J Intell Robot Syst (2008) 51:333–368
DOI 10.1007/s10846-007-9190-5

UNMANNED SYSTEM PAPER

# Soft Computing-Based Navigation Schemes for a Real Wheeled Robot Moving Among Static Obstacles

**Nirmal Baran Hui · Dilip Kumar Pratihar**

Received: 23 December 2006 / Accepted: 12 October 2007 /
Published online: 21 December 2007


**Abstract** Collision-free, time-optimal navigation of a real wheeled robot in the presence of some static obstacles is undertaken in the present study. Two soft computing-based approaches, namely genetic-fuzzy system and genetic-neural system and a conventional potential field approach have been developed for this purpose. Training is given to the soft computing-based navigation schemes off-line and the performance of the optimal motion planner is tested on a real robot. A CCD camera is used to collect information of the environment. After processing the collected data, the communication between the robot and the host computer is obtained with the help of a radio-frequency module. Both the soft computing-based approaches are found to perform better than the potential field method in terms of the traveling time taken by the robot. Moreover, the performance of fuzzy logic-based motion planner is found to be comparable with that of neural network-based motion planner, although the training of the former is seen to be computationally less expensive than the latter. Sometimes the potential field method is unable to yield any feasible solution, specifically when the obstacle is found to be just ahead of the robot, whereas soft computing-based approaches have tackled such a situation well.

**Keywords** Car-like robot · Navigation · Real experiment · Fuzzy logic · Neural network · Genetic algorithm · Potential field method

N. B. Hui · D. K. Pratihar (✉)
Department of Mechanical Engineering, Indian Institute of Technology, Kharagpur 721 302, India
e-mail: dkpra@mech.iitkgp.ernet.in

N. B. Hui
e-mail: nirmal@mech.iitkgp.ernet.in

**List of symbols**

| | |
|---|---|
| $\mu_f$ | Coefficient of sliding friction |
| $\dot{\phi}$ | Rate of change of steering angle during turning, rad/s |
| $\xi_{att}$ | Positive scaling factor for attractive potential |
| $\xi_{rep}$ | Positive scaling factor for repulsive potential |
| $\rho$ | Instantaneous radius of curvature of the CG of the robot during turning, mm |
| $\theta$ | Angle between the $X$-axis and the main axis of the robot, degrees |
| $\theta_1$ | Deviation of the robot, degrees |
| $a$ | Tangential acceleration of the robot, mm/s$^2$ |
| $C_p$ | Constant of the $p$th layer activation function |
| $d_{goal}$ | Distance between the robot and the goal, mm |
| $d_{min}$ | Minimum distance required by the robot to reach the goal with zero velocity, mm |
| $F(X)$ | Potential force function |
| $M$ | Mass of the robot, Kg |
| $N_m$ | Maximum angular speed of the wheels of the robot, r.p.m. |
| $P$ | Power of the motor, Watt |
| $r$ | Radius of the wheels, mm |
| $T$ | Traveling time, s |
| $\Delta T$ | Time step, s |
| $U(X)$ | Artificial potential energy function |
| $v$ | Tangential velocity of the CG of the robot, mm/s |
| $v_{ij}(t)$ | Connecting weights between $i$th input neuron and $j$th hidden neuron at iteration t |
| $w_{jk}(t)$ | Connecting weights between $j$th hidden neuron and $k$th output neuron at iteration t |
| $(\dot{x}, \dot{y})$ | Components of tangential velocity |

**List of abbreviations**

| | |
|---|---|
| AH | Ahead |
| AL | Ahead Left |
| AR | Ahead Right |
| BPNN | Back-Propagation Neural Network |
| CG | Center of Gravity |
| 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 |
| PFM | Potential Field Method |

| | |
|---|---|
| RT | Right |
| VF | Very Far |
| VH | Very High |
| VL | Very Low |
| VN | Very Near |

## 1 Introduction

The field of mobile robotics is undergoing a major shift in scope and dimension. Prime emphasis is given to increase the autonomy power of the robot. An autonomous and intelligent robot should be able to react to its environment without any human intervention. Thus, it should have a real-time sensing assembly, an intelligent motion planner and precise actuators. Motion planning of a car-like robot is found to be difficult, due to its kinematic and dynamic constraints [1]. Quite a few researchers have developed some suitable methods of motion planning for this purpose. These are working based on either algorithmic approaches or some soft computing techniques [2]. Algorithmic methods include both graph-based as well as analytical approaches. Latombe [1] provides an extensive survey on different algorithmic methods of robot motion planning. Visibility graph [3], Voronoi diagram [4], tangent graph [5], freeway net [6], cell decomposition [7], probabilistic road-map [8] are the widely used graph-based methods for solving the problem of robot motion planning among static obstacles. On the other hand, potential field method proposed by Khatib [9], is the most popular one out of all analytical methods for solving the similar type of problems [1, 10, 11]. But, all these algorithmic methods suffer from the following drawbacks: (a) not all the approaches are computationally tractable and thus, they may not be suitable for on-line implementations, (b) one method may be suitable for solving a particular type of problem and no versatile technique is available, (c) most of the approaches do not have any in-built optimization module and as a result of which, the generated path may not be optimal in any sense. It is important to mention that potential field method has got the maximum popularity among all the algorithmic approaches. It may be due to its elegant mathematical analysis and simplicity. However, it has the following disadvantages [12]: (a) It may not be able to provide with a completely local minima-free information, such problems are seen to occur, when the robot navigates among concave obstacles, (b) It may not find any feasible path for the robot, when it is moving among the obstacles lying just in front of it or when it is moving among some closely spaced obstacles. It may also happen, when the attractive potential balances the repulsive potential or the magnitude of the resultant potential comes out to be negligible. To overcome some of these drawbacks, Borenstein and Koren [13, 14] developed two modified versions of potential field approach, namely virtual force field and vector field histogram method for real-time obstacle avoidance of mobile robots. Thereafter, Slack [15] developed a navigation template method that added a circular field around any obstacle. But, all these modified potential field methods could not adjust their parameters to keep the robot at a safe distance from an obstacle and hence were ineffective in avoiding collision, especially against an obstacle lying in front of the robot. Thus, it is necessary to develop some efficient, adaptive, flexible and computationally tractable algorithm for solving the motion planning problems of mobile robots.

Soft computing-based approaches are becoming more and more popular, now-a-day, for solving navigation problems of the mobile robots [2]. It is due to the fact that the computational complexity of such methods is expected to be low, as most of them are heuristic in nature and an exact mathematical formulation of the problem is not required. Moreover, these methods could find some adaptive solutions to the problem. Both fuzzy logic-based as well as neural network-based navigation schemes are available in the literature. Some of these schemes are mentioned below.

Fraichard and Garnier [16], Abdessemed et al. [17] used a fuzzy logic controller (FLC) for planning collision-free motion of a car-like robot. The performance of an FLC depends on the selection of membership function distributions (known as data base) and its rule base. But, in most of the fuzzy control systems, fuzzy if-then rules are designed by human experts, who may sometimes find it difficult to express his/her actions or may decide on a subconscious level. Various investigators tried to optimize both the rule base as well as data base of the FLC, either separately or simultaneously. Several techniques are available in the literature for the said purpose. Some of the these works related to motion planning of car-like robots are mentioned below. Marichal et al. [18] proposed a mobile robot guiding mechanism based on a neuro-fuzzy approach. In their approach, a least mean squared algorithm was applied for the learning purposes and Kohonen's self organizing feature map algorithm was considered to obtain the initial number of fuzzy rules and fuzzy membership function centers. But, neither did they optimize the traveling time nor the approach was tested in a dynamic environment. Song and Sheen [19] suggested a pattern recognition approach based on fuzzy-neuro network for reactive navigation of a car-like robot. Li et al. [20] developed a neuro-fuzzy architecture for behavior-based control of a car-like robot, that navigates among static obstacles. Maaref and Baret [21] suggested a self-tunable fuzzy inference system (STFIS) for controlling the angular and linear speeds of a mobile robot. They followed an on-line optimization of the fuzzy inference system (FIS) by using a back-propagation training algorithm, which may suffer from the local-minima problem. Pratihar and Bibel [22] designed an FLC automatically by using a GA, that was intended to solve dynamic motion planning problem of multiple mobile robots working in the same environment. Later on, Hui and Pratihar [23] also followed an automatic design method of FLC, in which the whole task of designing an FLC was given to a GA. The GA evolves a suitable knowledge base of that FLC through the interactions between the robot and the environment. The main advantage of this method lies in the fact that the designer may not need to have the complete knowledge of the problem to be solved. Moreover, the entire optimization process was carried out off-line, thus once trained, it might be suitable for solving on-line navigation problems of a real robot. However, the performance of this method was verified through computer simulations.

Neural networks (NNs) had also been used by some other researchers for solving the said problem. In this connection, work of Yang and Meng [24], Floreano and Mondada [25], Nolfi and Parsi [26], Yamada [27], Pal and Kar [28], Gu and Hu [29] are important to mention. However, the performance of an NN depends on its architecture and connecting synaptic weights, optimal selection of which is a tedious job. A variety of tools based on supervised and reinforcement learning algorithms had been used by a few investigators for this purpose. Back-propagation algorithm is the most popular method to optimize the NN, but it may have the local minimum problem. Simulated annealing (SA) [30], genetic programming (GP) [31],

genetic algorithms (GA) [32] have also been used by some researchers for the said purpose. It is to be noted that GAs along with NN have added a new dimension to the field of robotic research, namely evolutionary robotics [33]. Here, a suitable NN architecture is evolved by using a GA through proper interactions with the environment. Pratihar [33] provided a comprehensive review on various aspects of evolutionary robotics. After realizing the advantages of GA-NN approach, Hui and Pratihar [34] studied its performance for solving the navigation problems of a car-like robot through computer simulations.

Most of the above researchers tested their motion planning algorithms through computer simulations. However, more recently, the importance of conducting experiments using the real robot to test the performance of motion planner has been felt by various investigators [35–38]. For the above purpose, the motion planner will have to depend on the sensors and/or cameras for collecting information of the environment. The choice of the sensors plays an important role in this regard. Sonar and laser sensors are found to be the most widely used ones for obstacle detection. Many works had been reported related to this, such as the CMU Navlab [39], navigation systems developed at the University of Maryland [40], University of Bonn [41], robot Khepara [25] and others. However, the main drawback of the sonar or laser sensors lies in the fact that one sensor is required for one distance measurement, that is, in order to obtain a complete picture of the environment around the vehicle, a number of sensors must be used. Moreover, to achieve the accuracy in detection, they will have to be placed perpendicular to the target. CCD cameras are also found to be useful for scene modeling, obstacle detection and representation of them. Quite a few camera-based navigation systems are available in the literature, such as the work done by Chen and Tsai [42], Ohya et al. [43], Zhang et al. [44], Winters and Victor [45], Choi and Lee [46]. But, in most of these approaches there is no separate motion planning scheme of the robot. The motion of the robot is determined directly based on the vision data. Moreover, vision system generates a substantial amount of data and processing of which may lead to a large computational load on a mobile robot. Thus, to build a fast and flexible mobile robot, camera-based vision system will have to be clubbed with its navigation scheme, on-line.

Stability is one of the most important criteria to be checked of a control system. It is, however, difficult to analyze the stability of a non-linear system like non-holonomic system controlled by using either fuzzy logic or neural network. Recently, there are some studies on the stability analysis of fuzzy control system [47–52] and neural control system [53, 54]. Most of them are based on the Lyapunov's direct method. However, none of the above work analyzed the stability of the system, on which the control algorithm was applied. Rather, they had considered the sensitivity of the controller irrespective of the system equation(s). A car-like robot is a non-holonomic system and its motion may be restricted due to its kinematic and dynamic constraints. Thus, it is interesting to test the stability of the whole navigation system, rather than analyzing the stability (sensitivity) of the controller only.

Our aim is to develop a suitable motion planning scheme for a car-like robot navigating among some static obstacles. A fuzzy logic-based motion planner was developed in our previous work [23], where the entire knowledge base of the fuzzy logic was optimized automatically using a GA. However, in that study, the performance of the motion planner was tested through computer simulations only. An NN-based motion planning scheme had also been proposed by the authors in

their earlier work [34]. However, its performance was also not tested on a real-robot. Marichal et al. [18] and Li et al. [20] developed navigation schemes based on neuro-fuzzy approaches. The main drawback of their methods lies in the fact that the FLCs were optimized using the principle of steepest descent method, which may have the local minima problem. In the present study, an attempt is made to tune the FL-based and NN-based motion planners and verify their effectiveness on a real car-like robot, so as to identify the better one in terms of performance, adaptability and others. Moreover, their performances have been compared with that of the potential field method. The stability of the developed navigation schemes have also been analyzed in the present work, based on the Lyapunov's theory.

The rest of the paper is structured as follows: In Section 2, motion planning problem of a car-like robot is stated along with its mathematical formulation. Developed navigation schemes are discussed in Section 3. Experimental set-up is described and the methods of conducting the experiments are explained in Section 4. Results are presented and discussed in Section 5 and stability of the developed navigation schemes is analyzed in Section 6. Comparisons of others' work have been made with the present work in Section 7. Finally, some concluding remarks are made in Section 8 and the scope for future work is indicated in Section 9.

