{\text{goal}}(X)$, as given below.

$$F_{\text{att}}(X) = \xi_{\text{att}} \times d_{\text{goal}}(X). \tag{28}$$

Fig. 14 shows a graph between attractive force ($F_{\text{att}}(X)$) and distance ($d_{\text{goal}}(X)$), from which, it can be observed that when the distance between the robot and goal becomes equal to zero, there will not be any attractive force. Moreover, attractive force increases with the distance $d_{\text{goal}}(X)$ in a linear fashion.

Similarly, the repulsive potential force $F_{\text{rep}}(X)$ can be determined as follows:

$$F_{\text{rep}}(X) = \xi_{\text{rep}} \times \frac{[1/d_{\text{obs}}(X) - 1/d_{\text{obs}}(0)]}{d^2_{\text{obs}}(X)}. \tag{29}$$

A graph between repulsive force ($F_{\text{rep}}(X)$) and obstacle distance ($d_{\text{obs}}(X)$) is shown in Fig. 15, from which, it is observed that when the robot is far away from the obstacle, the repulsive force will be less and in such a condition, the robot's motion will not be affected due to the presence of obstacle, whereas if the obstacle is found nearer to the robot, it exerts a repulsive force. The magnitude of the repulsive force will increase, as the distance between the robot and the obstacle decreases. Then, the resultant force $F(X)$ can be calculated by adding $F_{\text{att}}(X)$ with$F_{rep}(X)$ vectorically. In this approach, the acceleration output (a) is considered to be proportional to the magnitude of the resultant potential force $F(X)$ and the deviation output is considered as the angle made between the direction of the resultant potential force and the new reference line joining the CG of the robot and the goal position of the present distance step.

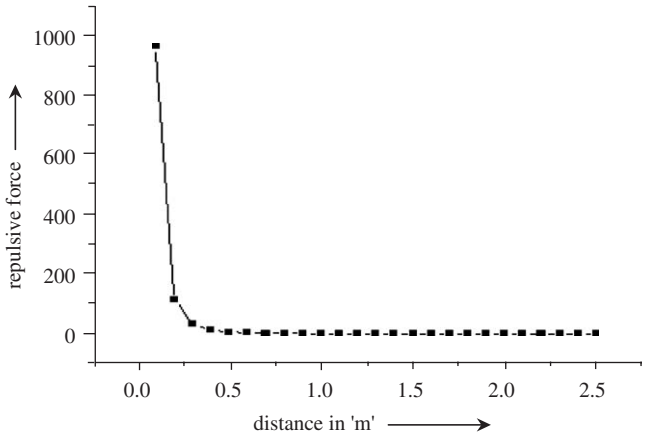

Fig. 15. $F_{\text{rep}}(X)$ vs. $\rho(X)$.

4. Results of computer simulation and discussion

The performances of the developed approaches have been compared among themselves as explained below. The physical parameters of the robot are considered to be as follows:

Size of the robot: 63 mm × 78 mm × 65 mm
Mass of the robot: 0.5 kg
Number of motors: 2
Maximum RPM of the motor: 7200 r.p.m
Power of the motor: 4.55 W
Number of wheels: 2
Diameter and thickness of the wheels: 43 mm
Thickness of the wheels: 7.36 mm
Transmission ratio of the gearbox: 7.5

The cycle time (ΔT) is assumed to be equal to 16 s for the computer simulations. The limits of acceleration (a), velocity (v), steering rate (φ) and the minimum radius of curvature ($\rho_{\min}$) of the robot are set, utilizing the physical parameters mentioned above and the constraints discussed in the Section 2, which are given below:

$$\begin{aligned} &\rho_{\min} \geqslant 0.063\,\text{m},\\ &0.005 \leqslant a \geqslant 0.05\,\text{m/s}^2,\\ &0.007 \leqslant v \geqslant 1\,\text{m/s},\\ &-30^\circ \leqslant \dot{\varphi} \geqslant 30^\circ \end{aligned} \tag{30}$$

In the developed neuro-fuzzy approaches, the FLC is trained off-line, with the help of either a BPNN or a GA, as explained in the last section. The computer simulation is carried out for four different cases. In the first case, four moving obstacles are considered, whereas eight, 12 and 16 moving obstacles are taken into consideration in the 2nd, 3rd and 4th cases, respectively. A field of size of 20 m × 20 m is considered in computer simulations. Considering the physical dimensions of the robot, a hypothetical field of size of 19.95 m × 19.95 m is used to prevent hitting of the robot with the boundary of the field. For tuning of the FLC, a set of 100 training data is created at random, in which initial position, size, velocity and direction of movement of the moving obstacles have been varied. With all such randomly-generated training data, the robot starts moving towards the goal and in a distance step, the amount of corrections necessary in the weight values are calculated. It is to be noted that a batch mode of training has been adopted in this paper. After the tuning of the FLC is over, the performances of the FLCs are compared among themselves and to those of Approaches 1 and 6, in terms of traveling time and their CPU times, for a set of 20 randomly-generated test scenarios. In the proceeding sub-sections, results of both the training as well as test scenarios are explained in detail, for all the four cases.

4.1. Navigation of the robot among 4 moving obstacles

Figs. 16 and 17 show the variation of average deviation error and average acceleration error, respectively with epochs for four obstacles case, during training using Approach 3. Both the deviation error as well as acceleration error are found to decrease with the epochs. The average deviation error reduces continuously, whereas the acceleration error becomes saturated after a particular epoch. This is because of the fact that for acceleration output, the target value is considered to be the maximum acceleration (a fixed value) and once this value has been reached, a further improvement is not possible. The variation of average traveling time is shown in Fig. 18, epoch-wise. The average traveling time is found to be minimum (147.578873 s) at 21st epoch and corresponding to this epoch, the set of updated weight values are considered to be the near-optimal one. The slight fluctuation of average traveling time may be due to the fact that the deviation is a relative quantity and each time it is measured with respect to a new moving reference line, but not a fixed one. It is important to note that average traveling time depends on both average acceleration error as well as deviation error (which is calculated based on a moving reference line). The average traveling time is found to be minimum at 21st epoch, although average deviation error is found not to be the least at that epoch. On the other hand, the average deviation error is seen to be the least at 50th epoch but at this epoch, the traveling time is not found to be the minimum. It happens due to the moving reference line used for calculating the average deviation error. To investigate the nature of randomly-generated 100 training scenarios, the minimum, maximum and mean values of traveling time are recorded, at a particular epoch (say, 20th) and those are found to be equal to 134.9257, 164.89358 and 147.90962 s, respectively. Thus, the standard deviation of traveling time for 100 training scenarios is seen to be equal to 20.348728 s. It indicates that the training scenarios, selected at random, are widely distributed.

