# Development of a technology for car's auto-parking using swarm search-based fuzzy control system

Mohamed Hanafy*, Mostafa M. Gomaa, Mohamed Taher and Ayman M. Wahba

Department of Computer and Systems Engineering,
Faculty of Engineering,
Ain Shams University,
Abbasia, Cairo, 11517, Egypt
E-mail: ta.m.hanfy@sha.edu.eg
E-mail: m_gomaa_eg@yahoo.com
E-mail: mtaher2000@gmail.com
E-mail: ayman.wahba@gmail.com
*Corresponding author

**Abstract:** This paper introduces a new technique for automatic column (parallel) parking of four-wheeled cars. The intended technique is divided into two main stages. The first stage is the generation of the optimal path trajectory that the car can follow so that it can be parked in a parking lot with the minimum acceptable dimensions, after the discovery of the parking place. This stage is supported with a full parking scenario which is suitable for practical implementation. The second stage is the path tracking problem, which is solved by the design of the intended fuzzy control system. The parameters of the fuzzy controller beside the required time for parking achievement all are tuned by the search algorithm artificial fish swarm algorithm (AFSA). The performance of the involved swarm-based fuzzy controller (SBFC) is measured for solving the path tracking problem while operating on the dynamic model of a car-like robot, so that a full practical technique for car's auto parking can be completely realised. The simulation results for the path tracking problem are emphasised at the end of the paper reflecting the success of the intended auto parking control system.

**Keywords:** automatic parking; swarm algorithm; fuzzy control; car's dynamic model; optimal path generation; parking scenario; path tracking problem; swarm search.

**Reference** to this paper should be made as follows: Hanafy, M., Gomaa, M.M., Taher, M. and Wahba, A.M. (2012) 'Development of a technology for car's auto-parking using swarm search-based fuzzy control system', *Int. J. Modelling, Identification and Control*, Vol. 17, No. 1, pp.85–97.

**Biographical notes:** Mohamed Hanafy is an Assistant Lecturer at Communication, Computer and Electronics Engineering Department, High Institute of Engineering, Shorouk City, Cairo, Egypt. He received his BS in Computer and Systems Engineering in 2006 from Computer and Systems Engineering Department, Ain Shams University. His research interests include intelligent control systems, and embedded systems.

Mostafa M. Gomaa is a Professor at Computer and Systems Engineering Department, Ain Shams University, Cairo, Egypt. He received his BSc and MSc in Electrical Engineering in 1989 and 1993, respectively, from Ain Shams University. He received his MSc and PhD in Systems Engineering in 1994 and 1997, respectively, from Grenoble University, Grenoble, France. His research interests include fault detection and diagnosis, intelligent control systems, hybrid systems modelling, robotics, simulation and control, and Petri net theory and applications.

Mohamed Taher is an Assistant Professor at Computer and Systems Engineering Department, Ain Shams University, Cairo, Egypt. He received his BS and MS in Electrical and Computer Engineering from Ain Shams University and PhD in Computer Engineering from the George Washington University, DC, USA. His research interests include high performance computing, parallel and distributed systems, reconfigurable computing, and embedded systems.

Ayman M. Wahba is a Professor at Computer and Systems Engineering Department, Ain Shams University, Cairo, Egypt. He received his BSc and MSc in Electrical Engineering from the same university in 1988 and 1991, respectively. He then moved to France where he received his Diplôme d'Études Approfondies (DEA) degree from the Institut National Polytechnique de Grenoble (INPG) in 1993, and then he received his PhD in Computer Engineering from the TIMA Laboratory, Joseph Fourier University in 1997. His research interests include high performance digital IC design, formal verification techniques, diagnosis of design errors, high level synthesis, design and test of computers, and embedded systems.

## 1 Introduction

### 1.1 Problem definition

One of the most important intelligent technologies applied to automobiles in recent years is the automatic parking technology. Parallel parking as named in Paromtchik and Laugier (1997) or column parking as named in Inoue et al. (2004) has the greatest importance for drivers to park their cars in narrow roads and traffic crowded streets highly spread at all cities.

Column parking means that all the cars should be parked in series to each other on one column (parking lane) parallel to the road curb or the street pavement.

In order to achieve column parking, the driver first discovers the parking lot suitable for his car, stops the car, and then moves backward in a curved path to manoeuvre the lot boundaries such that the car is placed inside the parking lot. Finally, he may need to move forward and backward usually many times until the car fits better the parking lot.

Many drivers suffer difficulties or make errors while column parking or pulling out of a parking place. Thus, a control approach to automatic parallel parking and pulling out manoeuvres has become a vital demand in recent years.

### 1.2 Our contribution

The main target for this work is to introduce a new efficient technique for car's auto-parking (column parking) based on an intelligent fuzzy controller whose all parameters (for fuzzification, defuzzification, and inference) are tuned by a fish swarm search algorithm, such that the swarm-search-based fuzzy controller (SBFC) can control a car to follow a certain designed optimal path so that column parking can be performed safely in short time.

The intended car's auto-parking technique is well tested on a rear-drive four-wheeled car model giving out splendid simulation results. Our contribution is developed in a group of sequential steps:

1 The optimal tracking path for column parking is generated, so that the car can be parked safely (without any collisions with park boundaries) in one shot track with minimum path length, in parking lot with the minimum length suitable for the car model dimensions.

2 Operating on the car's dynamic model, a four-input two-output fuzzy controller is designed with its rule base is estimated so that the control action behaviour tends to follow the designed path trajectory, where its inputs are the errors in the two-dimensional coordinates of the centre of the rear wheel axle between the desired path trajectory and the actual path generated by the controller, and the derivatives of these errors, while outputs are the voltage signals input to both the driving and the steering DC motors.

3 The fish swarm search algorithm is used to search for the designed fuzzy controller parameters (ten parameters as will be shown later) and the parking time (the elapsed time in following the intended path trajectory) that achieves the nearest generated path to the desired one with minimum errors. The swarm-search-based fuzzy controller (SBFC) obtained generates a path trajectory that is very close to the desired one as will be shown in the simulation results.

4 The performance of the intended swarm-search-based fuzzy controller (SBFC) is measured while applying external disturbance forces on the dynamic model of the car-like robot as will be shown in the simulation results reflecting the robustness of the involved control system.

### 1.3 Research works in this domain

Although the car parking can be carried out by the designed feedback laws and switching process discussed in Dao and Liu (2006), it however leads to the need for the car's motion forward and backward many times. Moreover, if a path planning approach is taken, the planned path should be a combination of circles and straight lines – which is also the case for the approach discussed in Hsieh and Özguner (2008) – that guarantee that the car does not get out of admissible parking space, where motion along straight lines is controlled by the designed feedback laws and the motion along circles is controlled by feedforward control, leading to a longer path compared to the piecewise 2nd order curves introduced in this paper.

The control approaches used in Dao and Liu (2006), and Inoue et al. (2004) use the chained form conversion as one of the non-linear control methods to control the steering and speed. However, it is not easy to implement in practice because of highly non-linear and computationally expensive conversion equations.

The control schemes introduced in Abdelmoula et al. (2009), and Miah, and Gueaieb (2007) do not deal with the column parking problem investigated in this paper; rather the parallel parking mentioned in Abdelmoula et al. (2009) and Miah and Gueaieb (2007) differs from the parallel parking mentioned in the abstract of this paper. In order to prevent misunderstanding, it is important to note that the parallel parking mentioned in Abdelmoula et al. (2009), and Miah and Gueaieb (2007) is equivalent to the row parking mentioned in Inoue et al. (2004), while the parallel parking mentioned in Paromtchik and Laugier (1997) and in this paper is equivalent to the column parking mentioned in Inoue et al. (2004).