## 2 Navigation Problem of a Car-Like Robot

A mobile robot will have to find its collision-free path during navigation among some static obstacles. Depending on the position and size of the obstacles, the robot may find a number of collision-free paths. However, our aim is to determine that particular path, which is not only the collision-free but also time-optimal. The performances of the developed motion planners have been tested on a real car-like robot as shown in Fig. 1. Its kinematic and dynamic constraints may impose some restrictions on its motion. Therefore, a particular collision-free path (may be time-optimal also) of the robot may not be possible to achieve until or unless the constraints are satisfied. Figure 2 shows a typical problem scenario, in which a car-like robot is moving among five obstacles, sharing the common workspace. The robot has to find its time-optimal and collision-free path during its movement between a starting position S and a goal position G. Now, to reduce the complexity of the problem, only one obstacle has been treated as the most critical one and the motion of the robot is planned based on that particular obstacle. Moreover, the wheels of the robot are allowed to move due to pure rolling action only and Coriolis component of the force is neglected, in the present study. Thereafter, the total path of the robot is assumed to be a collection of a number of small segments, each of which is traveled during a fixed time $\Delta T$. The robot negotiates its motion during those time steps, in order to avoid collision with the most critical obstacle. The critical obstacle has been identified depending on the relative position of the robot and the obstacles. 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 indicating the distance step, then it might be considered as a critical obstacle. Among all such obstacles, the physically closest one is taken as the most critical obstacle. Thus, although the obstacle $O_4$ (refer to Fig. 2) is the physically closest to the robot,

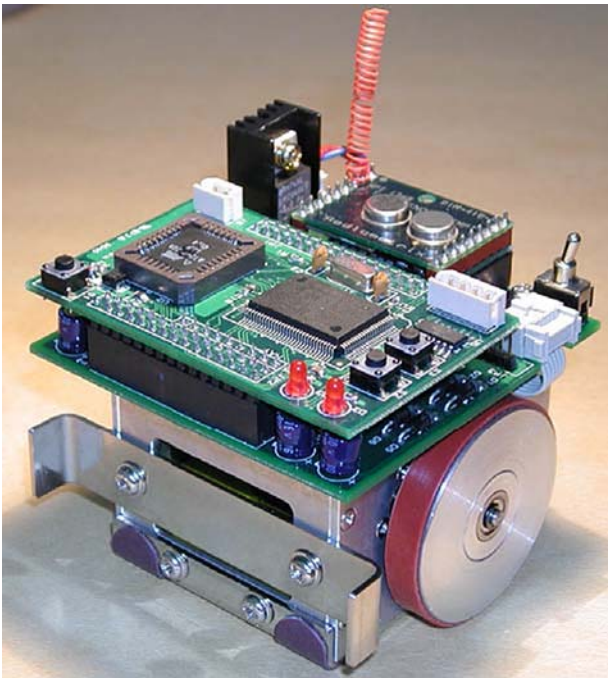

**Fig. 1** Photograph of the robot used in the experiment

it is not being treated as the most critical one. Rather the obstacle $O_3$ is considered to be the most critical one, because it lies within the angle of search. Now, the motion of the robot is planned based on the two inputs – *distance*, *angle* as shown in Fig. 2. The motion planner will determine acceleration and the angle through which it should

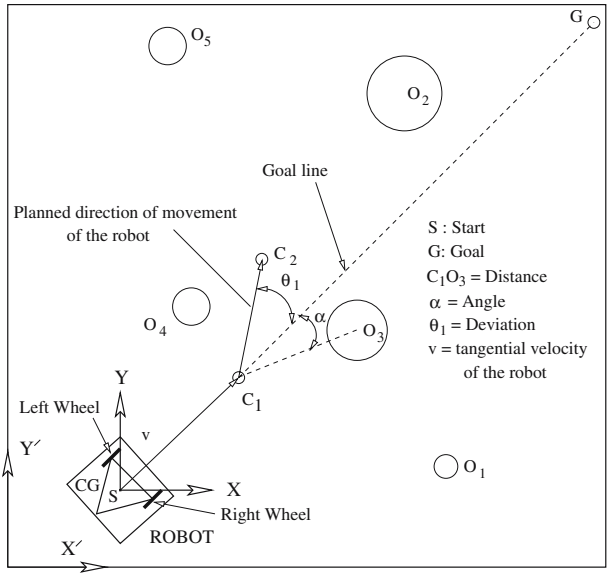


**Fig. 2** Navigation of a car-like robot among static obstacles

deviate with respect to the reference line, so that it can avoid collision with the obstacle.

## 2.1 Mathematical Formulation of the Problem

The developed motion planning scheme of the robot is explained with the help of Fig. 3. The total path (starting from a pre-defined position to a fixed goal) of the robot is assumed to be a collection of some small segments (either a straight one or a combination of straight and curved paths), each of which is traversed during a fixed time $\Delta T$. If the robot finds any critical obstacle ahead of it, the motion planner is activated. Otherwise, the robot moves toward the goal in a straight path with a maximum possible velocity. The task of the motion planner is to determine the *acceleration* ($a$) and *deviation* ($\theta_1$) of the robot based on the *distance* and *angle* inputs, to avoid collision with it. This process will continue, until the robot reaches its destination and total traveling time T is calculated by adding all intermediate time steps taken by the robot to reach it. It is important to mention that the last time step ($T_{\text{rem}}$) may not be a complete one and it depends on the distance left uncovered ($d_{\text{goal}}$) by the robot. If it (i.e., the goal distance $d_{\text{goal}}$) comes out to be less than or equal to a predefined minimum distance ($d_{\text{min}}$), it starts decelerating and stops at the goal. Again, sometimes the robot's motion as provided by the motion planner may violate its kinematic and/or dynamic constraints. In such a situation, the robot is stopped at the present position itself. This is a dead-lock situation, where the robot will not be able to find any feasible solution, as the obstacles are stationary. Our aim is to design a suitable adaptive motion planner, so that the robot will be able to reach

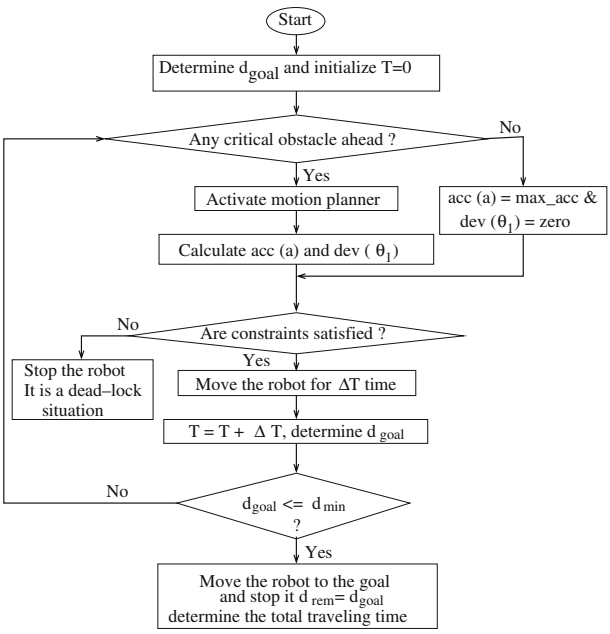


**Fig. 3** Flowchart of the motion-planning scheme

its destination with the lowest possible traveling time by avoiding collision with the obstacles. Therefore, the present problem can be treated as a constrained traveling time ($T$) minimization problem. Let us consider that the robot travels through $U$ number of complete time steps and a fractional time step $T_{\text{rem}}$ to reach its goal after starting from an initial position. Thus, the optimization problem can be stated as follows:

$$\text{Minimize } T = U \times \Delta T + T_{\text{rem}}, \tag{1}$$

subject to

- The path is collision-free,
- The following kinematic and dynamic constraints are satisfied.

$$\begin{aligned}
&(1) \quad -\dot{X}\cos\theta + \dot{Y}\sin\theta = 0,\\
&(2) \quad (\dot{X})^2 + (\dot{Y})^2 - (\rho_{\min}\dot{\phi})^2 \geq 0,\\
&(3) \quad -\sqrt{(\mu_f g)^2 - (v\dot{\phi})^2} \leq a \leq \sqrt{(\mu_f g)^2 - (v\dot{\phi})^2},\\
&(4) \quad a \geq \frac{60P}{2\pi r \times GR \times M \times N_m},
\end{aligned}$$

where $\dot{X}$ and $\dot{Y}$ are the components of tangential velocity along +ve $X$-axis and +ve $Y$-axis, respectively and $\theta$ is the angle between the $X$-axis and the main axis of the robot. The minimum radius of curvature is represented by $\rho_{\min}$ and $\dot{\phi}$ denotes the rate of steering angle during turning. Tangential velocity and acceleration of the CG of the robot are represented by $v$ and $a$, respectively and power required by the motor to create maximum angular speed $N_m$ is expressed by $P$. Moreover, GR represents the gear ratio of the wheels, $r$ is the radius of the wheels of the robot and the mass of the robot is denoted by $M$. Again, $\mu_f$ indicates the coefficient of friction between the wheels and the surface of the terrain and the acceleration due to gravity is represented by $g$.

It is important to mention that both the above objective function as well as constraints are dependent on the variables – *deviation* and *acceleration*, the outputs of the developed motion planner. It is also interesting to note that the traveling time will be minimum, when the robot moves with the maximum possible acceleration, after following the minimum deviation path. The deviation of a path is determined at each step with respect to the corresponding reference line joining the present position of the robot with its goal position.

## 3 Developed Navigation Schemes

Several methods had been tried by various investigators to solve the motion planning problem of a robot in the presence of some static obstacles. The authors have also developed some useful methods, some of which are discussed below, which were found to be effective through computer simulations.

### 3.1 Approach 1: Genetic-fuzzy System

An FLC may provide feasible solutions to the said problem. Two condition variables, such as (1) *distance* of the robot from the most critical obstacle and (2) *angle* between

the line joining the robot and the most critical obstacle and the reference line (joining the robot and its goal) are fed as inputs to the controller. The outputs of the controller are considered to be *deviation* and *acceleration* required by the robot to avoid collision with the most critical obstacle. In the present study, the range of *distance* is divided into four linguistic terms: very near (VN), near (NR), far (FR), very far (VF). Five linguistic terms have been considered for both the *angle* as well as *deviation*: left (LT), ahead left (AL), ahead (AH), ahead right (AR) and right (RT) and *acceleration* is considered to have four terms: very low (VL), low (L), high (H), very high (VH). Therefore, there will be a maximum of twenty input combinations, and for each input combination, there is a maximum of twenty output combinations. Thus, there is a maximum of 400 (i.e., $20 \times 20$) rules present in the rule base and a particular rule will look like the following.

IF *distance* is VF AND *angle* is LT, THEN *deviation* is AH and *acceleration* is VH.

For ease of implementations, membership function distributions of both the input as well as output variables are assumed to be symmetric triangles (refer to Fig. 4). Thus, the data base of the FLC may be represented by providing the four continuous variables representing the half base-widths of the triangular membership function distributions. The performance of an FLC depends on its both data base as well as rule base, which are to be optimized simultaneously. Different methods are available for the development of a suitable KB of an FLC and automatic design procedure using a GA is found to provide the best result for solving the navigation problems of a car-like robot, in simulation [23]. Thus, in the present work, an attempt is made to develop a good KB of an FLC automatically by using a binary-coded GA. A GA-string consisting of 440-bits is considered to represent the KB of the FLC as shown below.

$$\underbrace{10\cdots1\ \ 01\cdots1\ \ 10\cdots0\ \ 01\cdots0}_{\text{Data base}} \qquad \underbrace{10\cdots01}_{\text{Input combinations}} \qquad \underbrace{10101\ldots0101\ldots11001}_{\text{Consequent of the rules}}$$

The first 40-bits in this string represent the half base-widths of the four triangles (10 bits for each variable) and the next 20-bits are used to indicate the presence

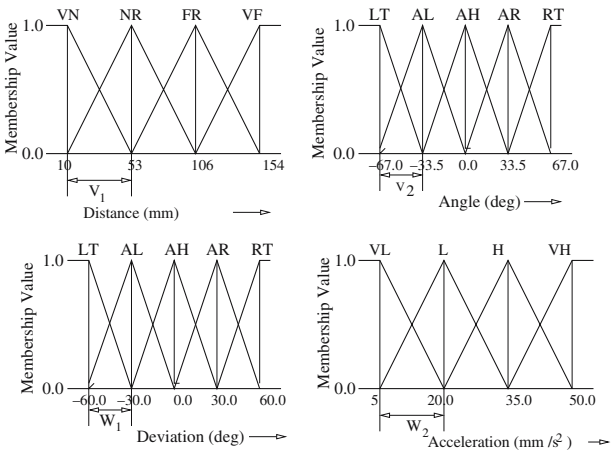


**Fig. 4** Membership function distributions for input and output variables of the FLC

or absence of the input combinations in the rule base (1 for presence and 0 for absence). Out of the remaining 380-bits of the string, every 19-bits will carry the information regarding the combination of the consequents, for a particular input condition. We count the number of 1*s* present in each 19-bits long sub-string. If it comes out to be zero, it will represent the first output combination, i.e., deviation is LT and acceleration is VL, and so on. Figure 5 shows the working principle of the combined GA-FLC/NN approach. The GA begins its search by randomly creating a number of solutions (equals to the population size) represented by the binary strings and each string indicates a typical fuzzy/neural network-based motion planner. Each solution in the population is then evaluated, to assign a fitness value. The fitness of a GA-string is calculated by using the equation given below.

$$\text{Fitness} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{U}\sum_{s=1}^{U}\sum_{v=1}^{2}(T_{\text{nsv}} - O_{\text{nsv}}) + \text{Penalty}, \tag{2}$$

where U denotes the total number of time steps in a planned path and the total number of training scenarios is indicated by N. $O_{\text{nsv}}$ and $T_{\text{nsv}}$ are representing the values of actual output and target output, respectively, of an output variable (say,$v$).