In approaches 4 and 5, a GA has been used to improve the performance of the FLC designed based on the Mamdani Approach and Takagi and Sugeno Approach, respectively. As the performance of a GA depends on its parameters, a detailed study is carried out separately for the above approaches. In Approach 4, the best performance is found with the following GA-parameters: crossover probability $= 0.74$, Mutation probability $= 0.008$, population size $= 60$, maximum number of generations $= 90$. On the other hand, in Approach 5, the GA shows the best performance with

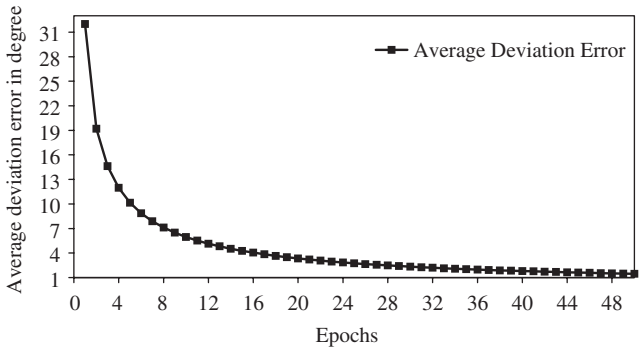

Fig. 16. Average deviation error vs. generation no. (4 obstacles case).

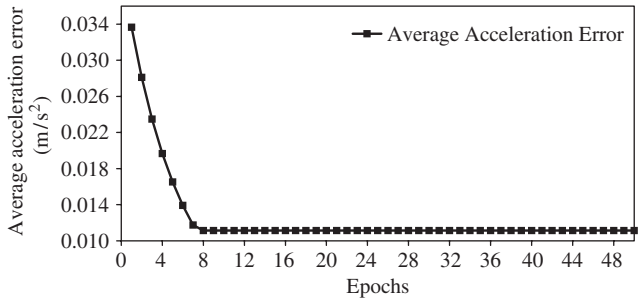

Fig. 17. Average acceleration error vs. generation no. (4 obstacles case).

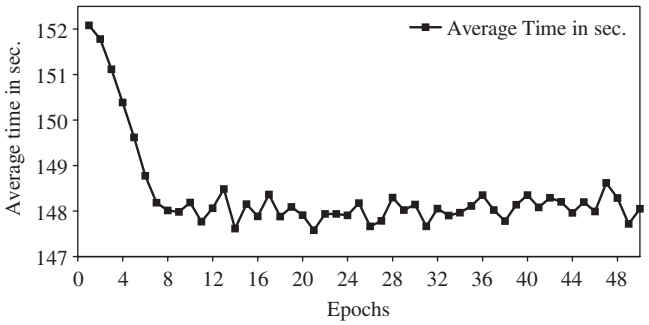

Fig. 18. Average time vs. generation no. (4 obstacle case).

the parameters as follows: crossover probability = 0.76, Mutation probability = 0.001, population size = 180, maximum number of generations = 190.

The performances of Approaches 1, 2, 3, 4, 5 and 6 are compared among themselves in terms of traveling time and CPU time, for 20 randomly-generated test scenarios. Fig. 19 shows a plot of traveling time obtained using all the above approaches for 20 test scenarios. The traveling time taken by the robot in Approach 1 is found to be the worst and the robot could not reach the goal in 14th scenario. It may be due to the fact that there is no motion planner in this approach. Thus, Approach 1 is unable to provide feasible solutions in some of the scenarios, particularly when the robot faces any critical obstacle ahead of it. It is also to be noted that the performance of Approach 2 is found to be worse compared to that of the Approaches 3, 4, 5 and 6, in most of the test scenarios. It could be due to the fact that the manually-constructed FLC used in Approach 2 is not optimal in any sense. Moreover, in most of the cases, the time taken by the robot using different neuro-fuzzy approaches (i.e., Approaches 3, 4, 5) is found to be less compared to that of Approach 6. This may be due to the fact that the performance of Approach 6 deteriorates, when the robot comes closer to the goal, and if a particular obstacle is treated as the critical one in two consecutive distance steps. Again, when the robot faces any obstacle just ahead of it, in Approach 6, it is not changing its direction but reduces the acceleration to a great extent and thus increases the traveling time. Even in such scenarios, it may so happen that the robot is getting zero deviation and zero acceleration (i.e., dead-lock situation). In some of the scenarios, the traveling time values are found to be the same in all the approaches. This may be due to the fact that the robot has not faced any critical obstacle ahead, in any of the distance steps. Approach 4 is found to perform better than Approach 3 in almost all the test scenarios. It could be due to the reason that the gradient-based optimization used in Approach 3 has been replaced by a GA-based optimization in Approach 4. As GA is a population-based search and optimization tool, the chance of its solutions for getting trapped into the local minima is less. Moreover, the performances of Approaches 4 and 5 are found to be comparable. However, Approach 5 has shown a slightly better performance compared to Approach 4. Mean and standard deviation values of traveling time taken by the robot over twenty test scenarios have been calculated and those are found to be equal to (193.015 and 71.205), (165.206 and 39.056), (143.391 and 15.271), (141.113 and 12.412), (140.605 and 13.586) and (145.799 and 16.191) seconds for the Approaches 1, 2, 3, 4, 5 and 6, respectively. Thus, the best and worst performance is recorded by Approaches 5 and 1, respectively. Moreover, Approaches 3–5 have outperformed Approaches 1 and 2 in most of the scenarios. It indicates that a fuzzy logic-based motion planner with an optimal knowledge base might be required to tackle the above problem.

A particular test scenario (say, 7th) is shown in Fig. 20, where the movements of both the robot as well as obstacles are shown. It is to be noted that the chances of collision might be more, when the robot moves with a fixed acceleration in every time step. Thus, the robot in Approach 1 is stopped at a particular position for 2nd to 12th time steps. Moreover, it is noticed that the robot gets less acceleration in Approach 6 (which depends on the nature of potential function), compared to that in Approaches 3–5, when it reaches near to the goal. It is also interesting to note that Approach 5 has identified a time-optimal, collision-free path lying on a separate route compared to that of the other approaches. It could be due to the reasons that the way the rules are represented in Approach 5 is different from the way it is done in Approaches 3 and 4. In this case involving four moving obstacles, CPU time values for Approaches 2, 3, 4, 5 and 6 are found to be equal to 0.0064, 0.0065, 0.002, 0.006 and 0.0006 seconds, respectively.