The research works in Dao and Liu (2006), and Hsieh and Özguner (2008) are also applied in the simulation results only on the row parking, while the control approach in Inoue et al. (2004) is simulated for column parking with a detailed procedure to determine its desired path trajectory. Despite of the efficiency of the generated path, it is not clear for the practical implementation as the centres of the circles are not determined by a practical parking scenario, and the parking lot dimensions are not taken in consideration.

The method developed in Paromtchik and Laugier (1997) has been tested on an experimental automatic car designed on the base of a Ligier electric car. It is concerned mainly with the column parking and pulling out manoeuvres leading to the forward and backward motions many times, without taking in consideration the timing cost and the parking lot dimensions efficiency.

Since man can usually perform steering control easily after acquisition of appropriate experience without deep understanding about the vehicle, then an intelligent control scheme such as fuzzy logic controller will be effective to achieve an automated vehicle control system. Because a fuzzy logic controller can be specified by a set of parameters, design of a fuzzy logic controller can be formulated as a parameter search problem for which evolutionary multi-objective optimisation approach in Lee et al. (2006) has shown a significant efficiency. The fuzzy logic controller designed in Lee et al. (2006) has two inputs (the errors in the x-direction and the orientation angle) and one output (the steering angle) assuming that the velocity in the backward direction is constant and the car is far from the parking lot along the y-direction so that the vertical position error is not considered. Thus, the designed fuzzy controller in Lee et al. (2006) is applicable only for special cases, while the controller involved in this paper which has four inputs (the errors in the two-dimensional coordinates, and their derivatives) and two outputs (the driving velocity and the steering angle) is more general in its cases of application.

As a new technique for solving the car parking problem, Imae et al. (2009) propose a real-time optimisation approach to the parallel-parking control problem, focusing on the lateral and yaw dynamics. In this case, there exists singularity problem and rank problem. Both the parking space constraints and the car's full dimensions have not been taken under consideration. Thus, inspite of its efficiency, the path trajectory generated by Imae et al. (2009) is not verified for practical implementation.

Li et al. (2010) have conducted a thorough study, including kinematic control design, simulations, and practical experiments, for the wall-following with a controller based on back propagation neural networks and the auto-parking using a fuzzy speed controller for the motion control of a car like mobile robot, without a clear identification of the control technique used for car auto-parking and the tracking of the parking path trajectory based on the car's kinematic model rather than the dynamic one.

Jallouli et al. (2010) have introduced a fuzzy logic controller to control the speeds at each wheel of a car-like robot to innovate a car autonomous navigation control system. The fuzzy logic controller is tuned using a genetics algorithm. The designed control system is built on the car's kinematics rather than the dynamics one. Moreover, their work deals with the car's navigation from a point to another without focusing on the parallel parking automation and its accompanied problems presented in optimising the parking space dimensions, the parking path, and the path tracking.

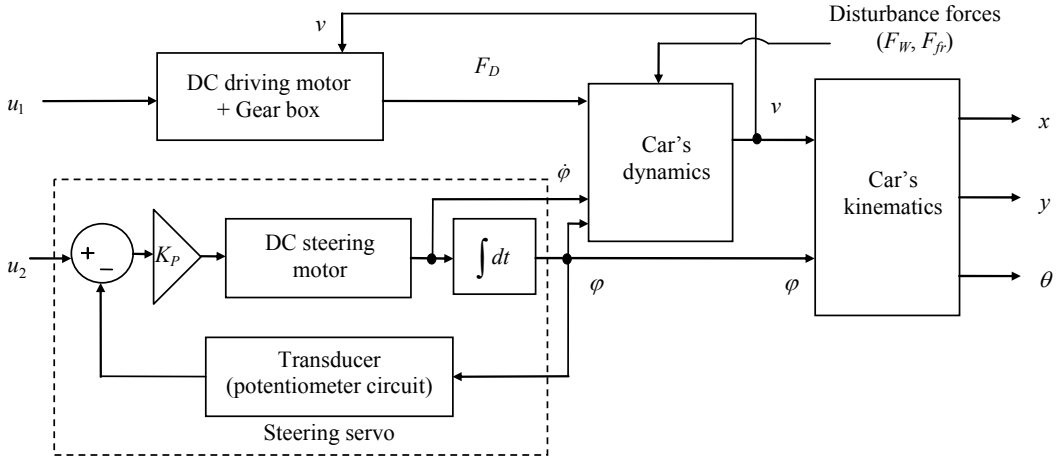
### *1.4 Paper structure*

The remainder of this paper is as follows: Section 2 investigates the dynamic model of rear-drive four-wheeled car-like robot which is the plant of the involved control system. Section 3 reveals the generation of the optimal parking path trajectory guided with the parking scenario steps. Section 4 explains the design of the swarm-search-based fuzzy controller (SBFC) in two main sections; the first one is the design of the fuzzy controller itself with its main parts, while the second is the application of the fish swarm algorithm to search for the controller parameters achieving the required trajectory. Section 5 presents simulation results to show the efficiency of the intended control system. Finally, Section 6 reveals the conclusions and the future work.

## 2 Dynamic model

According to the mathematical representation derived by Moret (2003), the dynamic model for the car-like robot – as shown in Figure 1 – can be divided into four main parts as follows.

**Figure 1** Block diagram for the dynamic model of the car-like robot



### 2.1 Car's kinematics

According to the rear-drive four-wheeled car model shown in Figure 2, with the assumption that there is no side slipping or skidding (the car must move in the direction of its wheels), the kinematic model can be illustrated in the following differential non-linear equations relating the outputs ($x$, $y$, $\theta$) with the inputs ($v$, $\varphi$):

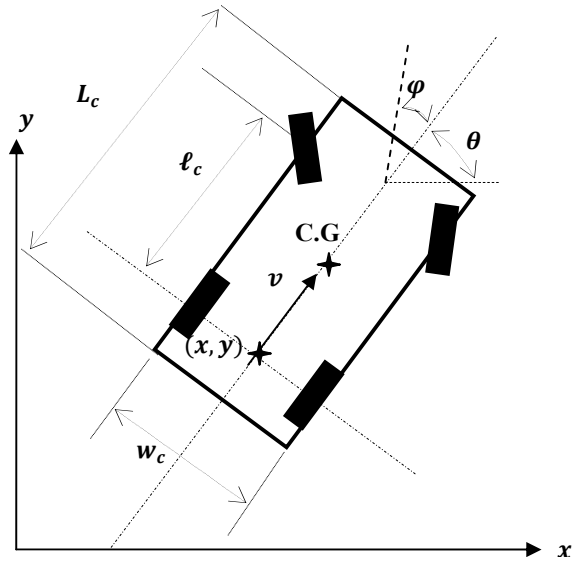
$$\dot{x} = v \times \cos\theta \tag{1}$$

$$\dot{y} = v \times \sin\theta \tag{2}$$

$$\dot{\theta} = \frac{\tan\varphi}{\ell_c} \times v \tag{3}$$

where ($x$, $y$) is the coordinate of the centre of rear wheel axle, $L_c$ is the total car length, $\ell_c$ is the wheel base, $w_c$ is the car width, $v$ is the rear driving velocity, $\varphi$ is the steering angle measured with respect to the car's axis of symmetry, and $\theta$ is the orientation angle of the car measured with respect to the x-direction, the positive direction for both $\varphi$ and $\theta$ is anti-clockwise. Moreover, for the symbols shown in Figure 1; $F_D$ is the rear driving force, $u_1$ and $u_2$ are the signal voltages input to the driving and the steering DC motors, respectively.