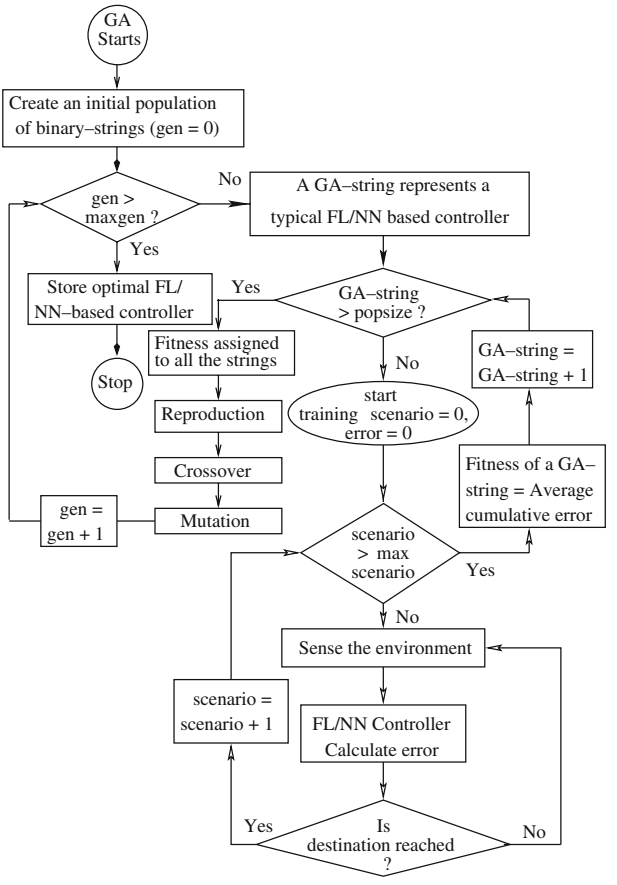


**Fig. 5** A schematic diagram showing the working principle of the genetic-fuzzy/neural system

The target output for deviation is considered to be equal to zero and that for acceleration is taken as the maximum permissible acceleration of the robot. A fixed penalty equals to 2,000 is given to the string, if the FLC represented by it, is unable to provide any solution particularly in case of non-firing situation or the generated motion of the robot fails to satisfy the dynamic and/or kinematic constraints.

After the fitness is assigned to each solution in the population, they are modified by using three operators – reproduction, crossover and bit-wise mutation. One iteration involving these three operators followed by the fitness evaluation is called a generation. Generations proceed until a termination criterion is satisfied. In this approach, the GA is allowed to run for a pre-specified number of generations.

During optimization, an optimal rule base of the FLC is determined by considering the importance of each rule [55], which is calculated as $I_{ij} = p_{ij}\,\overline{\overline{C}}_j$, where $p_{ij}$ denotes the probability of occurrence of $j$th output combination corresponding to $i$th input condition of the rule, where $i, j = 1, 2, \cdots, 20$ and $\overline{\overline{C}}_j = \frac{1}{2}\,(\overline{C}_q + \overline{C}_r)$, where $\overline{C}_q$ and $\overline{C}_r$ are the average worth of $q$th linguistic term of the first output (i.e., deviation) and $r$th term of acceleration output, respectively. It is important to note that the worth, corresponding to a linguistic term of an output, is determined by following the Gaussian distribution pattern, maximum being occurred for deviation output AH and acceleration output VH. It is also to be noted that during optimization, half-base width of four triangular membership function distributions are varied in the ranges of (40, 60), (20, 40), (20, 40) and (5, 15), respectively.

### 3.2 Approach 2: Genetic-neural (GA-NN) System

Neural network has the capability of solving different complex real-world problems and it may also provide a feasible solution to the present problem. However, the performance of an NN-controller depends on its topology, connecting (synaptic) weights and biases. Quite a few researchers tried to develop a suitable NN controller using a back-propagation algorithm. But, optimal design procedure of an NN architecture using a GA, resulted better solution for solving the similar problem, in simulation [34]. Thus, in the present study, simultaneous optimization of weights and the architecture of a neural network using a GA is followed. A three-layered fully-connected neural network architecture is considered in the present study. The first layer contains two neurons representing the two different inputs (i.e., *distance* and *angle*) of the controller. In the hidden layer, a maximum of twenty neurons have been assumed and the optimal number of neurons present in this layer is varied between 2 to 20, during optimization. In the output layer, two neurons have been considered, which represent the two different outputs (i.e., *deviation* and *acceleration*) of the motion planner. It is to be noted that the activation functions at each layer are assumed to be tan-hyperbolic in nature and bias values of all the neurons are kept constant through out the study. Now, to select proper magnitudes of the constant of activation functions ($C_1, C_2, C_3$) and to optimize the weights of the network, a binary-coded GA with 850-bits long string has been utilized. The first 30 bits will carry information of three continuous variables – $C_1, C_2, C_3$ (10 bits for each variable), representing the constants of hyperbolic functions at three different layers. Out of the remaining 820 bits, every 41 bits (starting from 31$^{st}$ bit location of 850-bits long string) are used to indicate the existence of a hidden neuron (1 for presence and 0 for absence) and its corresponding four synaptic weights (10 bits for each weight).

Therefore, a GA-string will look as follows (in which 41-bits are shown to indicate the presence of $j$th neuron and its connecting weights, such as $v_{1j}$, $v_{2j}$, $w_{j1}$, $w_{j2}$):

$$\underbrace{1\cdots1}_{C_1}\ \underbrace{0\cdots1}_{C_2}\ \underbrace{1\cdots0}_{C_3}\ldots\ldots\ \underbrace{\underbrace{1}_{j\text{th hidden neuron}}\ \underbrace{1\cdots1}_{v_{1j}}\ \underbrace{0\cdots1}_{v_{2j}}\ \underbrace{1\cdots0}_{w_{j1}}\ \underbrace{0\cdots0}_{w_{j2}}\ldots}_{\text{Architecture of NN}}$$

It is important to mention that we have restricted our search up to a maximum of twenty neurons lying in the hidden layer. During optimization, the constants of activation function for three layers are varied in a range of (0.1 to 15.0) and the weights are allowed to vary from 0.0 to 1.0. The ranges of variation of different variables are selected after a careful study. The working principle of the combined GA-NN approach is almost similar to the combined GA-fuzzy approach (refer to Fig. 5). The fitness of the GA-string has been calculated in the same way, as it has been done in case of the GA-fuzzy approach (refer to Eq. 2).

### 3.3 Approach 3: Potential Field Method

Potential field method, introduced by Khatib [9], is widely used for real time collision-free path planning of both manipulators as well as mobile robots. In this approach, the robot is modeled as a particle moving under the influence of an artificial potential field, which is determined by the set of obstacles and the target destination. The target is assumed to have an attractive potential and the obstacles generate the repulsive potentials. The movement of the robot is then achieved by determining the resultant of the above attractive and repulsive forces. However, 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 function, quadratic, exponential function, had been tried by various investigators [1, 10], out of which, parabolic and hyperbolic functions had been widely used for solving the similar problem [56], due to their nonlinear approximation capability about the system. The attractive potential field $U_{\text{att}}(X)$ can be defined as a parabolic-well as follows.The attractive $U_{\text{att}}(X)$ and repulsive $U_{\text{rep}}(X)$ potential fields, used in this study, can be expressed as follows.

$$U_{\text{att}}(X) = \frac{1}{2}\xi_{\text{att}}\, d^2_{\text{goal}}(X), \tag{3}$$

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

$$U_{\text{rep}}(X) = \frac{1}{2}\xi_{\text{rep}}\left[\frac{1}{d_{\text{obs}}(X)} - \frac{1}{d_{\text{obs}}(0)}\right]^2, \tag{4}$$

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

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

$$F_{\text{att}}(X) = \xi_{\text{att}}\, d_{\text{goal}}(X) \tag{5}$$

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

$$F_{\text{rep}}(X) = -\xi_{\text{rep}}\, \frac{1}{d^2_{\text{obs}}(X)} \left[ \frac{1}{d_{\text{obs}}(X)} - \frac{1}{d_{\text{obs}}(0)} \right] \tag{6}$$

The resultant potential force $F(X)$ is then calculated by adding $F_{\text{att}}(X)$ with $F_{\text{rep}}(X)$ vectorially. In this approach, the acceleration output is taken to be proportional to the magnitude of the resultant force $F(X)$ and 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 at the present time step and the goal position.

## 4 Description of the Experimental Set-up

A full package related to a soccer playing robot system has been purchased from Microrobot NA, South Korea, for the experimental verification of the developed motion planning schemes. The package consists of a robot, a CCD camera along with a frame grabber board and a radio-frequency module to ensure wireless communication between the robot and the host PC. Figure 6 shows the photograph of the experimental set-up. The robot and the obstacles are allowed to move over the field. A CCD camera mounted on a tripod is used for sensing the environment. The camera sends continuous electrical signals to the computer through a BNC video cable. Thereafter, a frame grabber board, namely vision board is used to convert the continuous signal to 2-D digital images. Once the images are captured and stored into the computer frame memory, these are analyzed to obtain the position and size of both the obstacles as well as the robot. Developed motion planning schemes are then utilized to find the safe path of the robot. Thereafter, controlled information is communicated to the robot by means of a radio-frequency module, which transmits data through a communication protocol.

### 4.1 Methods of Conducting the Experiments

The experiments are carried out by following the steps mentioned below.

1. *Camera Calibration:* A CCD camera is used for collecting information of the dynamic environment. However, the performance of the camera depends on some of its internal/geometric and external parameters. An optimal set of those parameters has been determined by using a binary-coded genetic algorithm, in the present study.
2. *On-line Image Processing:* The images captured with the help of the camera and its accessories are analyzed by developing a fast and noise insensitive image processing method. It includes the following modules:
   - Noise removal using median filtering,
   - Binarization of the images by means of a threshold value,


**Fig. 6** The photograph of the experimental set-up

- Estimation of the perimeters, area of the objects by using the perimeter descriptor,
- Labeling of the objects,
- Removal of extraneous components by using a size filter.

3. *Control of the robot:* The inputs of the motion planner are obtained by analyzing the images captured using the camera. The motion planner's outputs are then utilized to determine the angular speed of two wheels of the robot and these are implemented by following PD control law. Thereafter, speed information of the wheels are communicated to the robot by means of a radio-frequency (RF) module. Thus, it is possible to achieve a wireless communication between the robot and the host computer. Finally, actuation of the robot takes place with the help of two separately controlled differential drive DC motors.

## 5 Results and Discussion

Attempts have been made to solve the navigation problems of a real car-like robot moving among some static obstacles, in the present work. Three different motion planners have been developed as explained earlier for the said purpose. Since the

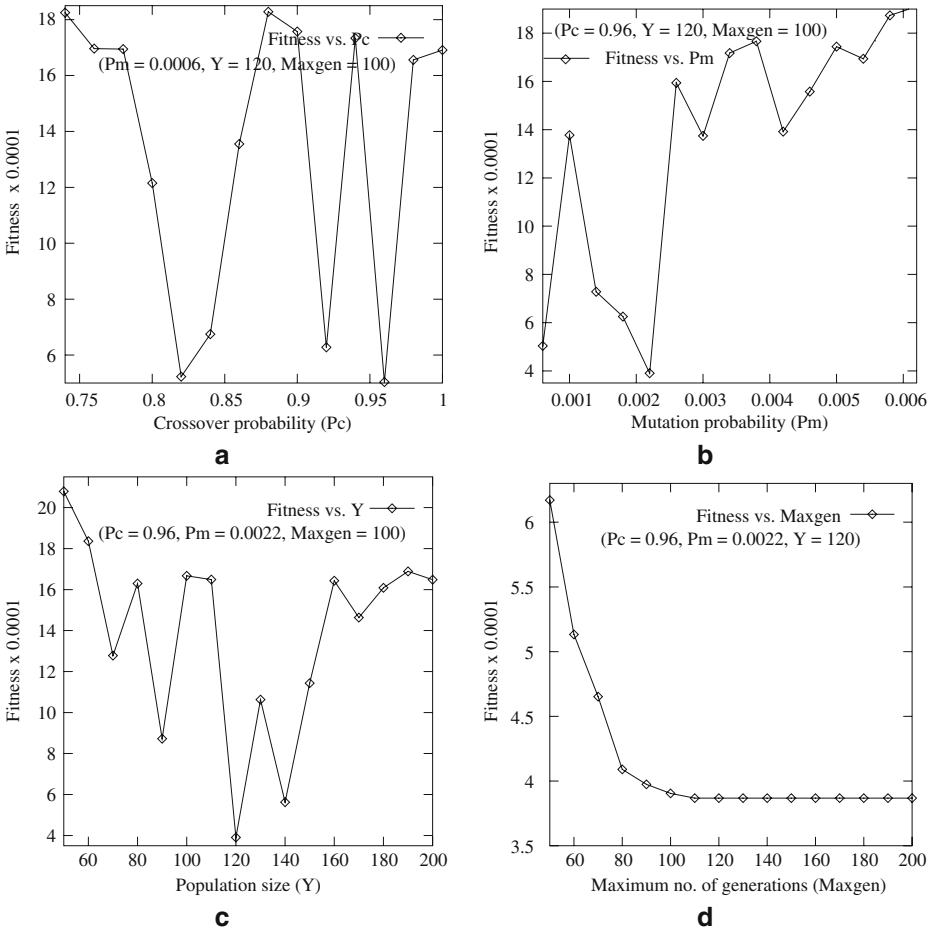


**Fig. 7** Results of the parametric study to obtain the optimal GA-parameters for determining a good KB of an FLC: **a** fitness vs. crossover probability, **b** fitness vs. mutation probability, **c** fitness vs. population size, **d** fitness vs. maximum no. of generations

design of both the fuzzy logic-based as well as neural network-based motion planners is a tedious job and requires training and/or learning to improve their performances, an off-line training procedure has been adopted by using a GA for the development of a suitable adaptive motion planner. A set of two hundred training scenarios consisting of the information of the starting position of the robot, positions of the obstacles and size of all the objects have been considered for this purpose. The time interval ($\Delta T$) is taken to be equal to 0.033 s. The robot is assumed to have a maximum and minimum acceleration of 50 and 5 mm/s$^2$, respectively. It is to be noted that during training the obstacles are assumed to be circular and the radii of their boundaries are varied between 10 to 30 mm, the maximum and minimum velocities of the robot are considered to be equal to 200 and 2 mm/s, respectively. Collision-free movements of the robot are presented for two different cases. In Case 1, the robot is allowed to navigate among two static obstacles, whereas more complex environment