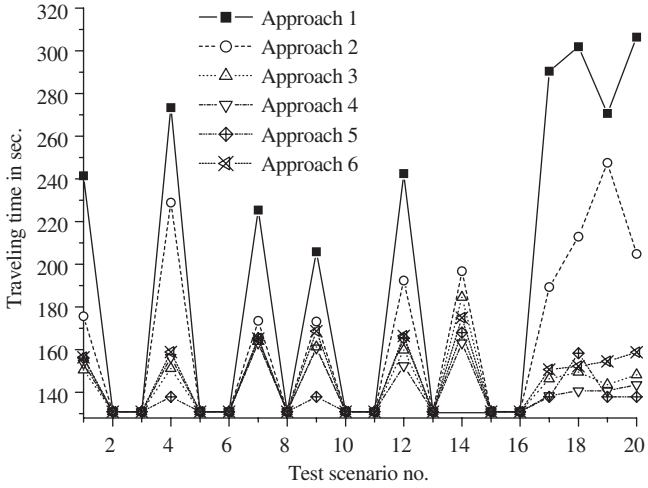

Fig. 19. Traveling time for 4 obstacles case.

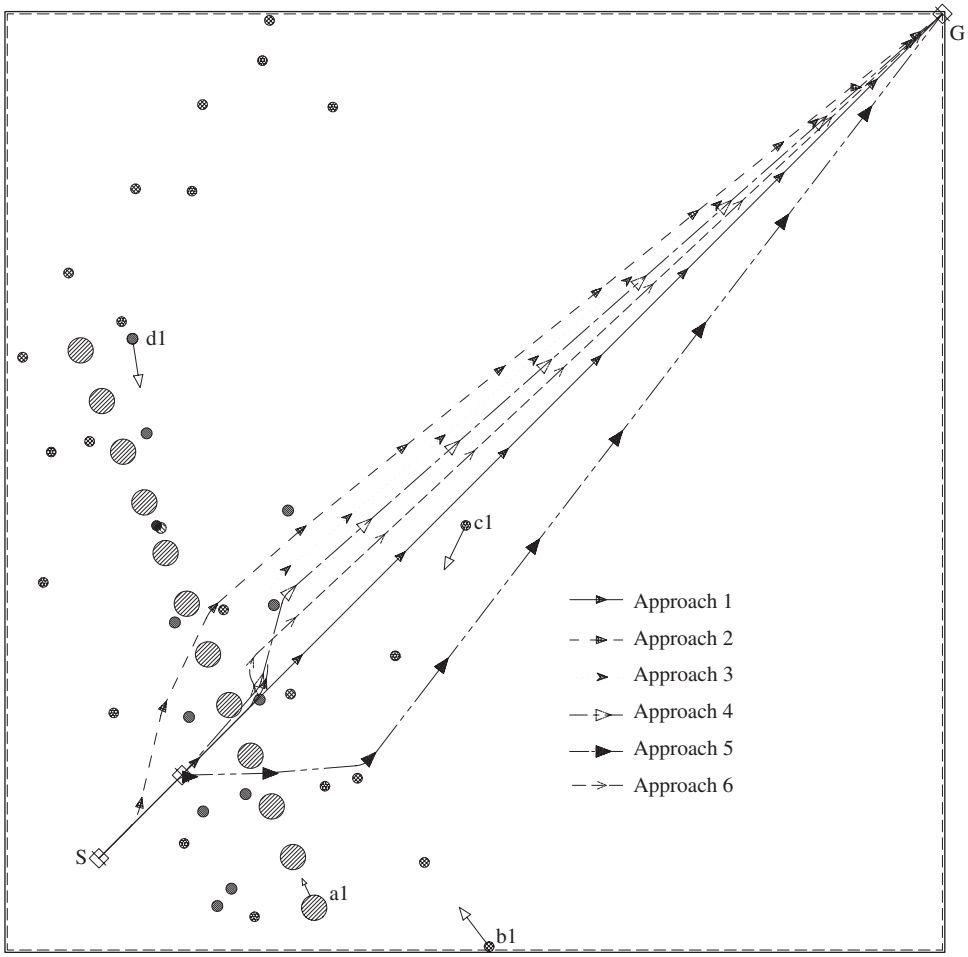

Fig. 20. Movement of the robot among 4 obstacles.

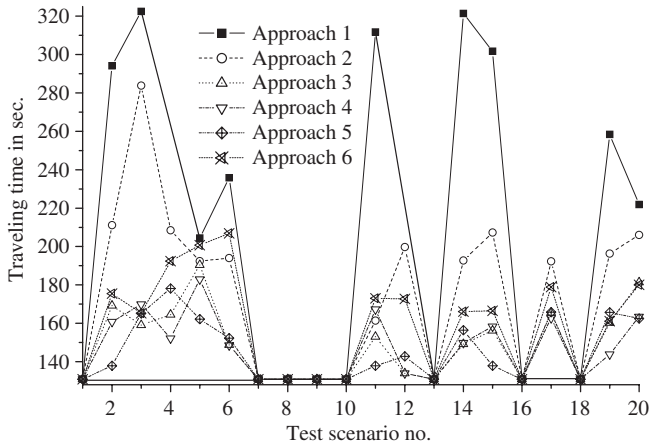

Fig. 21. Traveling time for 8 obstacles case.