**Figure 2** Rear-drive four-wheeled car model




### 2.2 Modelling of the driving DC motor and the attached gear box

The differential equation representing the rear-drive system (Driving DC motor + Gear box + Rear wheel axle) of our car model:

$$\left(L_{a_D} B_{m_D} + R_{a_D} J_{m_D}\right)\dot{v} + \left(R_{a_D} B_{m_D} + K_D^2\right)v = n \times r_w + K_D u_1 - R_{a_D} F_D - L_{a_D} \dot{F}_D \tag{4}$$

$$\dot{F}_D(t) = 2{,}000u_1(t) = 2{,}200v(t) - 2{,}000F_D(t) \tag{5}$$

### 2.3 Modelling of the steering servo system

The transfer function representing the steering servo system shown in Figure 1.

$$\frac{\Phi(s)}{U_2(s)} = \frac{K_p K_s}{\left(L_{a_S} B_{m_S} + R_{a_S} J_{m_S}\right)s^2 + \left(R_{a_S} B_{m_S} + K_s^2\right)s + K_T k_p K_s} \tag{6}$$

By substituting with the values mentioned in Table 1, and simplifying the equation; it can be reduced to a second order system, then by adjusting the gain ($K_P$) of the proportional controller so that the time constant ($\tau_S$) ≈ 0.2 sec (which is a reasonable time constant for a DC servo system); we can obtain the following differential equation with $\varphi$ and $\dot{\varphi}$ are in degrees and degrees per second, respectively:

$$\dot{\varphi}(t) = -5\varphi(t) + 40u_2(t) \tag{7}$$

**Table 1** The values of the parameters of the intended car model (car-like robot) presented in SI units

| *Type of parameter* | *Name* | *Symbol* | *Value* |
|---|---|---|---|
| Car dimensions | The total car length | $L_c$ | 4 m |
| | The wheel base | $\ell_c$ | 3 m |
| | The car width | $w_c$ | 2 m |
| | The wheel radius | $r_w$ | 0.2 m |
| Parameters of the driving permanent magnet DC motor and the attached gear box | The armature resistance | $R_{ad}$ | 1 Ω |
| | The armature inductance | $L_{ad}$ | 0.5 mH |
| | The back emf, and torque constants | $K_{bD}$, $K_{iD}$ ($K_D$) | 1 Nm/A<br>1 V/rad/sec |
| | The inertia of motor's shaft | $J_{mD}$ | 0.1 e-4 Kg.m$^2$ |
| | The friction on motor's shaft | $B_{mD}$ | 0.1 Nms |
| | The gear ratio | $n$ | 5 |
| Parameters of the steering permanent magnet DC motor | The armature resistance | $R_{aS}$ | 0.5 Ω |
| | The armature inductance | $L_{aS}$ | 0.5 mH |
| | The back emf, and torque constants | $L_{bS}$, $K_{iS}$ ($K_S$) | 0.1 Nm/A<br>0.1 V/rad/sec |
| | The inertia of motor's shaft | $J_{mS}$ | 0.1 e-4 Kg.m$^2$ |
| | The friction on motor's shaft | $B_{mS}$ | 0.01 Nms |
| | The transducer gain | $K_T$ | 0.125 V/deg |
| Car's inertia of motion | The car's mass | $M$ | 24 Kg |
| | The car's moment of inertia about its C.G. | $J_c$ | 40 Kg.m$^2$ |

### *2.4 Modelling of the dynamics of the car motion*

By applying Newton's laws of motion on the two-wheeled simplified car model introduced by Moret (2003) with the addition of external disturbance forces represented by $F_{fr}$ as the frictional force between the wheel and the ground, and $F_w$ as the wind frictional force; we can obtain the following differential equation, assuming that the centre of gravity of the car is concentrated at its geometric centre as shown in Figure 1:

$$\dot{v} = -\left(\left(\frac{\ell_c^2 M}{4} + J_c\right)\frac{\tan\varphi}{\gamma}\right)v\dot{\varphi} + \left(\frac{\ell_c^2 \cos^2\varphi}{\gamma}\right)F_D - \left(\frac{\ell_c^2 \cos^2\varphi}{\gamma}\right)F_W - \left(\frac{\ell_c^2 \cos\varphi}{\gamma}(1+\cos\varphi)\right)F_f \tag{8}$$

where $\gamma = \left(\ell_c^2 M + \left(\frac{\ell_c^2 M}{4} + J_c\right)\tan^2\varphi\right)\cos^2\varphi.$

By substituting with the values mentioned in Table 1, with $\dot{\varphi}$ represented in radians per second; the equation becomes:

$$\dot{v} = -\left(\frac{94\tan\varphi}{\gamma}\right)v\dot{\varphi} + \left(\frac{9\cos^2\varphi}{\gamma}\right)F_D - \left(\frac{9\cos\varphi}{\gamma}\right)F_{fr} - \left(\frac{9\cos^2\varphi}{\gamma}\right)\left[F_W + F_{fr}\right] \tag{9}$$

where $\gamma = (216 + 94\tan^2\varphi)\cos^2\varphi$.

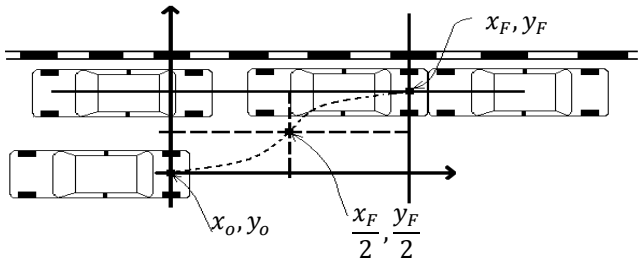
## 3 Path generation

The parking path trajectory by definition is the path which the car follows to move from a certain initial position outside the parking lot at which the car is stopped after discovering the suitable parking lot to a certain final position at the opposite far edge of the parking slot from which the car can slightly move forward only one time in order to best fit the parking lot. In our contribution, this trajectory is achieved by a one shot path (without any oscillations or moving forward and backward many times) represented in terms of the centre point of the car rear axle.

In order to obtain a closed form equation representing the parking path trajectory, two planar coordinates – as shown in Figure 3 – are placed with their origin at the centre point of the rear axle for the car initial position, and their positive directions are towards the backward direction of the car and towards the parking lane containing the parking lot for the x-axis and y-axis, respectively.