**Table 1** Importance factor of each rule of the optimized RB – Case 1

| Rule no. | Dist. | Ang. | Dev. | Accn. | Imp. factor |
|---|---|---|---|---|---|
| 1 | VN | LT | LT | VH | 0.0000408 |
| 2 | VN | AL | AL | VH | 0.0013100 |
| 3 | VN | AH | AH | VH | 0.0004950 |
| 4 | NR | AL | AH | H | 0.0020800 |
| 5 | NR | AH | AR | VH | 0.0052500 |
| 6 | NR | RT | AH | VH | 0.0024400 |
| 7 | FR | LT | AH | VH | 0.0005380 |
| 8 | FR | AL | AH | VH | 0.0009450 |
| 9 | FR | AH | RT | VH | 0.0002980 |
| 10 | FR | RT | AH | H | 0.0046500 |
| 11 | VF | AL | AL | VH | 0.0000023 |
| 12 | VF | AR | AH | H | 0.0004480 |
| 13 | VF | RT | AH | VL | 0.0000119 |

having three obstacles are considered in Case 2. The performances of the FL-based and NN-based motion planners are compared among themselves and to that of a conventional potential field method for solving the same problem.

### 5.1 Case-1: Navigation Among Two Obstacles

The navigation problem of the robot among two stationary objects is studied in a grid of $1.3 \times 1.1$ m. Since the performance of a GA depends on its parameter setting, experiments are carried out with different sets of parameters, to find the most suitable one. Results of the parametric study are shown in Fig. 7. The best results are obtained with the following GA-parameters: crossover probability $p_c = 0.96$, mutation probability $p_m = 0.0022$, population size $Y = 120$, maximum number of generation Maxgen $= 105$. After the training of FL-based motion planner is over, the GA has selected thirteen good rules from a total of twenty, as shown in Table 1 and the optimized membership function distribution is shown in Fig. 8. It is interesting to

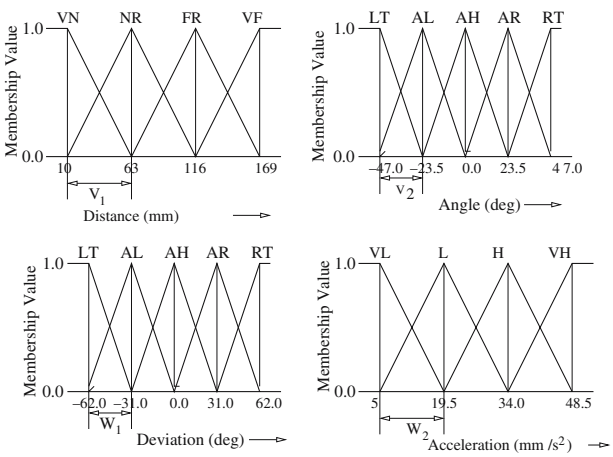


**Fig. 8** Optimized membership function distributions of the FLC – Case 1

note that in the rules involving VN distance (the first three rules of Table 1), linguistic terms for the input – *angle* and the output – *deviation* are coming to be the same. It may be due to the fact that the numerical values of the half base widths of the GA-optimized membership function distributions for *angle* input and *deviation* output are not the same. Moreover, the second output – *acceleration* has appeared to be VH. Thus, if the robot finds any obstacle in its close vicinity, it will try to come out of this situation with a very high velocity.

Due to the iterative nature of the GA, some redundant rules might be developed in the rule base. To identify them, an *importance factor* [55] of each rule appeared in the optimized rule base, is determined by multiplying its probability of occurrence with the worth. Table 1 shows the *importance factor* of each rule present in the optimized rule base. An experiment is also carried out to check whether the GA-designed rule base contain any redundant rule. When the FLC is run with the eleven optimal rules after removing the rule numbers 11 and 13 of Table 1 (based on the lowest importance factor), no incidence of non-firing is reported. But, whenever one more rule is removed from the rule base, non-firing situations are found to occur for 2,645 times for the training scenarios. It is also observed (as expected) that the number of non-firing cases increases with the reduction of number of rules present in the rule base (refer to Table 2). From this experiment, it has been concluded that the further reduction of the size of the GA-optimized rule base is possible and optimal rule base of the FLC will contain only eleven rules. In Approach 2 also, the parameters of GA are varied in their respective suitable ranges and the following GA-parameters have yielded the best result during training: uniform crossover with probability $p_c = 0.5$, $p_m = 0.00124$, $Y = 190$, Maxgen $= 80$. The GA-optimized NN is seen to contain only six neurons in the hidden layer and the constants of activation function for three layers are found to be equal to 14.836, 14.617 and 4.339, respectively. It is interesting to note that the synaptic weights between the hidden neurons and the neuron corresponding to the deviation output of the optimized network are found to lie on the lower side of their individual ranges. On the other hand, those between the hidden neurons and the neuron related to the acceleration output are seen to have the higher values. It may be due to the fact that the target value of the deviation output is assumed to be equal to zero and that for the acceleration output is considered to be its maximum value, in the present study. Moreover, the effect of distance input on the output values is found to be more than that of the angle input. It may happen due to the fact that the obstacles are stationary, and as a result of which, as the distance input decreases and the angle input will increase, when the robot approaches towards the obstacle.

After the GA-based off-line training is over, the effectiveness of the soft computing-based approaches are compared among themselves and with that of the potential field approach, for five random test scenarios (refer to Table 3), which are not included among the training scenarios. Table 4 compares the performances of three approaches for five test scenarios (refer to Table 3) in terms of deviation and acceleration errors. Approach 3 is found to be the best in terms of deviation error, whereas it has been defeated by other approaches in terms of acceleration error. The average error has been determined of the said two errors considering all the scenarios. Approaches 1 and 3 are found to be the best and worst, respectively, in

**Table 2** Number of rules present in RB vs. number of non-firing incidences

| No. of rules present in RB | Rule no. made absent | No. of non-firing incidences | Traveling time (s) |
|---|---|---|---|
| 13 | – | 0 | 13.8519 |
| 12 | 11 | 0 | 13.8717 |
| 11 | 11, 13 | 0 | 13.9404 |
| 10 | 11, 13, 1 | 2,645 | 18.5963 |
| 09 | 11, 13, 1, 9 | 2,795 | 19.1397 |
| 08 | 11, 13, 1, 9, 12 | 4,195 | 19.8977 |
| 07 | 11, 13, 1, 9, 12, 3 | 4,214 | 19.9797 |
| 06 | 11, 13, 1, 9, 12, 3, 7 | 5,641 | 20.6383 |

**Table 3** Five different test scenarios – 2-obstacles case

| Scenario number | Starting point of the robot (mm, mm) | Goal point of the robot (mm, mm) | Posn. of first obs. (mm, mm) | Posn. of second obs. (mm, mm) |
|---|---|---|---|---|
| 1 | (284, 128) | (1,300, 1,100) | (641, 115) | (546, 595) |
| 2 | (286, 124) | (1,300, 1,100) | (379, 820) | (613, 719) |
| 3 | (112, 201) | (1,300, 1,100) | (499, 320) | (903, 211) |
| 4 | (420, 160) | (1,300, 1,100) | (613, 595) | (304, 1,008) |
| 5 | (282, 224) | (1,300, 1,100) | (824, 861) | (455, 692) |

**Table 4** Comparison of three approaches in terms of accuracies – 2-obstacles case

| Scenario number | Deviation error | | | Acceleration error | | |
|---|---|---|---|---|---|---|
| | Approach 1 | Approach 2 | Approach 3 | Approach 1 | Approach 2 | Approach 3 |
| 1 | 0.168017 | 0.203950 | 0.184260 | 0.243792 | 0.139887 | 0.225970 |
| 2 | 0.134834 | 0.010620 | 0.143550 | 0.043509 | 0.016944 | 0.452600 |
| 3 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 4 | 0.098037 | 0.333708 | 0.000000 | 0.217256 | 0.379009 | 0.580060 |
| 5 | 0.040557 | 0.213684 | 0.000000 | 0.269234 | 0.291645 | 0.399390 |
| Avg. Error | Approach 1 0.121524 | Approach 2 0.158945 | Approach 3 0.198583 | | | |

**Table 5** Comparison of three approaches in terms of traveling time (seconds) – 2-obstacles case

| Scenario number | FLC (s) | NN (s) | PFM (s) |
|---|---|---|---|
| 1 | 21.1 | 21.7 | 23.7 |
| 2 | 16.1 | 14.5 | 18.4 |
| 3 | 13.2 | 13.2 | 13.2 |
| 4 | 15.5 | 16.1 | 16.8 |
| 5 | 13.5 | 13.2 | 14.1 |

terms of average error. Table 5 shows the traveling time taken by the robot following three approaches. The performance of Approach 1 is found to be better than that of Approach 2 in two test scenarios. Moreover, in four out of five scenarios, Approaches 1 and 2 have performed better than Approach 3. It may be due to the fact that there is no in-built optimization module in Approach 3. The values of traveling time taken by the robot after following all the approaches have come out to be the same in case of third scenario (refer to Table 5). It may happen due to the fact that the robot has never faced any critical obstacle ahead of it, during its motion. For a particular test scenario (i.e., 1st test scenario of Table 3), positions of the robot and the obstacles at eight different instants of time are shown in Figs. 9, 10 and 11 for Approaches 1, 2 and 3, respectively. Moreover, Fig. 12 shows the paths planned by the robot using all three approaches for the first test scenario of Table 3. The robot has taken left turn to avoid the second obstacle by following Approaches 1 and 3, whereas it takes right turn in Approach 2. A special situation as shown in Fig. 13 may also occur, where both the obstacles are residing very close to each other and the absolute values of their included angle with respect to the robot are coming out to be very low. In such cases, none of the motion planners is able to provide the feasible solution. It is solved by giving a geometric correction to the motion planner's deviation output. Out of these two obstacles, the one which is nearer to the robot is treated as the most critical one. Now, the movement of the robot is planned on the side opposite to that of the second obstacle with respect to the robot. In the present case, the second obstacle is residing on the left side of the critical obstacle. So, the movement of the robot is planned towards the right side of the critical obstacle (refer to Fig. 14).

### 5.2 Case-2: Navigation Among Three Obstacles

Collision-free navigation of a car-like robot in the presence of three stationary obstacles have been considered in the present case. The following GA-parameters have provided the best result in Approach 1: $p_c = 0.8$, $p_m = 0.0058$, $Y = 180$, Maxgen = 190. During optimization, the GA has selected eight good rules from a total of 20 possible rules through search. It is also interesting to note that only one rule (i.e., rule no. 20) has been identified as the redundant, out of eight good rules found by the GA. Redundancy of a rule is checked by using the concept of importance factor, as discussed earlier. The optimized membership function distributions are shown in Fig. 15 and the optimized rule base of the FLC consisting of seven rules is shown in Table 6 along with the importance factor of each rule. During the evolution of a suitable NN-based motion planner using a GA, the best result is obtained with the following GA parameters: $p_c = 0.5$, $p_m = 0.00108$, $Y = 200$, Maxgen = 120. The optimal number of neurons lying in the hidden layer comes out to be equal to three only and the constants of activation functions at three layers, for which the best result is obtained, are seen to be equal to 12.441, 10.566 and 5.817, respectively.