4.2. Navigation of the robot among eight moving obstacles

The planning robot tries to find its collision-free path while navigating among eight moving obstacles. The trends of variation of average deviation error and average acceleration error with the epochs are found to be similar to those of four obstacles case. Traveling time values for 20 different test scenarios obtained using the Approaches 1, 2, 3, 4, 5 and 6 are shown in Fig. 21. Table 2 shows the mean and standard deviation values of traveling time calculated over 20 test scenarios, each involving eight moving obstacles. Performances of both the Approaches 1 and 2 are found to be worse compared to those of other approaches, in most of the test scenarios and Approaches 3, 4 and 5 are found to perform better than Approach 6. Moreover, both Approaches 4 and 5 have yielded slightly better results compared to those obtained by Approach 3, and Approach 5 is seen to perform slightly better than Approach 4. Approach 1 is found to fail in three scenarios (out of twenty), in which the robot could not reach the goal in pre-specified twenty time steps. Results of a particular test scenario (say, 4th) are shown in Fig. 22. It is observed that at the 2nd step, the robot considers 1st obstacle to be critical, which is just ahead of itself. Thus, the robot takes left turn in Approaches 2, 3, 4; right turn in Approach 5, but in Approach 6, it moves almost straight with a small acceleration, which forces it to move slowly. Thus, it takes more time to reach the goal in Approach 6. Moreover, in Approach 1, the robot is unable to reach its goal in twenty time steps and it is found to be stationary during the 2nd to 6th, 9th to 11th and 13th to 19th time steps. The CPU time values of Approaches 2, 3, 4, 5 and 6 are seen to be equal to 0.0070, 0.0071, 0.0045, 0.0060 and 0.0017 s, respectively.

4.3. Navigation of the robot among 12 moving obstacles

Twelve obstacles are moving in a 2D plane and the robot will have to find its collision-free, time-optimal path starting from an initial position to reach a fixed destination. After the training of the neuro-fuzzy approaches is over, the performances of all the approaches are compared for twenty randomly-generated test scenarios. It is noticed that the performances of all the optimized FLC-based approaches (i.e., Approaches 3, 4, 5) are better than that of the other approaches (i.e., Approaches 1, 2, 6, refer to Fig. 23). The similar observations have also been made from Table 2. It is to be noted that Approach 1 is found to fail to reach the goal in seven scenarios out of a total of twenty, in this case. It is also interesting to note that Approach 4 has shown a slightly better performance compared to that of Approach 5. The complete paths of the robot obtained by using all these approaches, for a particular test scenario (say, 3rd), are shown in Fig. 24. The robot in Approach 1, is seen to move up to the point (18.369, 18.369), instead of reaching the goal, in twenty time steps. The CPU times of all these approaches are also determined and those are found to be equal to 0.0075, 0.0081, 0.0091, 0.0070 and 0.0018 s for the Approaches 2, 3, 4, 5 and 6, respectively.

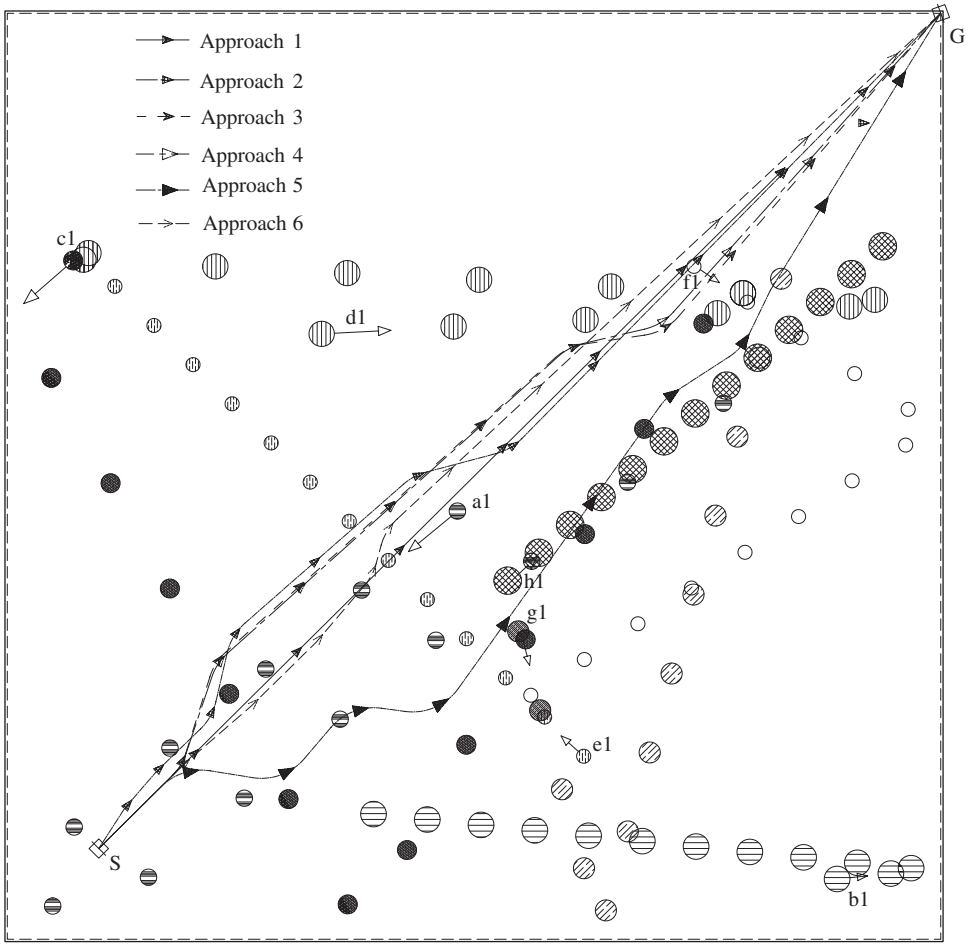

Fig. 22. Movement of the robot among 8 obstacles.

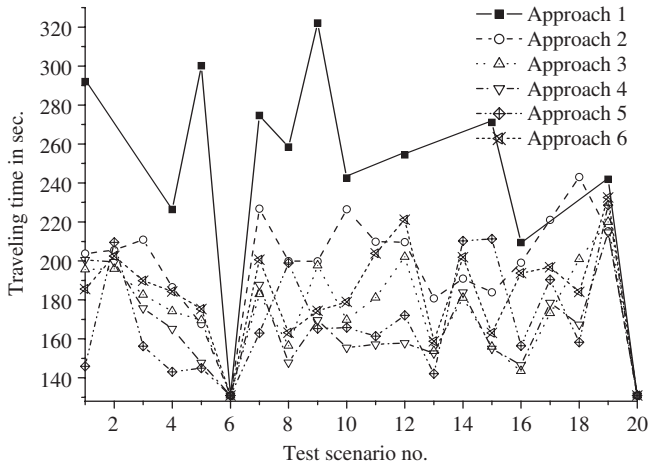

Fig. 23. Traveling time for 12 obstacles case.