There are four main well-known boundary conditions for the column parking path trajectory. These facts are that the car initial position is ($X_o$, $Y_o$), the car final position is ($X_F$, $Y_F$), the car initial orientation is 180° ($\theta_o$ = 180°), and the car final orientation is 180° ($\theta_F$ = 180°). Thus, the path trajectory equation can be a third-order equation with four unknowns formed from these four boundary conditions. For better performance (instead of solving four equations for four unknowns), the path trajectory – as shown in Figure 3 – can be formed from two smooth and continuous piecewise second-order curves, where the first curve starts from the initial position ($X_o$, $Y_o$) and ends at the mid-point position ($X_F$ / 2, $Y_F$ / 2), and the second curve starts from the mid-point position ($X_F$ / 2, $Y_F$ / 2) and ends at the final position ($X_F$, $Y_F$). That is because the third-order curve has a derivative of a higher order (second-order) than the two piecewise second-order curves having first-order derivative, which means that the third-order curve changes more fast specially at the boundary limits (($X_o$, $Y_o$) and ($X_F$, $Y_F$)) compared to the two piecewise second-order curves, leading to a curve with sharper edges that is more difficult to be tracked by the designed controller.

**Figure 3** The piecewise parking path trajectory




For our intended piecewise second-order curves, $y(x) = ax^2 + bx + c$, the boundary conditions for the first curve are that the car initial position is ($X_o$, $Y_o$) where $X_o = Y_o = 0$, the car final position is ($X_F$ / 2, $Y_F$ / 2), and the car initial orientation is 180° ($\theta_o$ = 180°). The boundary conditions for the second curve are that the car initial position is ($X_F$ / 2, $Y_F$ / 2), the car final position is ($X_F$, $Y_F$), and the car final orientation is 180° ($\theta_F$ = 180°).

We can easily derive from the kinematic model equations that by dividing the equations (2) / (1):

$$\begin{aligned} &\therefore \frac{\dot{y}}{\dot{x}} = \frac{dy/dt}{dx/dt} = \frac{dy}{dx} = \tan\theta \\ &\because \theta = 180° \text{ at boundaries} \\ &\therefore \frac{dy}{dx} = 0 \text{ at boundaries} \end{aligned} \tag{10}$$

By solving the equations of the two piecewise second-order curves at the three boundary conditions of each curve, we can obtain the general form of the desired second-order curve as follows:

$$y(x) = \begin{cases} \left(\frac{2Y_F}{X_F^2}x^2 \Rightarrow 0 \le x \le \frac{X_F}{2}\right) \\ \left(\frac{-2Y_F}{X_F^2}\right)x^2 + \left(\frac{4Y_F}{X_F}\right)x - Y_F \Rightarrow \frac{X_F}{2} < x \le X_F \end{cases} \tag{11}$$

It is clear from the path trajectory equations that the parking path trajectory is totally dependent on the final position values $X_F$ and $Y_F$. There are three main parameters ($d$, $L$, $ss$) – as shown in Figure 4 – that specify the final position values $X_F$ and $Y_F$ according to the following equations:

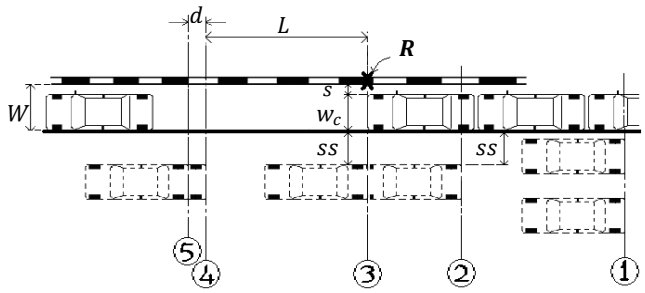
$$X_F = d + L, Y_F = ss + w_c \quad (12)$$

where $w_c$ is the car width as mentioned previously, $L$ is the length of the park lot, $d$ is the horizontal distance between the final backward edge of the car initial position and the start edge of the parking lot, $ss$ is the start shift or the vertical distance between the car initial position and the outer edge of the parking lane. There is also a fourth parameter which is the final shift ($s$) – as shown in Figure 4 – that represents the excess in the park lot width ($W$) than the car width ($w_c$) such that

$$W = w_c + s \quad (13)$$

where $s$ is the vertical distance (in positive direction of y-axis) between the cars final position and the internal park lot edge (road curb). So $W$ or $s$ beside $L$ represent the parking lot dimensions which are the measure of the admissibility of the parking lot for the car with its defined dimensions to be parked.

**Figure 4** The intended parking scenario




The question now is how the car can reach the initial position for column parking according to the preceding parameters. To answer this question, we intend an auto-parking scenario as a number of steps that the controller should follow so that the suitable parking lot can be discovered then the car can be parked successfully. The steps of the intended parking scenario as shown in Figure 4:

1 The car is at situation (1); the driver sends a command for the controller to auto-park, the controller uses two sensors, an upper sensor to detect the existence of a parked car, and a lower sensor to detect the internal edge of the parking lane (or the road curb). If the controller detects the existence of a parked car then it takes a decision either to get the car narrower or more far from the parked cars while moving forward and searching for the empty parking lot, so that the start shift ($ss$) distance between the car and the parked ones is adjusted to the specified value obtained from the search programme (situation (2) is reached). On the other hand, if the controller detects the internal edge of the parking lane (or the road curb) without the existence of a parked car then this case is identical to situation (3) that the controller can take a decision to go to step 3.

2 The car is at situation (2); the controller detects that the required value for the start shift ($ss$) between the car and the parked ones is reached, so the controller takes decision to stop getting narrower or more far from the parked cars and move forward only to preserve the achieved value of $ss$, while searching for a parking lot.

3 The car is at situation (3); the controller discovers a parking lot such that it detects the internal edge of the parking lane without the existence of a parked car. The controller considers this internal edge as a reference point ($R$). The distance between the car and this reference point should not be less that the value of ($ss + w_c + s$), where s is the minimum shift between the final position of the parked car and the road curb required for safe parking (without collisions) and its value is obtained from the search programme. The controller measures the distance between the car and the reference point ($R$), if it is found less than the value of ($ss + w_c + s$), then the controller takes decision to get the car far from the park lot while moving forward until the required distance ($ss + w_c + s$) is achieved. Moreover, the controller after detecting the reference point takes decision to move the car forward for a distance $L$ – which is the minimum park lot length suitable for the car to park also obtained from the search programme – while scanning the internal edge of the park (road curb) and adjusting the distance from it. Briefly, we can say that the controller at this step ensures that the parking lot detected has dimensions (length and width) suitable for the car to park in.

4 The car is at situation (4); the controller detects that the distance $L$ has been covered starting from the reference point with the lot edge (road curb) still detected (there is a park lot with length $L$). The controller takes decision to stop the car so that column parking can be started correctly.

5 The car is at situation (5); the car has reached the desired initial position for column parking. The car stopped after a distance ($d$) from the previous situation which has a minimum predefined value according to the inertia of motion of the car. The car is now ready to follow the parking path trajectory to park safely in the defined lot in one shot with minimum distance.

Search programme is performed to search for the values of ($L$, $ss$, $d$) that achieves the optimum path trajectory to park in lot with minimum length, beside the minimum value of ($s$) corresponding to this trajectory so that the car can park safely without any collisions. The specifications of the search programme are as follows:

- Parameters: $L$, $s$, $ss$, $d$
- Constraints:
  1 Safety (no collisions).

2 The parking path should be generated through an admissible steering angle $\varphi(t)$ ($-\varphi_{max} \le \varphi(t) \le \varphi_{max}$), for $t \in [0, T_F]$, where $\varphi_{max}$ is specified as one of the car properties ($\varphi_{max} = 40°$), and $T_F$ is the time at which the car reaches the final position. Every generated path is tested for its admissibility by performing inverse-kinematics to obtain the corresponding driving velocity ($v(t)$) and steering angle ($\varphi(t)$) values, so that any parking path whose the corresponding steering angle ($\varphi(t)$) is not admissible can be rejected.