Table 7 shows five different test scenarios. Comparisons have been made of the three approaches in terms of their deviation error, acceleration error in Table 8

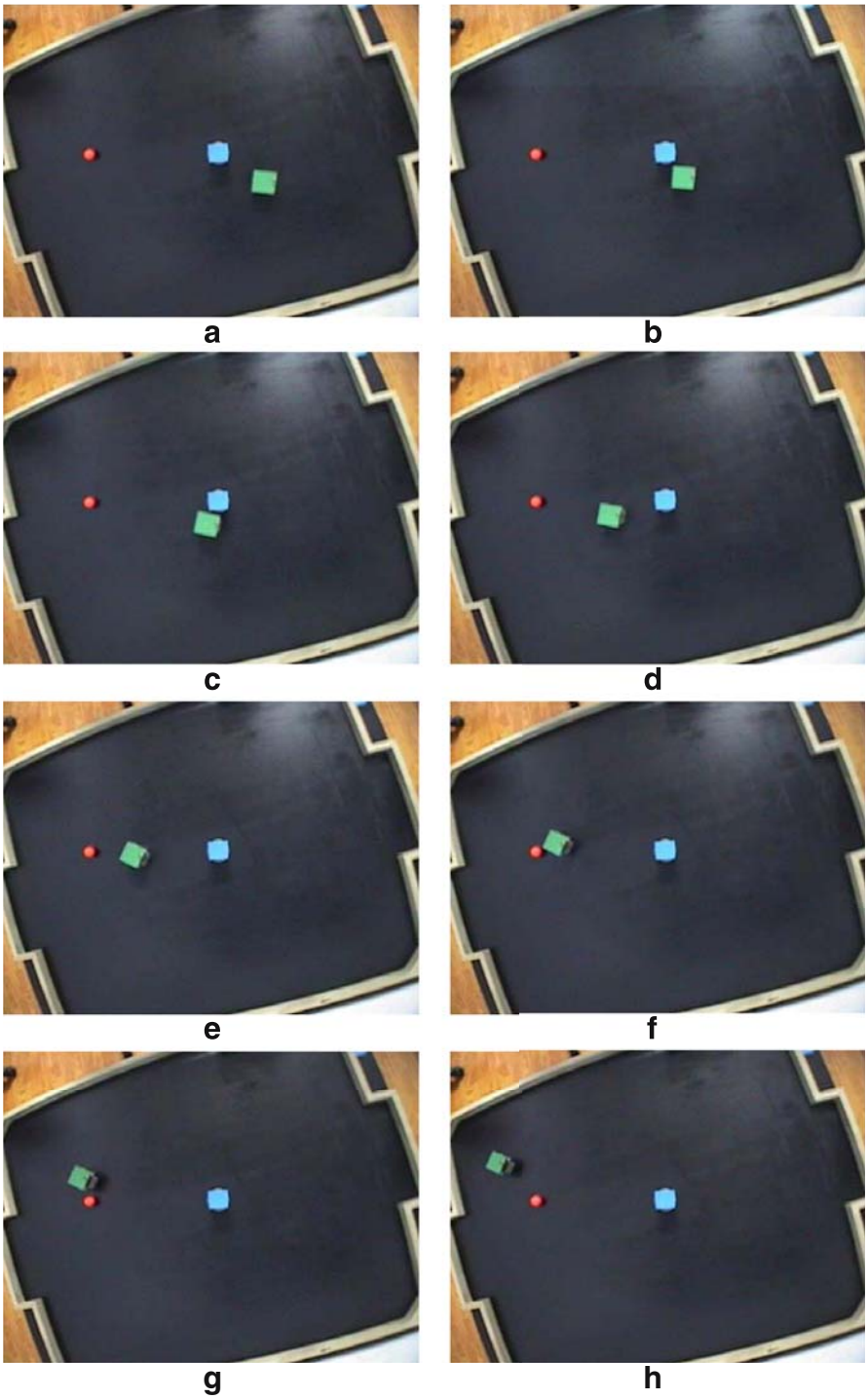


**Fig. 9** Positions of the robot among two static obstacles at eight instants of time – Approach 1

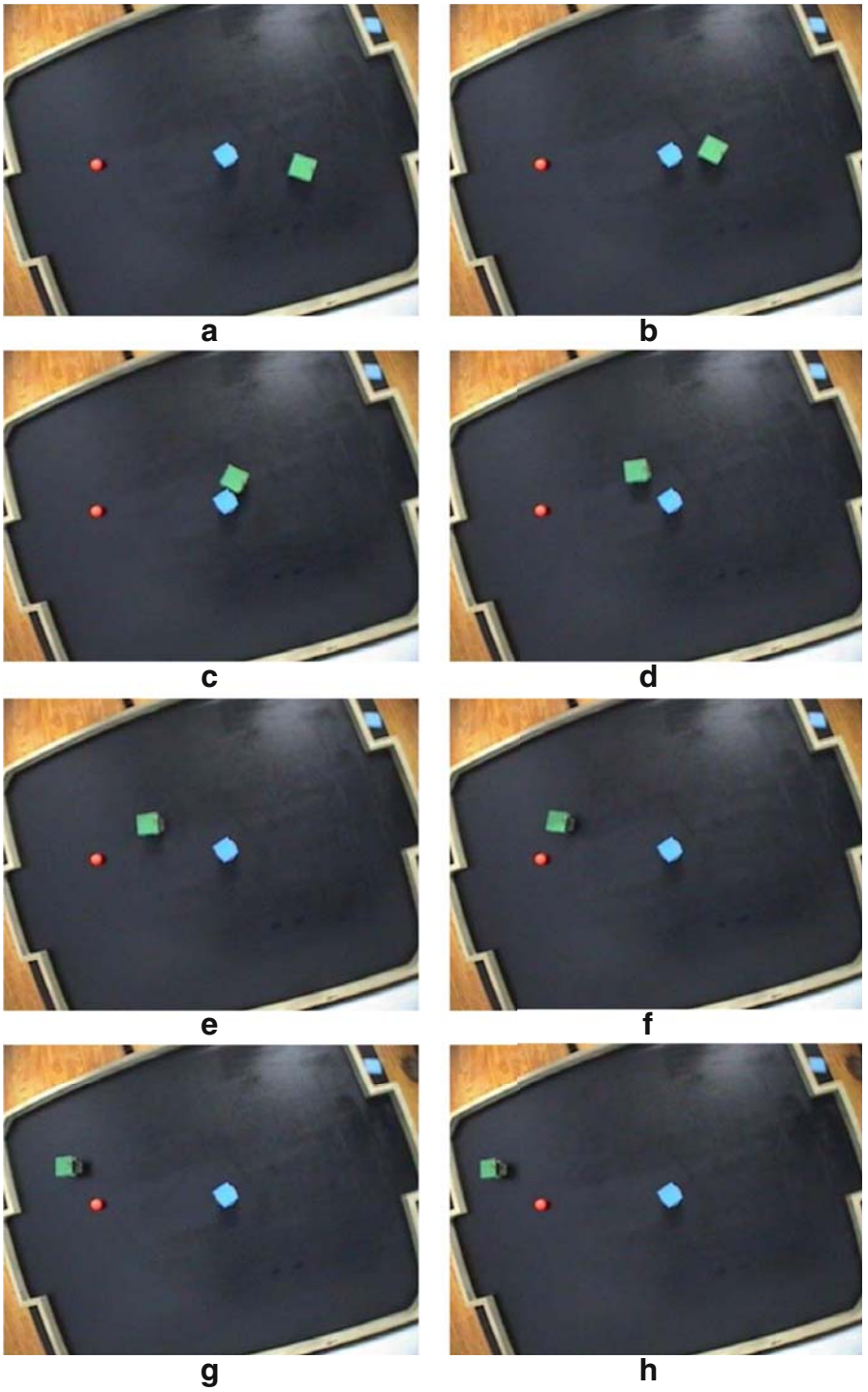


**Fig. 10** Positions of the robot among two static obstacles at eight instants of time – Approach 2

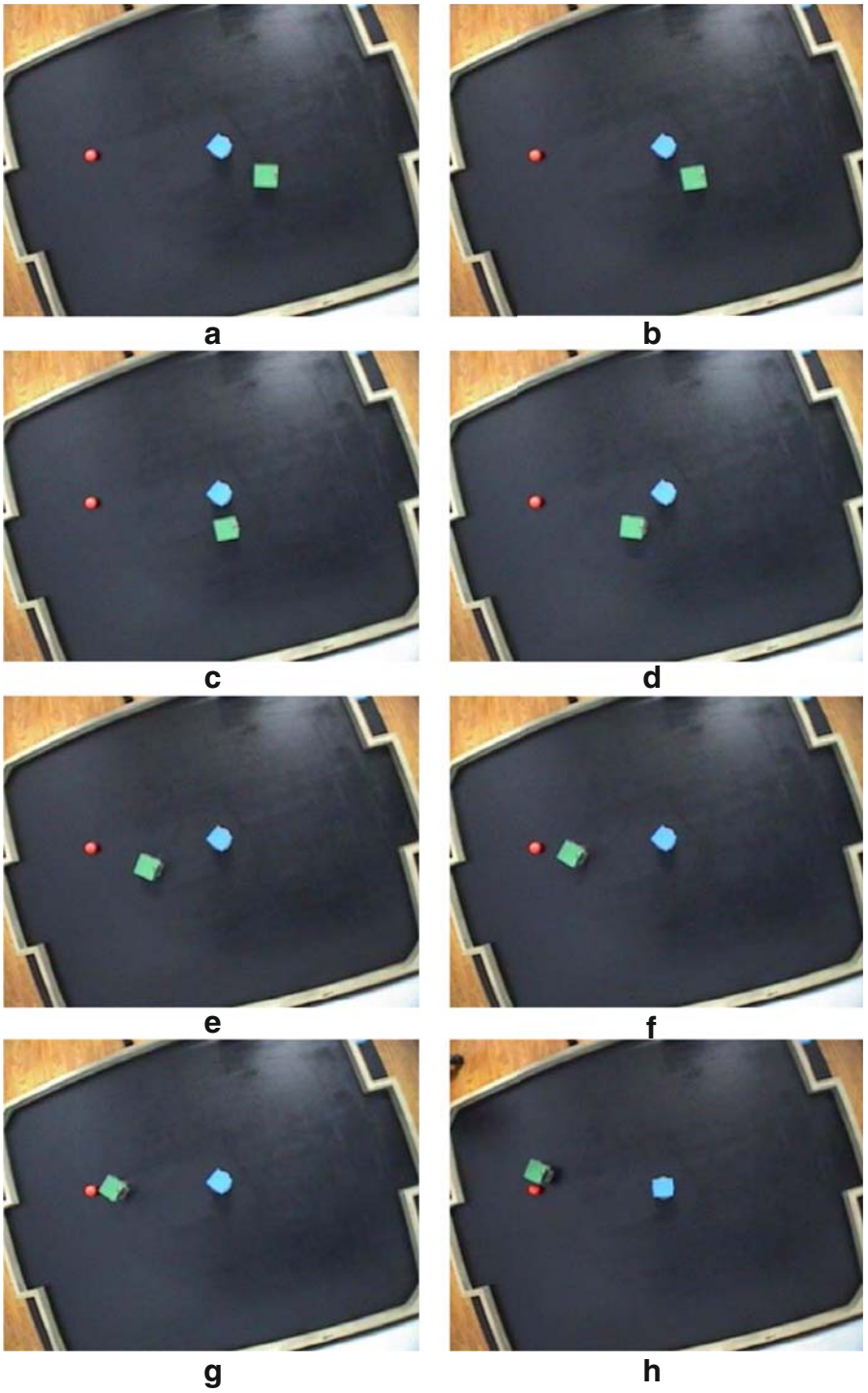


**Fig. 11** Positions of the robot among two static obstacles at eight instants of time – Approach 3

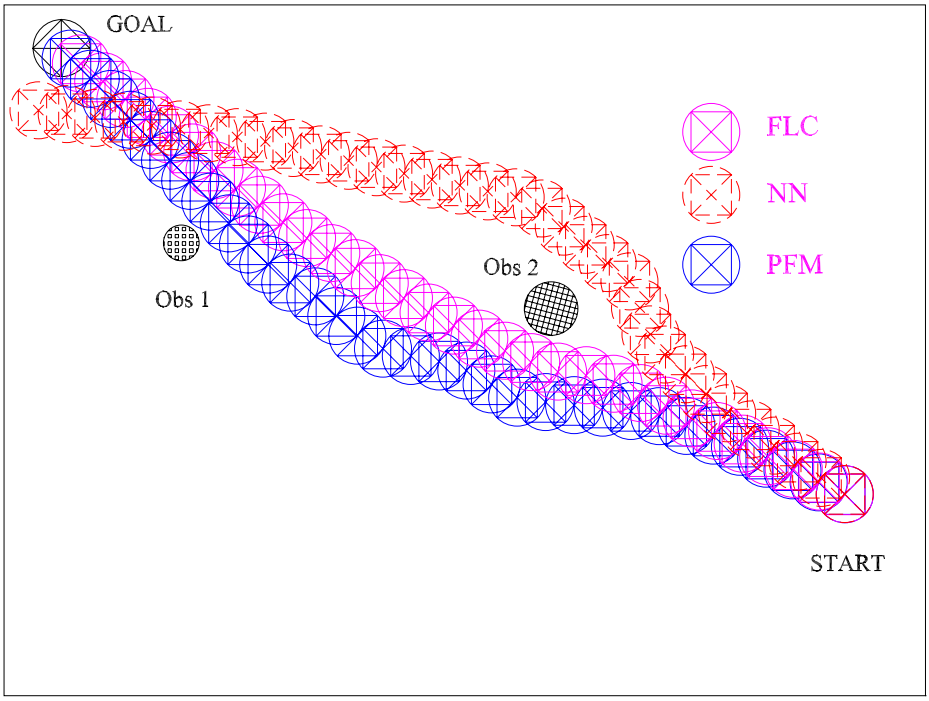


**Fig. 12** Movement of the robot among two static obstacles in Scenario 1 of Table 3

for all the five test scenarios shown in Table 7. The average values of the errors obtained by different approaches have been calculated considering all the scenarios. It is interesting to note that both Approaches 1 and 2 have outperformed Approach 3, in terms of average error. Moreover, the performance of Approaches 1 and 2 are found to be comparable.