Table 2
Mean and standard deviation of traveling time (s) values for different obstacles cases

Obs. case	Approach 1		Approach 2		Approach 3		Approach 4		Approach 5		Approach 6	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
8	206.432	80.654	174.652	42.518	148.936	18.932	147.047	16.765	145.598	16.017	159.328	26.249
12	242.688	58.168	197.122	28.676	174.915	24.173	166.259	22.706	169.275	29.027	183.615	25.849
16	249.029	68.089	190.951	26.656	179.178	21.865	171.861	21.835	171.016	21.520	182.328	35.047

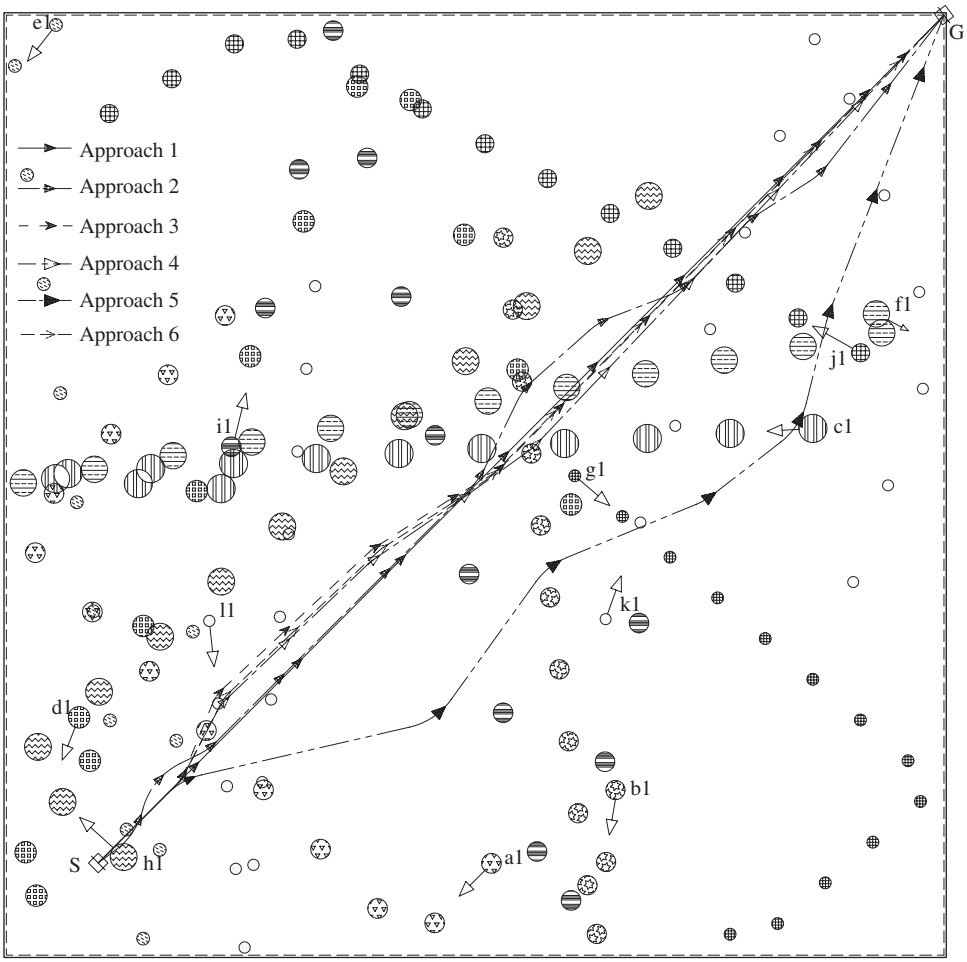

Fig. 24. Movement of the robot among 12 obstacles.

4.4. Navigation of the robot among 16 moving obstacles

Collision-free, time-optimal path generation problems of a car-like robot moving among sixteen obstacles have been considered in this case. Traveling time values of the robot obtained by using six different approaches, for solving twenty randomly-generated test scenarios, are shown in Fig. 25. It is noticed that in most of the test scenarios, the GA-based neuro-fuzzy approaches (i.e., Approaches 4 and 5) perform better than the Approaches 1, 2, 3 and 6. The similar information has been obtained from Table 2 also. In nine different scenarios, the robot is found not to be reaching the goal point. A typical test scenario (say, 4th) is shown in Fig. 26, in which the robot's path obtained by using all the

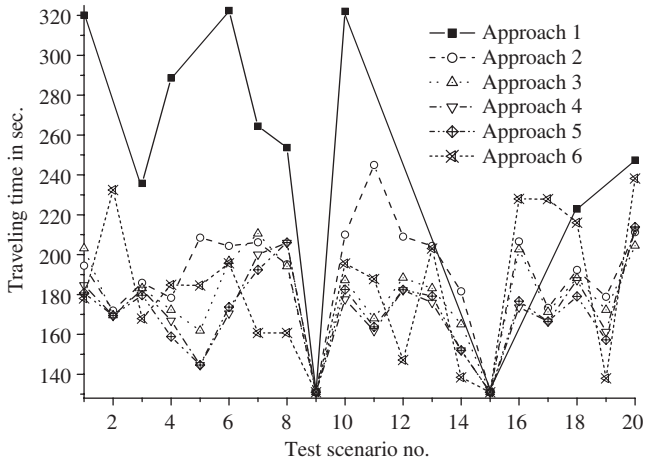

Fig. 25. Traveling time for 16 obstacles case.