3 The minimum value of ($d$) – which is the distance the car needs to be stopped after the controller has counted a distance ($L$) from the reference point ($R$), and has taken a decision to stop the car – is assumed to be = 2 m.

4 The minimum value of the start shift ($ss$) distance – so that there is an enough safety distance between the car and the parked ones or the external edge of the parking lane – is assumed for our study case to be = 1 m.

- Performance measure:

1 The parking path trajectory generated should achieve minimum path length which is equivalent to the minimal integral value:

$$\int_{x=0}^{x=X_F} \left(\sqrt{(dx)^2 + (dy)^2}\right) = \int_{x=0}^{x=X_F} \left(\sqrt{1 + \left(\frac{dy}{dx}\right)^2}\right) dx \tag{14}$$

2 The car is required to park in a parking lot with the minimum suitable dimensions [length ($L$) and width ($W$)].

The search is performed with a high resolution (0.1 m = 10 cm), with a reasonable safety margin to avoid collisions (threshold distance = 0.5 m), that any parking path far from the lot boundaries with a distance less than this threshold is discarded.

The search results that the best path with minimum length is:

For $d = 2.000000$ m, $ss = 1.600000$ m, $L = 7.000000$ m

$W = 2.600000$ m : $Path\_Length = 9.884070$ m.

$\therefore Y_F = 1.6 + 2 = 3.6$ m, $\therefore X_F = 7 + 2 = 9$ m

The algorithm of the search programme is shown below.

**Algorithm 1** The algorithm of the best path search program

*For $L = L_c$ to $2L_c$ with step 0.1*
*For $s$ =0 to $w_c$ with step 0.1*
*For $d$ = 2 to 3 with step 0.1*
*For $ss$ = 1 to 2 with step 0.1*
*Generate the car path trajectory.*
*If (the car boundaries are not far from the lot boundaries with a distance > threshold distance)*
*Reject the path.*
*End*
*[$v$, $\varphi$] = Inverse_Kinematics($x$, $y$, $\theta$)*
*If (any value of $\varphi > \varphi_{max}$ or any value of $\varphi < \varphi_{min}$)*
*Reject the path.*
*End*
*Path_Length* = (9).
*If (this the first admissible path or Path_Length<Min_Path_Length and $L$ < Min_L)*
*Min_Path_Length = Path_Length.*
*Min_L = L.*
*Store ($L$, $s$, $d$, $ss$) as the parameters of the path with the best length and minimum lot length.*
*End*
*Print the path parameters and Path_Length.*
*End*
*End*
*End*
*Print the parameters ($L$, $s$, $d$, $ss$) of the path with the best length and minimum lot length.*

The length of the straight line (shortest path) connecting the start and final positions $= \sqrt{3.6^2 + 9^2} = 9.69$ which is very close to the length of the generated path.

Thus, the equation of the generated path trajectory is:

$$\therefore y(x) = \begin{cases} \left(\frac{7.2}{81}\right)x^2 \Rightarrow 0 \le x \le 4.5 \\ \left(\frac{-7.2}{81}\right)x^2 + \left(\frac{14.4}{9}\right)x - 3.6 \Rightarrow 4.5 < x \le 9 \end{cases} \tag{15}$$

By operating on the generated path trajectory for our simulation, beside choosing the maximum velocity of car in the backward direction to be ($v_{max}$ = 3 m/s), the inverse kinematics is performed to get ($v$, $\varphi$) for the required path ($x$, $y$) while $T_f$ is decreased every iteration by 0.1 second to search for the lower limit of $T_f$ at which the velocity of car ($v$) exceeds the maximum value chosen ($v_{max}$). After simulation for different values of $T_f$; the search results that: $Tf\_\min = 3.9$ sec.

We can easily derive from the dynamic model that the maximum steering angle ($\varphi_{max}$ = 40°) corresponds to maximum input voltage to the steering motor ($u_2\_max$ = 5 V), while the maximum value for the voltage signal input to the driving motor is taken for our car-like robot as ($u_1\_max$ = 10 V). It should be noted that $u_1$ can be positive or negative corresponding to forward or backward motions of the car, respectively. The positive values of u1 during the backward motion for car's auto parking provide braking action for the car's motion. Thus, the boundary values of $u_1$: $-10 \text{ V} \le u_1 \le 10$ V.

## 4 Swarm-based fuzzy controller (SBFC)

### 4.1 Control system overview

The parking path trajectory is represented by $R_x$ and $R_y$ which are fed into the control system as the desired set point values for the planar coordinates of the rear axle centre point ($x$, $y$). By assuming that $R_x(t)$ is linear with time, where $R_x(0) = 0$ and $R_x(T_F) = X_F = 9$; then

$$\therefore R_x(t) = \left(\frac{9}{T_F}\right)t \tag{16}$$

$$\therefore R_y(t) = \begin{cases} \left(\dfrac{7.2}{T_F^2}\right)t^2 \Rightarrow t \in \left[0, \dfrac{T_F}{2}\right] \\ \left(\dfrac{-7.2}{T_F^2}\right)t^2 + \left(\dfrac{14.4}{T_F}\right) - 3.6 \Rightarrow t \in \left[\dfrac{T_F}{2}, T_F\right] \end{cases} \tag{17}$$

An overview of the car's dynamic model control system is shown in Figure 5. The output controls from SBFC operating on the derived dynamic model are the voltage signals ($u_1$) and ($u_2$) input to the driving and steering motors, respectively, after being passed with both a limiter and a low pass filter.

### 4.2 Specifications of fuzzy control

The real input signals to the fuzzy controller are:

$$e_r, e_\theta, \dot{e}_r, \dot{e}_\theta$$

defined by the equations:

$$\begin{aligned} e_r &= f\left(R_x, R_y, x, y\right) \\ &= \left(\operatorname{sgn}\left(R_x - x\right)\right)\sqrt{\left(R_x - x\right)^2 + \left(R_y - y\right)^2} \end{aligned} \tag{18}$$

$$e_\theta = g\left(R_x, R_y, x, y\right) = \tan^{-1}\left(\frac{dR_y}{dR_x}\right) - \tan^{-1}\left(\frac{dy}{dx}\right) \tag{19}$$

$$\dot{e}_r = \frac{d}{dt}\left(\left|e_r\right|\right), \dot{e}_\theta = \frac{d}{dt}\left(e_\theta\right) \tag{20}$$

The linguistic values defined on the input ($e_r$) are:

- Negative (N), zero (Z), low (L), medium (M), and high (H)

The linguistic values defined on the input ($e_\theta$) are:

- High negative (HN), medium negative (MN), low negative (LN), zero (Z), low positive (LP), medium positive (MP), and high positive (HP)

The linguistic values defined on the inputs ($\dot{e}_r$ and $\dot{e}_\theta$) are:

- Negative (N), zero (Z), and positive (P)

The linguistic values defined on the output ($u_1$) are:

- High (H), medium (M), low (L), zero (Z), and positive (P)

The linguistic values defined on the output ($u_2$) are:

- High negative (HN), medium negative (MN), low negative (LN), zero (Z), low positive (LP), medium positive (MP), and high positive (HP)

The membership functions of the linguistic values defined on the domains of the inputs ($e_r, e_\theta, \dot{e}_r, \dot{e}_\theta$) and the outputs ($u_1$ and $u_2$) are well distributed as shown in Figures 6 and 7, respectively.