**Fig. 13** A typical scenario showing the position of the robot among two closely spaced static obstacles

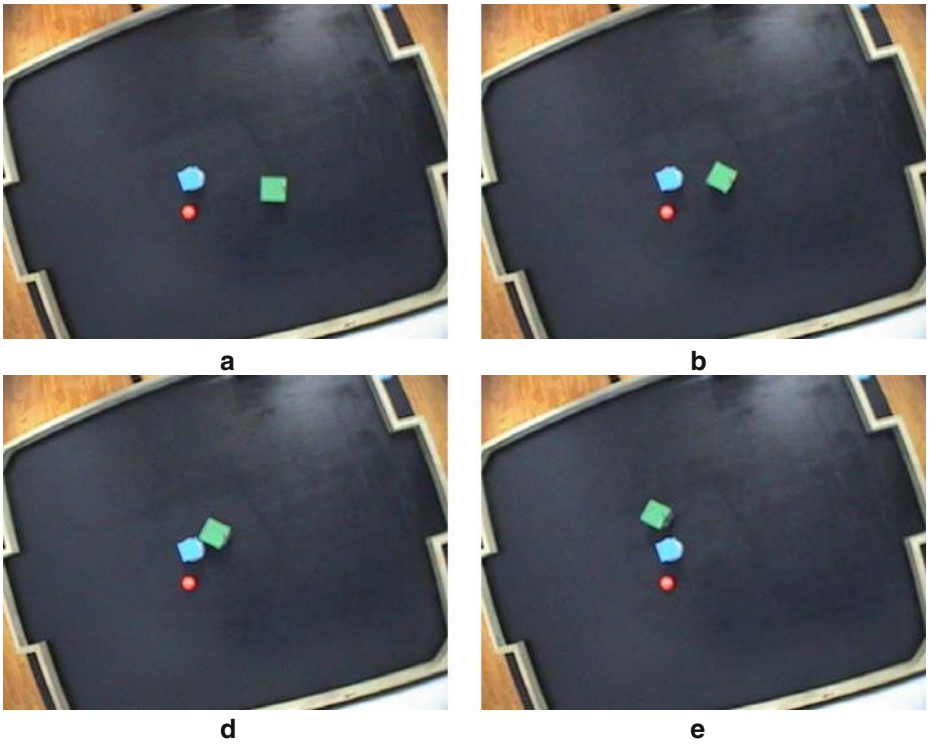


**Fig. 14** A possible solution of the typical scenario shown in Fig. 13

Traveling time values of three different approaches for five test scenarios, are shown in Table 9 and Approach 2 is found to perform better than other approaches, in most of the test scenarios. Moreover, traveling time taken by the robot while following Approach 3 is found to be the maximum. It could be due to the fact that

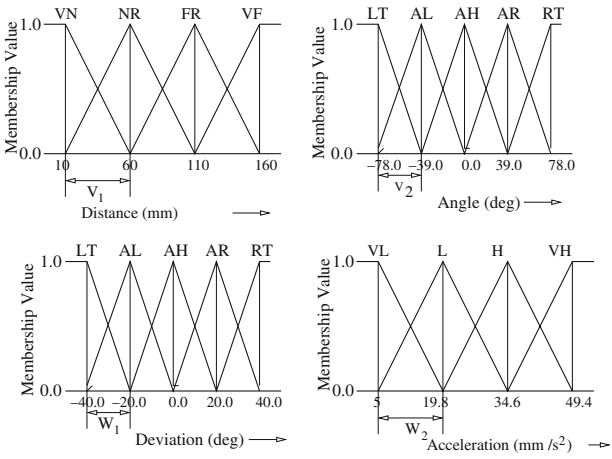


**Fig. 15** Optimized membership function distributions of the FLC for three obstacles case (Case 2)

**Table 6** Importance factor of each rule of the optimized rule base for three obstacles case

| Rule no. | Dist. | Ang. | Dev. | Accn. | Imp. factor |
|---|---|---|---|---|---|
| 1 | VN | AL | LT | VH | 0.000027907 |
| 2 | VN | AH | AH | VH | 0.003069424 |
| 3 | VN | RT | AH | VH | 0.000558174 |
| 4 | NR | RT | AH | VH | 0.007500713 |
| 5 | FR | AL | AH | VH | 0.013002920 |
| 6 | FR | AH | AL | VH | 0.020882550 |
| 7 | VF | AR | AH | VH | 0.001008886 |

**Table 7** Five different test scenarios – 3-obstacles case

| Scenario number | Starting point of the robot (mm, mm) | Goal point of the robot (mm, mm) | Posn. of first obs. (mm, mm) | Posn. of second obs. (mm, mm) | Posn. of third obs. (mm, mm) |
|---|---|---|---|---|---|
| 1 | (282, 279) | (1,300, 1,100) | (247, 870) | (544, 1,182) | (186, 971) |
| 2 | (276, 256) | (1,300, 1,100) | (227, 893) | (455, 742) | (187, 971) |
| 3 | (365, 160) | (1,300, 1,100) | (142, 1,255) | (841, 586) | (239, 623) |
| 4 | (806, 188) | (1,300, 1,100) | (166, 1,072) | (540, 348) | (140, 921) |
| 5 | (331, 481) | (1,300, 1,100) | (426, 206) | (290, 1,063) | (802, 825) |

**Table 8** Comparison of three approaches in terms of accuracies – 3-obstacles case

| Scenario number | Deviation error | | | Acceleration error | | |
|---|---|---|---|---|---|---|
| | Approach 1 | Approach 2 | Approach 3 | Approach 1 | Approach 2 | Approach 3 |
| 1 | 0.076962 | 0.259627 | 0.000000 | 0.116307 | 0.197070 | 0.341300 |
| 2 | 0.062872 | 0.131874 | 0.119649 | 0.120476 | 0.208765 | 0.281140 |
| 3 | 0.010022 | 0.103031 | 0.031064 | 0.121000 | 0.107023 | 0.333333 |
| 4 | 0.001018 | 0.090203 | 0.000000 | 0.102008 | 0.070707 | 0.261917 |
| 5 | 0.143000 | 0.020560 | 0.095155 | 0.471910 | 0.032635 | 0.813076 |
| Avg. Error | Approach 1 0.122558 | Approach 2 0.122150 | Approach 3 0.227663 | | | |

**Table 9** Comparison of three approaches in terms of traveling time (seconds) – 3-obstacles case

| Scenario number | FLC (s) | NN (s) | PFM (s) |
|---|---|---|---|
| 1 | 24.4 | 23.1 | 27.0 |
| 2 | 25.0 | 25.7 | 29.0 |
| 3 | 26.4 | 25.7 | 31.6 |
| 4 | 25.0 | 22.4 | 27.0 |
| 5 | 20.4 | 19.8 | 21.7 |