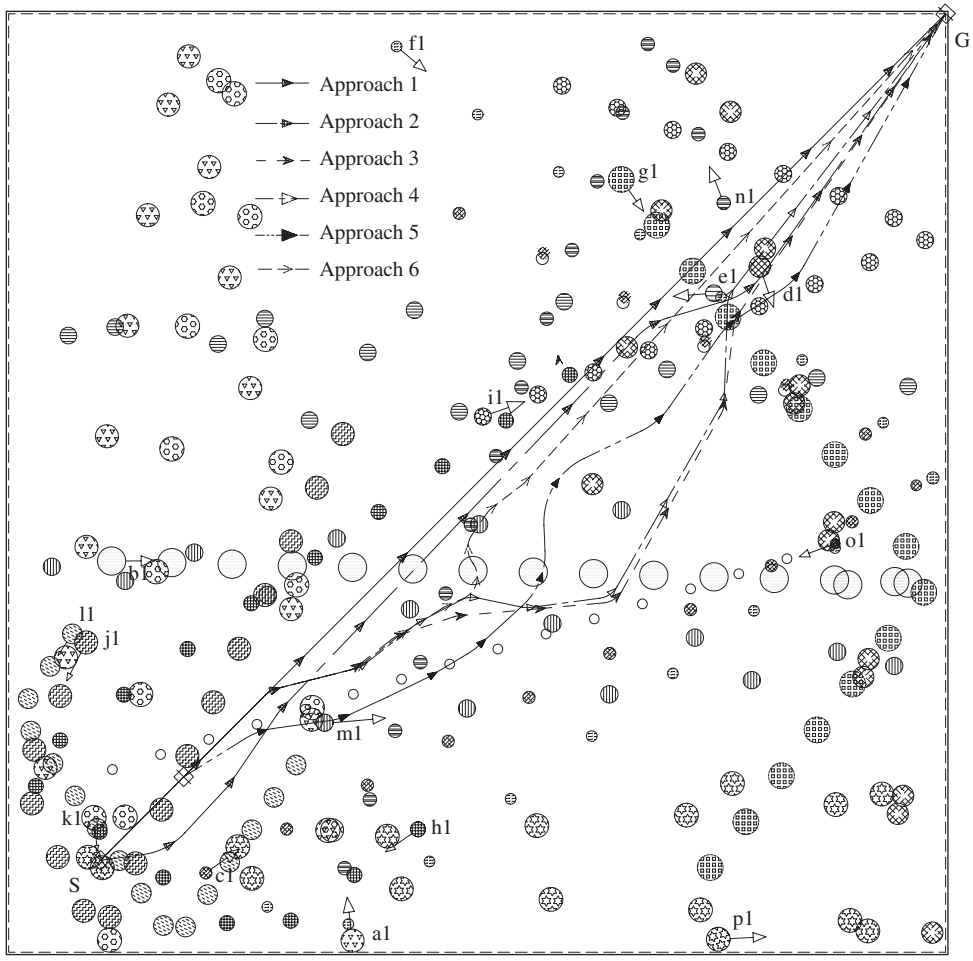

Fig. 26. Movement of the robot among 16 obstacles.

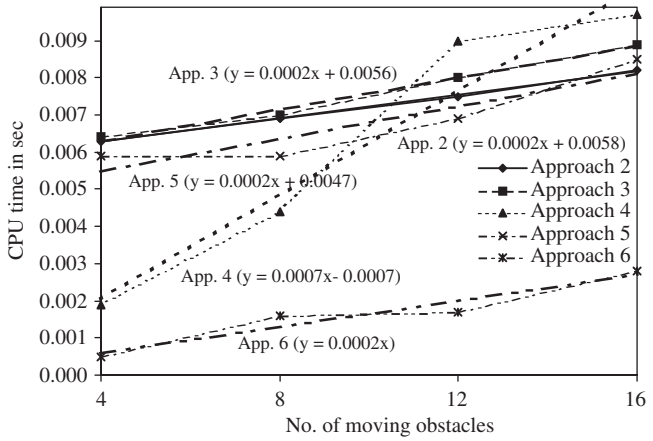

Fig. 27. Total CPU time (in s) for different obstacle cases.

approaches and the obstacles' movements are shown. It is important to note that in Approach 1, the robot is forced to be stationary during the 2nd to 5th and 10th to 13th time steps. The CPU time values of the Approaches 2, 3, 4, 5 and 6 are found to be equal to 0.0083, 0.0090, 0.0098, 0.0086 and 0.0029 s, respectively.

In the present work, navigation schemes of a car-like robot have been developed based on a few important assumptions, one of which is such that only one obstacle is considered to be critical in a time step and no two obstacles are allowed to overlap. Moreover, time step is assumed to be a fixed quantity, whereas the distance traversed by the robot in a time step may vary depending on its acceleration. In Approach 1, since the robot moves with the maximum possible acceleration, it will reach its hypothetical boundary in every time step. Under these circumstances, the robot might have to face a number of critical obstacles in a particular time step, and finding a collision-free path could be difficult in Approach 1, as there is no motion planning scheme (due to which the distance steps could be different in different time steps) but a collision-avoidance scheme only. As a result, the robot may not be able to find collision-free path in a number of consecutive time steps and will be stopped at that position. Therefore, the robot in Approach 1 is unable to reach its goal within the pre-specified maximum number of time steps and the number of such incidences is found to increase with the scenarios involving more number of moving obstacles. Thus, the motion planning schemes, particularly those developed based on the neuro-fuzzy approaches, might be necessary for solving the navigation problems of a robot.

4.5. CPU time comparison

The CPU time values for all the four above-mentioned cases, as obtained by using Approaches 2, 3, 4, 5 and 6, are noted and the best-fit line is drawn for each approach based on the least square error method (refer to Fig. 27). It is important to mention that the experiments are conducted on a Pentium - IV PC. The potential field method is found to be the fastest of all. However, the slope of the best-fit line for Approach 4 has appeared to be a slightly higher compared to that of the other three approaches. It indicates that with an increase in environmental complexity, computational complexity of Approach 4 will increase at a faster rate than that of the other three approaches. However, Approach 4 is found to be faster than Approach 5, for particularly the cases involving the less number of obstacles.

4.6. Comparison of the present work with others work

In this work, some neuro-fuzzy systems have been developed to fine tune the performance of an FLC and their performances are tested for solving navigation problems of a car-like mobile robot. Fraichard et al. [7] successfully developed a fuzzy control mechanism known as execution monitor (EM), for a car-like vehicle. They had implemented and tested their approach, on a real computer-controlled car. But, they did not use the optimized FLC. Marichal et al.

[28] proposed a neuro-fuzzy approach to guide a mobile robot. They considered the least mean squared algorithm for the learning purposes and Kohonen's self-organizing feature map algorithm had been applied to obtain the initial number of fuzzy rules and membership function centers. But, in their approach, they did not optimize the traveling time. Moreover, they considered only two kinds of static obstacles (rectangular and corner shaped), but not the moving obstacles. Li et al. [24] developed a neuro-fuzzy system architecture for behavior-based control of a mobile robot. In their approach, an NN was used to understand the environments and behavior fusion was done by a fuzzy logic algorithm. But, the performance of their technique was tested for static obstacles only. Song and Sheen [36] suggested a heuristic fuzzy-neuro network for pattern mapping between quantized sensory data and velocity commands of a mobile robot. In their approach, fuzzy Kohonen clustering network was used to build the desired mapping between perception and motion of the robot. But, the developed method might not give the time-optimal path of the robot. In the present work, some neuro-fuzzy approaches have been developed, to generate time-optimal, collision-free path of a car-like mobile robot while navigating in the presence of some moving obstacles.