The mathematical representation of the membership functions, e.g., of the input $e_r$:

$$\mu_N\left(e_r\right) = \begin{cases} 1 \Rightarrow e_r \le -L_r \\ -\dfrac{e_r}{L_r} \Rightarrow -L_r < e_r \le 0 \\ 0 \Rightarrow e_r > 0 \end{cases} \tag{21}$$

$$\mu_Z\left(e_r\right) = \begin{cases} 0 \Rightarrow e_r \le -L_r \\ 1 + \dfrac{e_r}{L_r} \Rightarrow -L_r < e_r \le 0 \\ 1 - \dfrac{e_r}{L_r} \Rightarrow 0 < e_r < L_r \\ 0 \Rightarrow e_r \ge L_r \end{cases} \tag{22}$$

$$\mu_L\left(e_r\right) = \begin{cases} 0 \Rightarrow e_r \le 0 \\ \dfrac{e_r}{L_r} \Rightarrow 0 < e_r \le L_r \\ 2 - \dfrac{e_r}{L_r} \Rightarrow L_r < e_r < 2L_r \\ 0 \Rightarrow e_r \ge 2L_r \end{cases} \tag{23}$$

$$\mu_M\left(e_r\right) = \begin{cases} 0 \Rightarrow e_r \le L_r \\ \dfrac{e_r}{L_r} - 1 \Rightarrow L_r < e_r \le 2L_r \\ 3 - \dfrac{e_r}{L_r} \Rightarrow 2L_r < e_r < 3L_r \\ 0 \Rightarrow e_r \ge 3L_r \end{cases} \tag{24}$$

$$\mu_H\left(e_r\right) = \begin{cases} 0 \Rightarrow e_r \le 2L_r \\ \dfrac{e_r}{L_r} - 2 \Rightarrow 2L_r < e_r < 3L_r \\ 1 \Rightarrow e_r \ge 3L_r \end{cases} \tag{25}$$

**Figure 5** The control system with SBFC operating on the car dynamic model

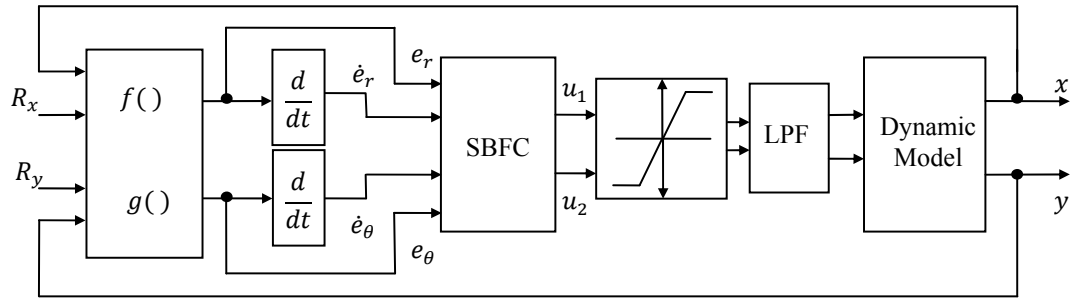



**Figure 6** The membership functions defining the linguistic values for the inputs of the fuzzy controller

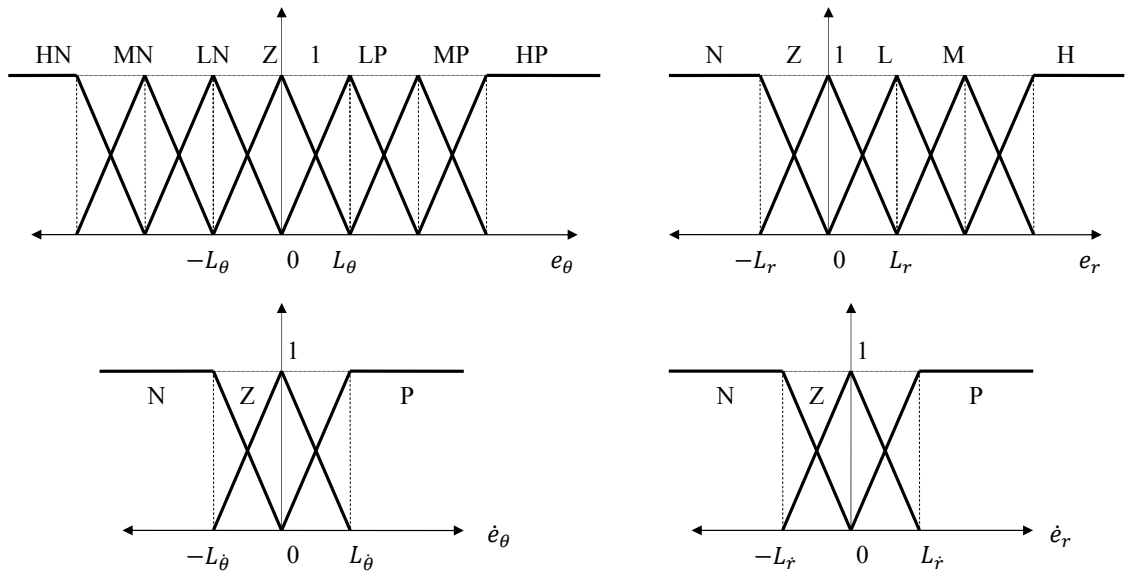
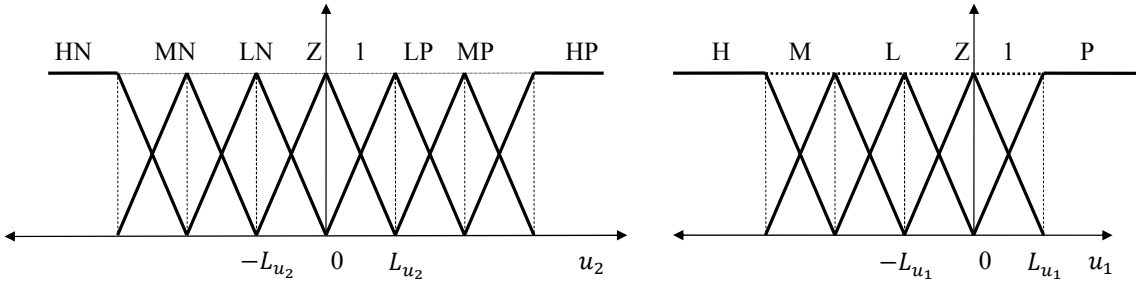



**Figure 7** The membership functions defining the linguistic values for the outputs of the fuzzy controller




The fuzzifier used is the triangular fuzzifier with its base lengths $2F_r$, $2F_\theta$, $2F_{\dot r}$, and $2F_{\dot\theta}$ for the inputs: $e_r$, $e_\theta$, $\dot e_r$, and $\dot e_\theta$, respectively. Triangular fuzzifier – as detailed in Wang (1997) – has a superior advantage that it suppresses input noise and provides simple computations. Mathematical representation of the triangular fuzzifier e.g. for the input $e_r = e_r^*$, where $e_r^*$ is the real input value to the controller:

$$\mu_{A_r'}(e_r) = \begin{cases} 1 - \dfrac{|e_r - e_r^*|}{F_r} \Rightarrow |e_r - e_r^*| < F_r \\ 0 \Rightarrow |e_r - e_r^*| \geq F_r \end{cases} \quad (26)$$

As the magnitude of the voltage signal input to the driving motor (| $u_1$ |) is proportional to the radial distance between the actual position of the car and the desired one (| $e_r$ |), i.e., | $u_1$ | $\propto$ | $e_r$ |; then the rule base relating the states of the output $u_1$ with all the possible configurations of the inputs $e_r$ and $\dot e_r$ can be investigated as shown in Table 2, taking in consideration that the positive sign of $e_r$ corresponds to a need for backward motion (a negative sign of $u_1$) and vice versa i.e., $u_1 \propto -e_r$ with the weight of proportionality increases as $\dot e_r$ increases and decreases as $\dot e_r$ decreases.