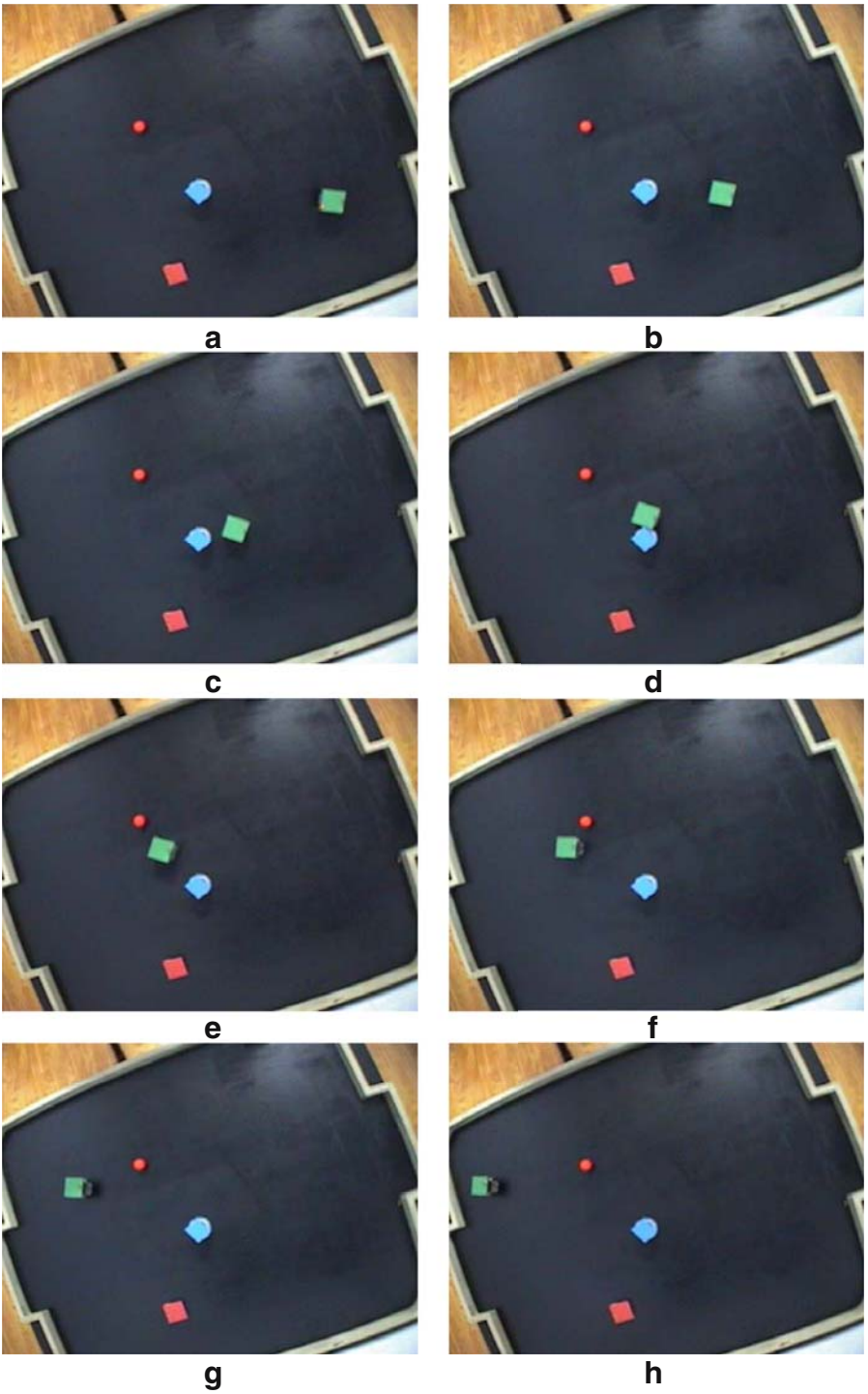


**Fig. 16** Positions of the robot among three static obstacles at eight instants of time – Approach 1

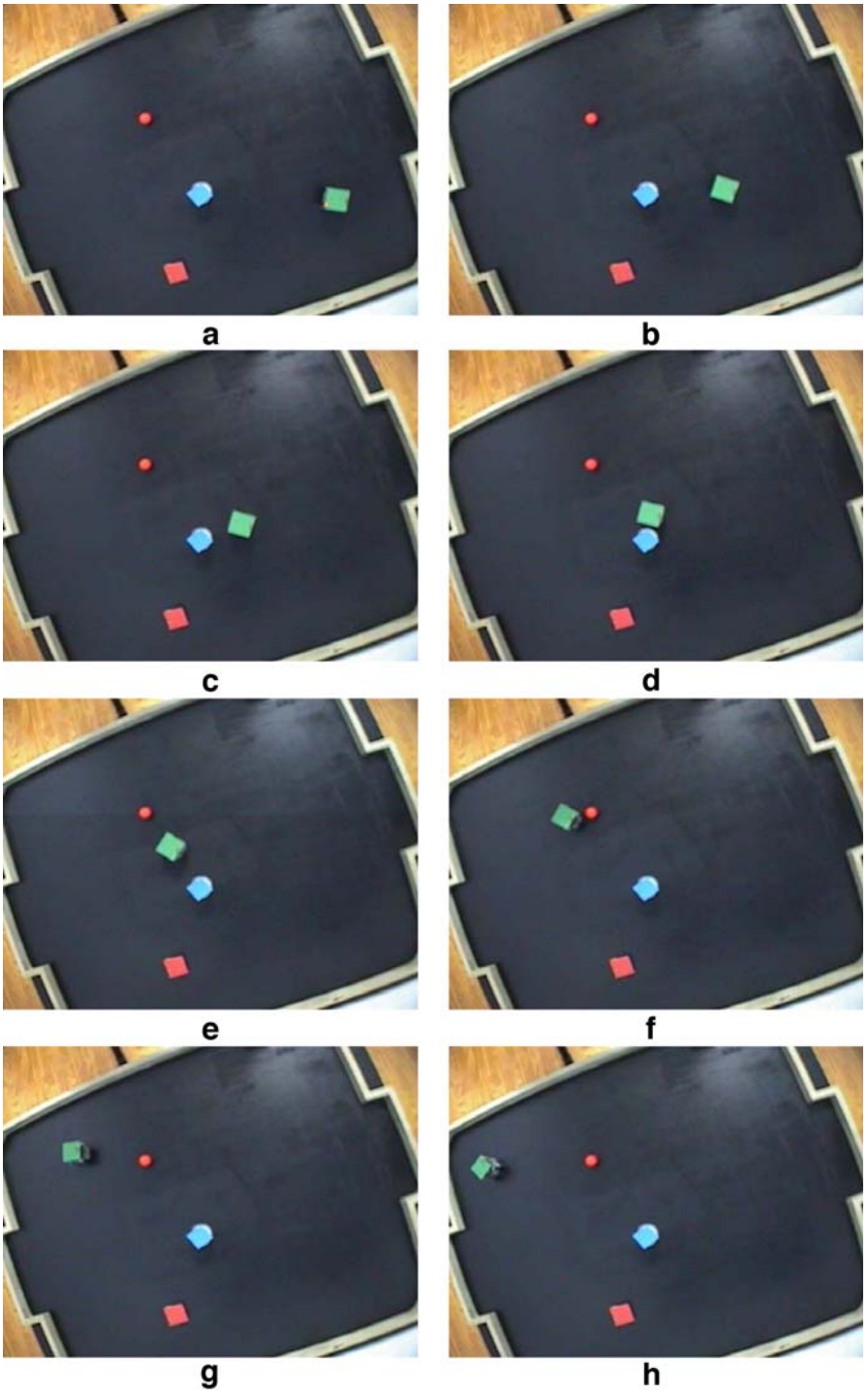


**Fig. 17** Positions of the robot among three static obstacles at eight instants of time – Approach 2

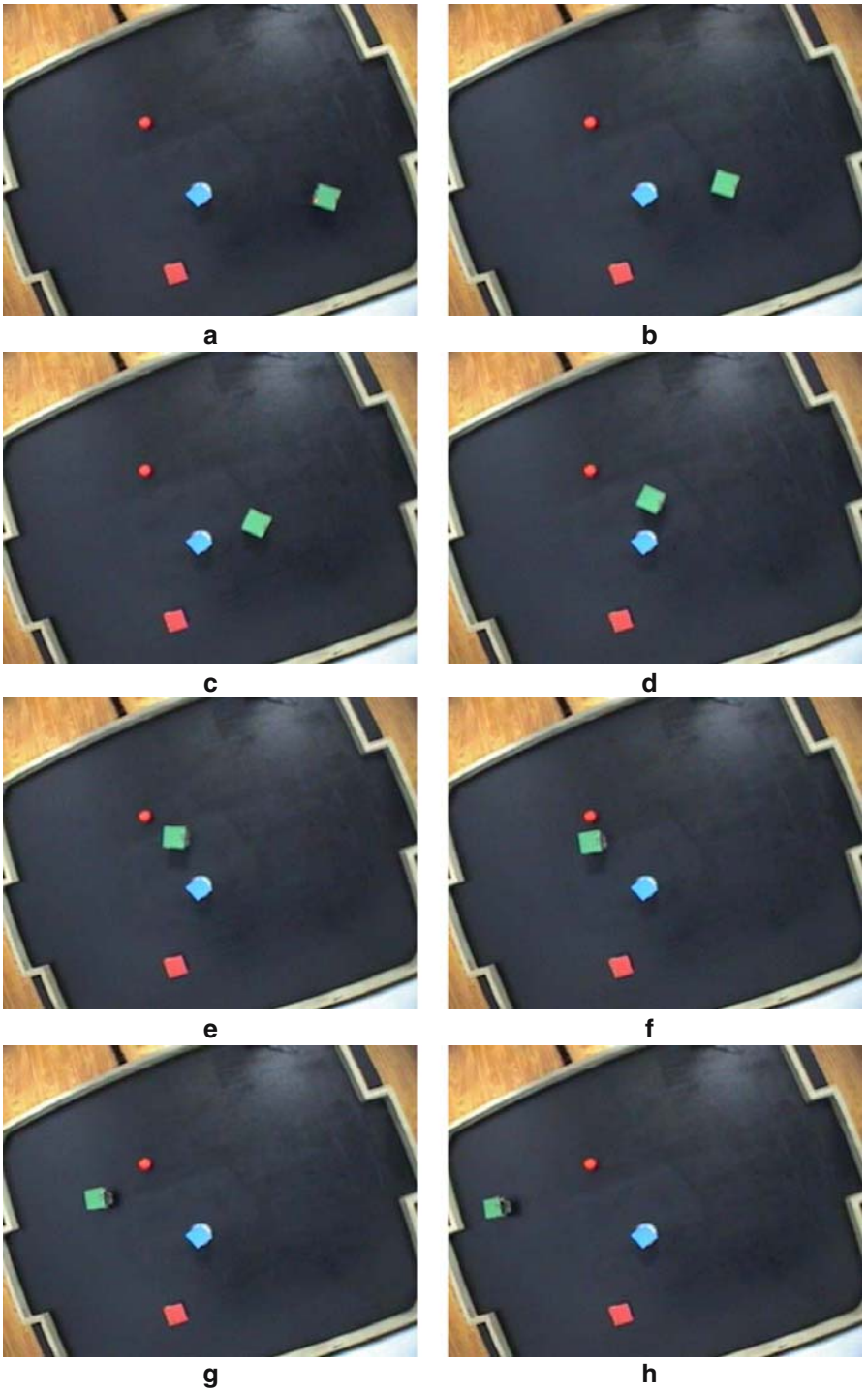


**Fig. 18** Positions of the robot among three static obstacles at eight instants of time – Approach 3

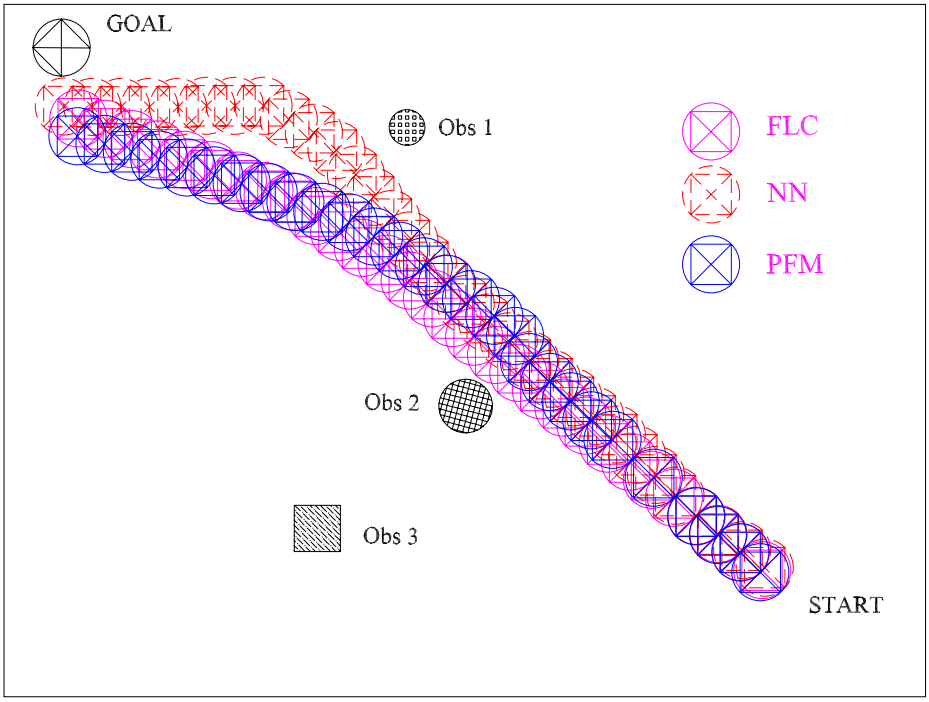


**Fig. 19** Movement of the robot among three static obstacles in Scenario 1 of Table 7

Approach 3 does not have any in-built optimization module and the robot may also get trapped into the dead-lock situation. The positions of the robot among the obstacles for eight successive instants of time are shown in Figs. 16, 17 and 18, respectively, as obtained by the three approaches.The paths planned by the robot among three static obstacles by following all three approaches are shown in Fig. 19 for the first test scenario of Table 7.

It is to be noted that although the robot travels the maximum path by following Approach 2, it moves with the reasonably high velocity. On the other hand, the robot moves very slowly in Approach 3, while avoiding any critical obstacles. Thus, the traveling time taken by the robot in Approach 3 is found to be more compared to that in Approaches 1 and 2. Moreover, in Approach 3, the speed of the robot decreases, as it comes nearer to the goal. It is interesting to note that some actuation error of the motors has been noticed, while performing the experiment. It may happen due to the following reasons: (a) time lag between the consecutive communication signals, (b) fluctuations in the supplied voltage of the battery.

In the present study, only one obstacle has been treated as the most critical one during a time step and the robot plans its collision-free path accordingly. Thus, the developed motion planners will be able to tackle the problems involving different number of obstacles present in the environment.

## 6 Stability Analysis

In the present study, an attempt is made to test the stability of the navigation system based on the Lyapunov's stability theory, which is explained below.

**Lyapunov theory:** Assume that there exist a scalar function $V$ of the state $\mathbf{q} = [x, y, \theta]^T$, with continuous first order derivatives such that

1. $V(\mathbf{q})$ is a positive definite,
2. $\dot{V}(\mathbf{q})$ is a negative definite,
3. $V(\mathbf{q}) \to \infty$ as $\| \mathbf{q} \| \to \infty$.

If the above conditions are satisfied, then the equilibrium point at the origin is globally asymptotically stable.

Let us consider a Lyapunov function as mentioned below.

$$V(\mathbf{q}) = \frac{1}{2}\mathbf{q}A\mathbf{q}^T = \frac{1}{2}[h(x^2 + y^2) + \theta^2] \tag{7}$$

Now, to satisfy the first criterion of the Lyapunov stability, the value of $h$ will have to be positive. Thereafter, differentiating the above equation with respect to the time, we get,

$$\dot{V}(\mathbf{q}) = h(x\dot{x} + y\dot{y}) + \theta\dot{\theta} = hv_t(x\cos\theta + y\sin\theta) + \theta\omega, \tag{8}$$

where $\dot{\mathbf{q}} = [\dot{x}, \dot{y}, \dot{\theta}]^T = [v_t x\cos\theta, v_t y\sin\theta, \omega]^T$ Thus, the point $\mathbf{q}$ will be globally asymptotically stable, if and only if $\dot{V}(\mathbf{q}) < 0$, i.e.,

$$hv_t(x\cos\theta + y\sin\theta) + \theta\omega < 0$$
$$or,\quad \omega < -\frac{hv_t(x\cos\theta + y\sin\theta)}{\theta}. \tag{9}$$

To equalize the above in-equation, we consider that there is some noise in the system, which forces the system towards the unstable zone. Let us assume that the above in-equation (9) can be converted into an equation as follows.

$$\omega = -k\theta - \frac{hv_t(x\cos\theta + y\sin\theta)}{\theta}, \tag{10}$$

where both $k$ and $h$ are positive quantity.

Now, to study the present stability condition in a more deeper sense, two assumptions are made as below.

**Assumption 1** The robot is allowed to navigate in the first quadrant only, i.e., both the initial position $(x_{\text{init}}, y_{\text{init}})$ as well as the goal position $(x_{\text{goal}}, y_{\text{goal}})$ are assumed to be lying in the first quadrant. Therefore, $x_{\text{init}}, y_{\text{init}}, x_{\text{goal}}, y_{\text{goal}} \geq 0$.

**Assumption 2** The robot is allowed to move in the forward direction only, i.e., tangential velocity $(v_t)$ is always positive $(0 \leq v_t \leq v_{\max})$.

In the present study, the main aim of a navigation scheme is to determine acceleration ($a$) and deviation ($\theta_1$) of the CG of the robot, necessary to avoid collision with the obstacles as discussed earlier. Now, from the values of $a$ and $\theta_1$, tangential velocity ($v_t$) and steering rate ($\omega$) of the CG of the robot are calculated. Moreover, the direction ($\theta$) along which the robot is moving can also be obtained from the previous time step calculation. Thus, by knowing $v_t$, $\omega$ and $\theta$, it is possible to calculate the value of $k$ by following the Eq. 10. Now, if $k$ comes out to be positive, then the system at that point will be stable. On the other hand, if it is found to be negative, then the system at that particular point will not be stable.

Let us take an example, in which the robot's present position is at (958, 703 mm) and present direction of movement is seen to be 2.7 radian with the positive $X$-axis of the global coordinate system. At this position, the motion planner's output, i.e., acceleration and deviation are found to be equal to 124.15 mm/s$^2$ and 1.22 radian, respectively. The velocity and steering rate of the CG of the robot before turning are calculated and those are found to be equal to 29.7 mm/s and 0.019 rad/s, respectively. The value of k is then calculated by following Eq. 10 and it is found to be equal to 77.588. Thus, the robot is found to be stable at the above point.

In the developed motion planning schemes, the above mentioned stability analysis of the robot has been carried out at each time step before the execution of its movement. If the robot is found to be unstable during a time step, it has been stopped during the said step but it will continue its planning for the next time step.