5. Concluding remarks

In order to navigate a car-like robot, in a dynamic environment (among moving obstacles), a time-optimal (at least near-optimal), collision-free path is to be determined on-line, after satisfying both the static as well as dynamic constraints. Six different approaches, namely Approach 1 (default behavior), Approach 2 (manually-constructed FLC), Approach 3 (NN-tuned FLC), Approach 4 (GANNFLC), Approach 5 (GA-tuned ANFIS) and Approach 6 (conventional potential field method) have been developed to solve the above problem. In Approaches 2, 3 and 4, a Mamdani type fuzzy logic controller is developed and the FLC in Approaches 3 and 4 have been optimized using a back-propagation algorithm and a GA, respectively. On the other hand, in Approach 5, a Takagi and Sugeno-type neuro-fuzzy approach optimized by a GA, has been developed. Once the tuning of the FLC is done either by using a BPNN or a GA off-line, it can be utilized to solve the navigation problems on-line.

The performance of Approach 1 is found to be the worst and in some of the scenarios, the robot has failed to reach the goal. Moreover, the number of failures is found to increase with the scenarios involving more number of moving obstacles. It could be due to the fact that there is no motion planner or decision maker in Approach 1. Thus, the robot in Approach 1 is unable to provide feasible solution, in most of the scenarios. Moreover, the performance of the FLC used in Approach 2 is appeared to be worse compared to the optimized FLC-based approaches (i.e., Approaches 3–5). It also indicates that optimization of the knowledge base of the FLC is really necessary. Approaches 3, 4 and 5 are found to perform better than the conventional potential field method (i.e., Approach 6) in most of the test scenarios. It may happen because in Approach 6, there is no optimization module in-built and there is a chance of its solution for getting trapped into local minima. Moreover, in Approach 6, there is a possibility that the same obstacle will be treated as the most critical one, more than once in successive steps. In Approach 3, the knowledge base of the FLC has been optimized by using a steepest descent method, in which a penalty has been given to ensure collision-free movement after satisfying the kinematic and dynamic constraints of the robot. The difference in performance of this approach from that of Approaches 4 and 5 might have been noticed due to the following reasons: (i) penalty added in Approach 3 is not the same with that considered in Approaches 4 and 5, due to its difficulty in implementation, (ii) gradient-based search used in Approach 3 has been replaced by a GA-based search. It is to be noted that Approach 4 has performed better than Approach 3, in most of the test scenarios. Moreover, the performances of both the GA-optimized neuro-fuzzy approaches are found to be comparable, but a slightly better performance is observed in Approach 5 compared to that of Approach 4. Computational complexities of all the approaches have also been compared, in the present work. Potential field method is found to be the fastest of all approaches, computationally. However, all the neuro-fuzzy approaches have provided adaptive solutions to the present problem. Moreover, for the cases involving less number of moving obstacles, Approach 4 is found to be slightly computational faster compared to Approach 5. But, as the number of obstacles increases, the CPU time of Approach 5 is seen to be less compared to that of Approach 4. It indicates that Approach 4 might not be suitable for a highly cluttered environment, as the defuzzification module of Approach 4 is computationally expensive.

As the performance of the neuro-fuzzy approaches depends on the training data, it may not work well, particularly when the test scenarios are widely different from the training scenarios.

6. Scope for future work

The present work can be extended in a number of ways and some of these are as follows:

- The performance of an FLC depends on the rule base and it has been designed manually based on the authors' knowledge of the problem in Approaches 3 and 4. But, it may not be the optimal one. In the present work, an attempt has been made to select the good rules during optimization from a set of pre-defined rules. However, the membership function distributions of the variables have been determined automatically in the present paper. Thus, a method for automatic design of the KB of an FLC will be developed, in which the whole responsibility of designing a good KB will be given to the GA, which could be philosophically similar to the work of Marichal et al. [28].
- The developed neuro-fuzzy approaches are tested on computer simulations. It will be interesting to test their performances on a real robot.
- In the present work, the membership function distributions are assumed to be triangular. Other smoother distribution patterns like Gaussian or exponential may be used to have smooth control of the robot. In such cases, computational complexity is supposed to be higher.
- In the present work, a fuzzy logic-based controller has been developed, but in future, an NN-based controller can be designed to solve the similar problems. Moreover, the performance of both the controllers may be compared in terms of traveling time and their CPU times.
- The neuro-fuzzy controllers are developed and tested for a single robot navigating in dynamic environments. It will be more interesting, if they can be used to solve coordination problems of multiple robots working in the same workspace. Since all the robots will try to find optimal/near-optimal collision-free path, the coordination effect among the robots will be a great issue in this context.

Acknowledgements

This work is supported by the Department of Science and Technology, Govt. of India (Sanction No. SR/S3/RM/28/2003 Dt. 12.12.2003). The help and suggestions of Dr. A. Roy Choudhury, IIT Kharagpur, India, are gratefully acknowledged. The authors are profoundly grateful to the anonymous reviewers, whose suggestions helped a lot to improve the quality of the paper.