**Table 2** Rule base for the output $u_1$

| | | $e_r$ | | | | |
|---|---|---|---|---|---|---|
| | | *N* | *Z* | *L* | *M* | *H* |
| $\dot{e}_r$ | P | P | L | M | H | H |
| | Z | P | Z | L | M | H |
| | N | Z | Z | Z | L | M |

Similarly, the magnitude of the voltage signal input to the steering motor (| $u_2$ |) is proportional to the difference between the actual orientation angle of the car and the desired one (| $e_\theta$ |) but with a negative sign due to the inverse relation between $\varphi$ and $\theta$ at the backward motion of the car where $\theta$ decreases as $\varphi$ increases and vice versa, i.e., $u_2 \propto -e_\theta$ with the weight of proportionality is proportional to $\dot{e}_\theta$. Thus, the rule base relating the states of the output $u_2$ with all the possible configurations of the inputs $e_\theta$ and $\dot{e}_\theta$ can be investigated as shown in Table 3.

**Table 3** Rule base for the output $u_2$

| | | $e_\theta$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | *HN* | *MZ* | *LN* | *Z* | *LP* | *MP* | *HP* |
| $\dot{e}_\theta$ | P | MP | LP | Z | N | MN | HN | HN |
| | Z | HP | MP | LP | Z | LN | MN | HN |
| | N | HP | HP | MP | LP | Z | LN | MN |

There are 15 rules (5 × 3) representing the output $u_1$ and 21 rules (7 × 3) representing the output $u_2$ as shown in Tables 2 and 3, respectively.

Since $u_1$ may have positive values to provide braking action during the car's backward motion then there is another situation – where the car tends to move forward – that should be considered in the design of the rule base representing the output $u_2$. Thus, a new rule is added (no. 22) to state that *if* $e_r$ is negative (N) (the car tends to move forward) *then* $u_2$ is zero (Z) so that the car tends only to overcome the distance error without being deviated from the required position.

For the rules specified in table 2, the general form for the rule no. $k$ where $k \in \{1, 2, 3, \ldots, 15\}$:

if $e_r$ is $A_r^k$ and $\dot{e}_r$ is $A_{\dot{r}}^k$ then $u_1$ is $B_1^k$

where $A_r^k \in \{N, Z, L, M, H\}$, $A_{\dot{r}}^k \in \{N, Z, P\}$,

and $B_1^k \in \{H, M, L, Z, P\}$

For the rules specified in Table 3, the general form for the rule no. $i$ where $i \in \{1, 2, 3, \ldots, 21\}$:

if $e_\theta$ is $A_\theta^k$ and $\dot{e}_\theta$ is $A_{\dot{\theta}}^k$ then $u_2$ is $B_2^k$

where $A_\theta^k \in \{HN, MN, LN, Z, LP, MP, HP\}$,

$A_{\dot{\theta}}^k \in \{N, Z, P\}$, and $B_2^k \in \{HN, MN, LN, Z, LP, MP, HP\}$

The specifications of the inference engine used are:

1 individual-rule-based inference

2 Mamdani's product implication

3 *min* for all the *t*-norm operators and *max* for all the *s*-norm operators.

Thus, the mathematical representation of the inference engine for $u_1$ for e.g.:

$$\mu_{B_1'}(u_1) = \max_{k=1}^{15}\left[\min_{j\in\{r,\dot{r}\}}\left(\sup_{\forall e_j}\left[\min\left(\mu_{A_j'}(e_j), \mu_{A_j^k}(e_j)\right)\right]\right) \times \mu_{B_1^k}(u_1)\right] \quad (27)$$

The defuzzifier used for our controller is the centre average defuzzifier. Because the fuzzy set $B'$ generated from the inference engine is the union of a number of fuzzy sets defined in the domain of the controller outputs, then a good approximation of the centre of gravity defuzzifier (which specifies $u_i^*$ as the centre of the area covered by the membership function of $B_i'$) is the weighted average of the centres of these $J$ fuzzy sets (where $J = 5$ for $u_1$ and $J = 7$ for $u_2$) with the weights equal the heights of the corresponding fuzzy sets as discussed in Wang (1997). Specifically, let $u_i^\ell$ be the centre of the $\ell^{\text{th}}$ fuzzy set and $w_\ell$ be its height; then the centre average defuzzifier determines $u_i^*$ as:

$$u_i^* = \frac{\sum_{\ell=1}^{J} u_i^\ell w_\ell}{\sum_{\ell=1}^{J} w_\ell} \quad (28)$$

### *4.3 Swarm search algorithm*

The search algorithm used to optimise the controller performance is the artificial fish swarm algorithm (AFSA) discussed in Zhang et al. (2006a, 2006b) with some improvements to speed up the search process.

AFSA is a stochastic and effective global optimisation algorithm which mainly simulates fish schooling the behaviour of prey, swarm, and follow. In comparison with genetic algorithm, AFSA possess similar attractive features like independence from gradient information of the objective function and the ability to solve complex non-linear high dimensional problems (which is the case for the parameters tuning problem of our control system). Furthermore, they can achieve faster convergence speed (especially with the improvements added in this paper) and require few parameters to be adjusted. Whereas the AFSA does not possess the crossover and mutation processes used in GA, so it could be performed more easily. AFSA is also an optimiser based on population. The system is initialised firstly in a set of randomly generated potential solution, and then performs the search for the optimum one iteratively.

Information about good solutions spreads through the swarm, and thus the AF tend to move to good areas in the search space. The general principles for the AFSA are stated as follows. Suppose that the searching space is $D$-dimensional and $N$ fish form the colony. The $i^{\text{th}}$ AF

represents a $D$-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ ($i = 1, 2, \ldots, N$, here $N$ is the swarm size)that is used to evaluate the quality of the AF. The state of each AF is a potential result. Therefore we could calculate the AF's fitness by putting its state into a designated objective function with a required global minimum. When the objective function (AF's fitness) is lower, the corresponding $X_i$ is better. Given a fitness function $f(X)$ where $X$ is a vector of $D$ real-valued random variables; the AFSA initialises firstly a swarm of fish, each fish is randomly positioned in the $D$-dimensional real number space, and is a candidate solution to the fitness function; and then performs the search for the optimum one iteratively.