## 7 Comparisons with Others' Work

The prime aim of this study is to design and develop an adaptive robot motion planner that can plan and control the motion of a wheeled robot, navigating among some static obstacles. In the past, several attempts were made by various investigators to develop motion planners for the said purpose. Some of those are mentioned below for the purpose of comparison with the approaches developed in the present work.

Marichal et al. [18] 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. However, in their approach, they did not optimize the traveling time. Moreover, their method was tested among two kinds of static obstacles (rectangular and corner shaped). Li et al. [20] 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 using a fuzzy logic algorithm. However, the performance of their technique was not tested on a real robot. Gu and Hu [29] developed a path tracking scheme for a car-like robot based on a neural network. However, their model may fail to perform well in a situation, where the robot is subjected to some dynamic constraints, as those were not taken into account.

In the present work, both FL-based as well as NN-based motion planners have been developed to generate time-optimal, collision-free path of a real mobile robot navigating in the presence of some static obstacles. Both kinematic as well as dynamic constraints of the robot have been considered. GA-based optimization has been attempted to eliminate the local minima problem associated with the back-propagation algorithm.

## 8 Concluding Remarks

The prime aim of this study is to develop an adaptive navigation schemes for a real car-like robot moving among some static obstacles. A fuzzy logic-based and a neural network-based motion planner had been developed by the authors in the past (refer to the references [34, 55]). However, performances of those developed motion planners were tested through computer simulations only. In the present study, an attempt is made to test the effectiveness of both the fuzzy logic as well as neural network-based motion planners on a real robot to identify the best one in terms of traveling time and adaptability. Training to both the FL and NN-based motion planner is given off-line using a GA. During training, computational complexity involved to converge to a fixed accuracy level for the FL-based approach is seen to be low compared to that of the NN-based motion planner. It may be due to the fact that a longer binary string is required to represent an NN compared to that necessary for indicating an FLC. Once the optimization is over, performances of both these soft computing-based approaches are compared among themselves and with that of a conventional potential field method (i.e., Approach 3) for solving the navigation problems of a real wheeled robot. The performances of both the soft computing-based approaches are found to be comparable. The traveling time taken by the robot by following Approach 3 has come out to be the maximum in most of the cases. It may be due to the reason that there is no in-built optimization module. Moreover, some cases have been noticed, where the robot in Approach 3, has failed to find any feasible solution. This may happen when the repulsive potential balances the attractive potential. Some other important features have been revealed during the experimentation, which are mentioned below.

- Performances of the soft computing-based approaches are found to be better than the conventional potential field method in terms of traveling time taken by the robot to reach the goal.
- If at any instant of time, motion of the robot is restricted due to its kinematic and/or dynamic constraints, the motion planner is unable to provide with any other feasible solution. This is commonly known as the dead-lock situation. It has happened due to the fact that the positions of the obstacles are fixed. It is to be noted that Approach 3 has yielded a maximum number of dead-lock situations.
- As the attractive potential force decreases linearly, when the robot comes closer to the goal, the motion planner in Approach 3 is unable to yield a high value of acceleration, irrespective of the obstacle's position in the environment.
- Although Approach 2 sometimes has generated the longest distance path, its performance in terms of traveling time is not found to be the worst, due to the relatively higher speed of the robot during its movement.

The soft computing-based navigation schemes have come out to be promising for the development of intelligent and autonomous robots. However, design and development of a suitable soft computing-based motion planner is not an easy task. Once optimized, they will perform in an optimal sense and provide with some feasible solutions in an adaptive manner, so that it will be able to tackle some unknown situations effectively.

## 9 Scope for Future Work

Motion planning problems of a mobile robot among a few static obstacles have been considered in the present study. It will be more interesting and difficult too, to tackle the motion planning problems of a mobile robot moving in the presence of some moving obstacles. The modified versions of the developed algorithms may provide with some feasible solutions to the said problem. However, the soft computing-based approaches are expected to be more adaptive compared to the conventional potential field approach. Moreover, it will be more interesting to study the coordination issues of multiple mobile robots working in a common environment. In a dynamic environment, the robot plans its motion based on the collected information with the help of a vision system, on-line. Thus, any delay in processing the vision algorithm may deteriorate the performance of the robot significantly. The authors are working on these issues currently.

## References

1. Latombe, J.C.: Robot Motion Planning. Kluwer (1991)
2. Pratihar, D.K.: Algorithmic and soft computing approaches to robot motion planning. Mach. Intell. Robot. Control **5**, 1–16 (2003)
3. Lozano-P'erez, T.: A simple motion planning algorithm for general robot manipulators. IEEE J. Robot. Autom. **3**, 224–238 (1987)
4. Leven, D., Sharir, M.: Planning a purely translational motion for a convex object in two-dimensional space using generalized voronoi diagrams. Discrete Comput. Geom. **2**, 9–31 (1987)
5. Liu, Y.H., Arimoto, S.: Path planning using a tangent graph for mobile robots among polynomial and curved obstacles. Int. J. Rob. Res. **11**, 376–382 (1992)
6. Brooks, R.: Solving the find-path problem by good representation of free space. IEEE Trans. Syst. Man Cybern. **SMC-13**, 190–197 (1983)
7. Lozano-P'erez, T.: Spatial planning: a configuration space approach. IEEE Trans. Comput. **C-32**, 108–120 (1983)
8. Kavraki, L.E., Sevstka, P., Latombe, J.C., Overmars, M.: Probabilistic road maps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **13**, 566–580 (1996)
9. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Rob. Res. **5**, 90–98 (1986)
10. Bemporad, A., Luca, A.D., Orilo, G.: Local incremental planning for a car-like robot navigating among obstacles. In: Proceedings of the IEEE Conference Robotics and Automation, Minneapolis, Minnesoia, pp. 1205–1211 (1996)
11. Barraquand, J., Langlois, B., Latombe, J.C.: Numerical potential field techniques for robot path planning. IEEE Trans. Syst. Man Cybern. **22**, 224–241 (1992)
12. Borenstein, J., Koren, Y.: Potential field methods and their inherent limitations for mobile robot navigation. In: Proceedings of the IEEE Conference on Robotics and Automation, Sacramento, CA, pp. 1398–1404 (1991)
13. Borenstein, J., Koren, Y.: Real time obstacle avoidance for fast mobile robots. IEEE Trans. Syst. Man Cybern. **19**, 1179–1187 (1989)
14. Borenstein, J., Koren, Y.: The vector field histogram – fast obstacle avoidance for mobile robots. IEEE Trans. Robot. Autom. **7**, 278–288 (1991)
15. Slack, M.: Navigation template: mediating qualitative guidance control in mobile robots. IEEE Trans. Syst. Man Cybern. **23**, 452–466 (1993)
16. Fraichard, T., Garnier, P.: Fuzzy control to drive car-like vehicles. Robot. Auton. Syst. **34**, 1–22 (2001)
17. Abdessemed, F., Benmahammed, K., Monacelli, E.: A fuzzybased reactive controller for a non-holonomic mobile robot. Robot. Auton. Syst. **47**, 31–46 (2004)
18. Marichal, G., Acosta, L., Moreno, L., Mendez, J., Rodrigo, J., Sigut, M.: Obstacle avoidance for a mobile robot: a neuro fuzzy approach. Fuzzy Sets Syst. **124**, 171–179 (2001)

19. Song, K.T., Sheen, L.H.: Heuristic fuzzy-neuro network and its application to reactive navigation of a mobile robot. Fuzzy Sets Syst. **110**, 331–340 (2000)
20. Li, W., Ma, C., Wahl, F.M.: A neuro-fuzzy system architecture for behavior based control of a mobile robot in unknown environments. Fuzzy Sets Syst. **87**, 133–140 (1997)
21. Maaref, H., Barret, C.: Sensor- based navigation of a mobile robot in an indoor environment. Robot. Auton. Syst. **38**, 1–18 (2002)
22. Pratihar, D.K., Bibel, W.: Near-optimal, collision-free path generation for multiple robots working in the same workspace using a genetic-fuzzy systems. Mach. Intell. Robot. Control **5**, 45–58 (2003)
23. Hui, N.B., Pratihar, D.K.: Mobile robot navigation: potential field approach vs. genetic-fuzzy system. In: Proceedings of 10th On-line World Conference on Soft Computing (WSC10) (2005)
24. Yang, S.X., Meng, M.: An efficient neural network approach to dynamic robot motion planning. Neural Netw. **13**, 143–148 (2000)
25. Mondada, F., Floreano, D.: Evolution of neural control structures: some experiments on mobile robots. Robot. Auton. Syst. **16**, 183–195 (1995)
26. Nolfi, S., Parsi, D.: Learning to adapt to changing environments in evolving neural networks. Adapt. Behav. **5**, 75–98 (1997)
27. Yamada, S.: Evolutionary behavior learning for action based environment modeling by a mobile robot. Applied Soft Computing **5**, 245–257 (2005)
28. Pal, P.K., Kar, A.: Sonar-based mobile robot navigation through supervised learning on a neural net. Auton. Robots **3**, 355–374 (1996)
29. Gu, D., Hu, H.: Neural predictive control for a car-like mobile robot. Robot. Auton. Syst. **39**, 73–86 (2002)
30. Goffe, W.L., Ferrier, G.D., Rogers, J.: Global optimization of statistical functions with simulated annealing. J. Econom. **60**, 65–99 (1994)
31. Koza, J.: Genetic Programming. The MIT Press (1992)
32. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, Machine Learning. Addison-Wesley, Reading, MA, USA (1989)
33. Pratihar, D.K.: Evolutionary robotics - a review. Sadhana **28**, 999–1003 (2003)
34. Hui, N.B., Pratihar, D.K.: Neural network-based approaches vs. potential field approach for solving navigation problems of a car-like robot. Mach. Intell. Robot. Control **6**, 29–60 (2004)
35. Thrun, S.: An approach to learning mobile robot navigation. Robot. Auton. Syst. **15**, 301–309 (1996)
36. Kim, J.H., Kim, D.H., Kim, Y.J., Seow, K.T.: Soccer robotics. Springer, Amsterdam (2004)
37. Bruce, J., Veloso, M.: Real-time randomized path planning for robot navigation. In: Proceedings of the International Conference on Intelligent Robots and Systems, EPFL, Lausanne, Switzerland, pp. 2383–2388 (2002)
38. Akbarzadeh, M., Kumbla, K., Tunstel, E., Jamshidi, M.: Soft computing for autonomous robotic systems. Comput. Electr. Eng. **26**, 5–32 (2000)
39. Thorpe, C., Hebert, M., Kanade, T., Shafer, S.: Vision and navigation for carnegie-mellon navlab. IEEE Trans. Pattern Anal. Mach. Intell. **10**, 362–373 (1988)
40. Davis, L.: Visual navigation at the university of maryland. Robot. Auton. Syst. **7**, 99–110 (1991)
41. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. **23**, 23–33 (1997)
42. Chen, K.H., Tsai, W.H.: Vision-based obstacle detection and avoidance for autonomous land vehicle navigation in outdoor roads. Autom. Constr. **10**, 1–25 (2000)
43. Ohya, A., Kosaka, A., Kak, A.: Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. IEEE Trans. Robot. Autom. **14**, 969–978 (1998)
44. Zhang, S., Xie, L., Adams, M.D.: Feature extraction for outdoor mobile robot navigation based on a modified Gauss–Newton optimization approach. Robot. Auton. Syst. **54**, 277–287 (2006)
45. Winters, N., Santos-Victor, J.: Information sampling for vision-based robot navigation. Robot. Auton. Syst. **41**, 145–159 (2002)
46. Choi, S.Y., Lee, J.M.: Applications of moving windows technique to autonomous vehicle navigation. Image Vis. Comput. **24**, 120–130 (2003)
47. Tanaka, K., Sugeno, M.: Stability analysis and design of fuzzy control systems. Fuzzy Sets Syst. **45**, 135–156 (1992)
48. Tanaka, K., Sano, M.: Stability analysis of neural networks using stability conditions of fuzzy systems. In: Second IEEE International Conference on Fuzzy Systems, vol. 1, pp. 422–428 (1993)

49. Tanaka, K.: Stability and stabilizability of fuzzy-neural-linear control systems. IEEE Trans. Fuzzy Syst. **3**, 438–447 (1995)
50. Tanaka, K., Sano, M.: Concept of stability margin for fuzzy systems and design of robust fuzzy controllers. In: Second IEEE International Conference on Fuzzy Systems, vol. 1, pp. 29–34 (1993)
51. Thathachar, M.A.L., Vishwanath, P.: On the stability of fuzzy systems. IEEE Trans. Fuzzy Syst. **5**, 145–151 (1997)
52. Calcev, G.: Some remarks on the stability of mamdani fuzzy control systems. IEEE Trans. Fuzzy Syst. **6**, 436–442 (1998)
53. Guo, S., Huang, L.: Stability analysis of cohen-grossberg neural networks. IEEE Trans. Neural Netw. **17**, 106–116 (2006)
54. Zhang, J., Jin, X.: Global stability analysis in delayed hopfield neural models. Neural Netw. **13**, 745–753 (2000)
55. Hui, N.B., Pratihar, D.K.: Automatic design of fuzzy logic controller using a genetic algorithm for collision-free, time-optimal navigation of a car-like robot. International Journal of Hybrid Intelligent Systems **5**, 161–187 (2005)
56. Biswas, D.K.: Path planning of multiple robot working in the same workspace - potential field approach. Master's thesis, Regional Engineering College, Durgapur, India (2003)