References

[1] F. Abdessemed, K. Benmahammed, E. Monacelli, A Fuzzy-based reactive controller for a non-holonomic mobile robot, Robot. Autonom. Systems 47 (2004) 31–46.
[2] M.R. Akbarzadeh-T, K. Kumbla, E. Tunstel, M. Jamshidi, Soft computing for autonomous robotic systems, Comput. Elect. Eng. 26 (2000) 5–32.
[3] A. Bemporad, A.D. Luca, G. Oriolo, Local incremental planning for a car-like robot navigating among obstacles, Proceedings of IEEE Conf. on Robotics and Automation, Minneapolis, Minnesota, April 1996, pp. 1205–1211.
[4] D.K. Biswas, Path planning of multiple robot working in the same workspace—potential field approach, M. Tech thesis, REC Durgapur, India, 2003, pp 394–421.
[5] D. Feng, H. Krogh, H. Bruce, Dynamic steering control of conventionally-steered mobile robots, Proc. IEEE Conf. Robotics Automation, 1990, pp. 390–395.
[6] T. Fraichard, Dynamic trajectory planning with dynamic constraints: a 'state-time space' approach, Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Yokohama, Japan, July 1993, pp. 1393–1400.
[7] T. Fraichard, P. Garnier, Fuzzy control to drive car-like vehicles, Robot. Autonom. Systems 34 (2001) 1–22.
[8] T. Fraichard, A. Scheuer, Car-like robots and moving obstacles, Proc. IEEE Conf. on Robotics and Automation 1996, pp. 64–69.
[9] T. Fukuda, Y. Hasegawa, K. Shimojima, F. Saito, Reinforcement learning method for generating fuzzy controller, IEEE Int. Conf. on Evolutionary Computation, 1995, pp. 273–278.
[10] P. Garnier, T. Fraichard, A fuzzy motion controller for a car-like vehicle. Proc. IROS, IEEE 1996.
[11] J. Godjevac, N. Steele, Adaptive Neuro-Fuzzy controller for navigation of a mobile robot, AT'96, Int. Symp. Neuro-Fuzzy systems, August 1996, pp. 111–119.
[12] D. Gu, H. Hu, Neural predictive control for a car-like mobile robot, Robo. Autonom. Systems 39 (2002) 73–86.
[13] C.M. Higgins, R.M. Goodman, Learning fuzzy rule-based neural networks for function approximation, Proc. IJCNN, vol. 1, 1992, pp. 251–256.
[14] S. Hong, S. Kim, K. Park, J. Lee, Local motion planner for nonholonomic mobile robots in the presence of the unknown obstacles, Proc. IEEE Conf. on Robotics and Automation, Minneapolis, Minnesota, April 1996, pp. 1212–1216.
[15] S. Horikawa, T. Furuhashi, et al., A study on fuzzy modeling using fuzzy neural networks, Proc. IFES, 1991, pp. 562–573.
[16] N.B. Hui, D.K. Pratihar, Time-optimal, collision-free navigation of a car-like robot using a genetic-neuro-fuzzy systems, Proc. Int. Conf. Theoretical & Computational Mechanics, IIT Kharagpur, India, 2004, pp. 96–99.

[17] J.S.R. Jang, Self-learning fuzzy controllers based on temporal back propagation, IEEE Trans. Neural Networks 3 (5) (1992).
[18] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference systems, IEEE Trans. Systems, Man Cybernet. 23 (3) (1993) 665–685.
[19] J.-S.R. Jang, C.-T. Sun, E. Mizutani, Neuro-fuzzy and soft computing, Prentice-Hall, Englewood, NJ, 1997.
[20] K.J. Kyriakopoulos, P. Kakambouras, N.J. Krikelis, Navigation of nonholonomic vehicles in complex environments with potential fields and tracking, Proc. IEEE Int. Conf. on Robotics and Automation, Minneapolis, Minnesoia, April 1996, pp. 3389–3394.
[21] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Dordrecht, 1991.
[22] C.S. Lee-George, Neuro-fuzzy systems, Handbook of Industrial Robotics, second ed., 1999, pp. 394–421.
[23] W. Li, C. Ma, F.M. Wahl, A Neuro-fuzzy system architecture for behavior based control of a mobile robot in unknown environments, Fuzzy Sets and Systems 87 (1997) 133–140.
[24] C.T. Lin, C.S.G. Lee, Neural network based fuzzy logic control and decision system, IEEE Trans. Comput. 40 (12) (1991).
[25] A.D. Luca, G. Oriolo, Local incremental planning for non-holonomic mobile robots, 1994 IEEE Int. Conference on Robotics and Automation, San Diego, CA, 1994, pp. 104–110.
[26] H. Maaref, C. Barret, Sensor- based navigation of a mobile robot in an indoor environment, Robo. Autonom. Systems 38 (2002) 1–18.
[27] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, Int. J. Man–Mach. Stud. 7 (1) (1975) 1–13.
[28] G.N. Marichal, L. Acosta, L. Moreno, J.A. Mendez, J.J. Rodrigo, M. Sigut, Obstacle avoidance for a mobile robot: a neuro fuzzy approach, Fuzzy Sets and Systems 124 (2001) 171–179.
[29] D. Nauck, F. Klawonn, R. Kruse, Combining neural networks and fuzzy controllers, Presentation at FLAI'93, Linz, Austria, June 28–July 2, 1993.
[30] D. Nauck, F. Klawonn, R. Kruse, Combining neural networks and fuzzy controllers, in: E.-P. Klement, W. Slany (Eds.), Fuzzy Logic in Artificial Intelligence, Springer, Berlin, 1993.
[31] H. Nomura, I. Hayashi, N. Wakami, A learning method of fuzzy inference rules by descent method, Proc. IEEE Int. Conf. Fuzzy Systems, San Diego, CA, 1992.
[32] D.K. Pratihar, Path and gait generation of legged robots using GA-fuzzy approach, Ph.D. Thesis, IIT Kanpur, 2000.
[33] D.K. Pratihar, Algorithmic and soft computing approaches to robot motion planning, Mach. Intell. Rob. Control 5 (1) (2003) 1–16.
[34] D.K. Pratihar, Evolutionary robotics-a review, Sadhana 28 (6) (2003) 999–1009.
[35] D.K. Pratihar, W. Bibel, Near-optimal, collision-free path generation for multiple robots working in the same workspace using a genetic-fuzzy system, Machine Intell. Robot. Control 5 (2) (2003) 45–58.
[36] K.T. Song, L.H. Sheen, Heuristic fuzzy-neuro network and its application to reactive navigation of a mobile robot, Fuzzy Sets and Systems 110 (2000) 331–340.
[37] S.M. Sulzberger, N.N.T. Gurman, S.J. Vestli, FUN: Optimization of fuzzy rule based systems using neural networks, IEEE Int. Conf. Neural Networks 1 (April 1993) 312–316.
[38] H. Takagi, I. Hayashi, NN-driven fuzzy reasoning, Int. J. Approx. Reason. 5 (1991) 191–212.
[39] T. Takagi, M. Sugeno, Derivation of fuzzy control rules from human operator's control action, Proc. IFAC Symp. On Fuzzy Information, Knowledge Representation and Decision Analysis, Marseilles, France, 1983, pp. 55–60.
[40] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Systems, Man Cybern. 15 (1985) 116–132.
[41] A.M. Tang, C. Quek, G.S. Ng, GA-TSKfnn: parameters tuning of fuzzy neural network using genetic algorithms, Expert Systems Appl. 29 (2005) 769–781.
[42] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Systems, Man Cybern. 22 (6) (1992) 1414–1427.
[43] B. Yegnanarayana, Artificial Neural Networks, Prentice-Hall, New Delhi.
[44] L.A. Zadeh, Fuzzy sets, Inform. Control 8 (1965) 338–353.