During the search process, each AF successively adjusts its state toward the good areas according to the behaviour of prey, swarm and follow, and finally, gains the global optimum. The behaviour description of the AF is as follows:

1 *Prey behaviour*: Suppose that an AF's current state is $X_i$. We randomly select a new state $X_j$ in its visual field according to the following equation:

$$X_j = X_i + Rand() \times Visual \tag{29}$$

where *Rand()* is a random function in the range [0,1], and *Visual* represents the visual distance of AF. If $f(X_j) < f(X_i)$; that satisfies the onward condition, then AF's state at the next iteration is calculated by following equation:

$$X_i^{(t+1)} = X_i^{(t)} + Rand() \times Step \times \frac{\left(x_j - x_i^{(t)}\right)}{\left\|x_j - x_i^{(t)}\right\|} \tag{30}$$

where $X_i^{(t+1)}$ represents the AF's next state with the current state $X_i^{(t)}$; $d_{i,j} = \| X_j - X_i \|$ represents the Euclidean relative distance between $X_i$ and $X_j$; and *Step* represents the distance that AF can move for each step. In the opposite case, if $f(X_j) \geq f(X_i)$, then a new state $X_j$ is selected according to equation (21), and judge whether it satisfy the onward condition or not; after trying several times, if still did not find to satisfy the onward condition, then random move one step according to the following equation:

$$X_i^{(t+1)} = X_i^{(t)} + Rand() \times Step \tag{31}$$

2 *Swarm behaviour*: An AF with the current state $X_i$ seeks the companion's number in its neighbourhood where satisfy $d_{i,j} < Visual$; and calculate their centre position $X_{centre}$. If $n_f.f(X_{centre}) < \delta.f(X_i)$; that satisfies the onward condition, then the AF's state at the next iteration is calculated as follows:

$$X_i^{(t+1)} = X_i^{(t)} + Rand() \times Step \times \frac{\left(X_{centre} - X_i^{(t)}\right)}{\left\|X_{centre} - X_i^{(t)}\right\|} \tag{32}$$

where $n_f$ represents the companion's number in the AF current neighbourhood; and $\delta$ is a positive constant of greater than 1, called the crowded degree factor. Otherwise, the AF carries out the prey behaviour.

3 *Follow behaviour*: An AF with the current state $X_i$ seeks the companion $X_{\min}$ that the optimisation function value is minimum among all ones in its current neighbourhood ($d_{i,j} < Visual$). If $n_f.f(X_{min}) < \delta.f(X_i)$; that satisfies the onward condition, then the AF's state at the next iteration is calculated as follows:

$$X_i^{(t+1)} = X_i^{(t)} + Rand() \times Step \times \frac{\left(X_{\min} - X_i^{(t)}\right)}{\left\|X_{\min} - X_i^{(t)}\right\|} \tag{33}$$

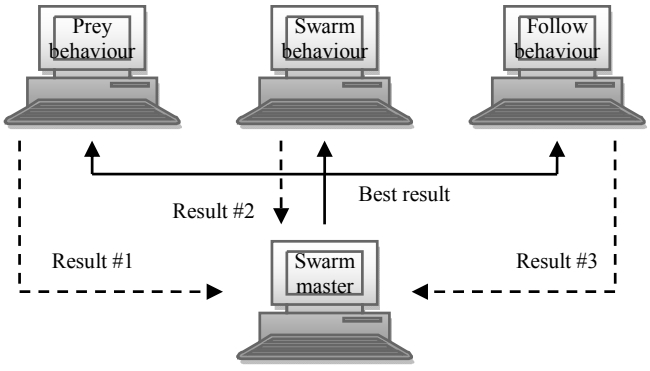
Otherwise, the AF carries out the prey behaviour.

Finally, the most AF can gather in the global optimum surroundings generally. The search is stopped when the average companion's number in the AF current neighbourhood reaches a reasonable value compared to the swarm size (total no. of AFs).

As the swarm search aims to find the values of the fuzzy controller parameters and the parking time – as detailed in the next section – that best controls our car's complex non-linear dynamic model described by the differential equations (5), (7), and (9) to track the desired parking path; it is so important to introduce a new improvement for the AFSA to enhance both the timing of search and its efficiency.

Our improvement is to perform the three behaviours of AFSA for every AF concurrently on three machines rather than performing them sequentially on one machine, where there is a fourth master machine that receives the results got by the three clients represented in the new positions of the AFs after every iteration, beside the smallest AF's fitness obtained, and the average AF's number of companions. The master machine acts as a judger that determines the resulted AFs with the minimum smallest fitness, then transmits it to the three clients to be the initial state of the AFs at the next iteration, while for the first iteration, the master transmits a randomly generated AFs as the initial state for the swarm search. Figure 8 shows a diagram for the contributed AFSA used to design our fuzzy controller.

**Figure 8** Schematic diagram for the modified AFSA




### *4.4 Swarm-based fuzzy controller*

The domain of search for the AFSA is composed of 11 parameters, where the fuzzy controller is encoded into a chromosome of ten parameters ($L_r$, $L_\theta$, $L_{\dot{r}}$, $L_\theta$, $L_{u_1}$, $L_{u_2}$,

$F_r$, $F_\theta$, $F_{\dot{r}}$, and $F_{\dot{\theta}}$), and the 11th parameter is $T_F$. The objective fitness function used for the evaluation of AF is the *root integral square error* (RISE) that is required to be minimised.

$$\text{RISE} = \left[\sqrt{\int_{t=0}^{T_F}\left(\left(e_x(t)\right)^2+\left(e_y(t)\right)^2\right)dt}\right] + 1{,}000\times\left(\left(e_x(t)\right)^2+\left(e_y(t)\right)^2\right) \quad (34)$$

It is clear that there is a specific high weight for the error at the final position ($T_F$) in equation (34) to ensure the great importance for the controller to reach the final point correctly.

The specifications of the swarm search are that:

- The swarm size = 10 AFs.
- The maximum no. of trials for prey behaviour = 10.
- The maximum no. of iterations = 200.
- The crowded degree factor ($\delta$) = 3.
- The 11-dimensional search domain: $\{L_r, L_\theta, L_{\dot{r}}, L_{\dot{\theta}}, u_1\_\max/L_{u_1}, u_2\_\max/L_{u_2}, F_r, F_\theta, F_{\dot{r}}$, and $F_{\dot{\theta}}\}$
- The search step = {0.1, 0.4, 0.2, 0.8, 1, 1, 0.1, 0.4, 0.2, 0.8, 6}.
- The upper bound = {1, 4, 4, 8, 6, 6, 1, 4, 3, 6, 60}.
- The lower bound = {0.05, 0.5, 0.05, 0.5, 0.001, 0.001, 0.05, 0.5, 0.05, 0.5, 4}.

## 5 Simulation results

The best control performance is obtained with RISE = 4.3707 at the chromosome parameters: {0.0866, 1.4094, 1.8897, 4.8088, 1.7181, 0.9722, 0.05, 1.7824, 0.5068, 3.5292, 38.7133}. The plots of ($R_x(t)$ and $x(t)$), ($R_y(t)$ and $y(t)$), ($R_\theta(t)$ and $\theta(t)$), and ($R_y(R_x)$ and $y(x)$) are shown in Figures 9, 10, 11, and 12, respectively, where $R_\theta(t)$ and $\theta(t)$ are obtained from equation (10). As shown in figures, the generated path trajectory is too close to the required one.

By applying disturbance forces ($F_{fr}$ and $F_W$) on the car's dynamic model (the plant of our control system) as:

$$F_{fr}(t) = B_{wheel}\times v(t)$$

$$F_W(t) = \begin{cases} 0 \Rightarrow t \le T_F/4 \\ C \Rightarrow T_F/4 < t < 3T_F/4 \\ 0 \Rightarrow t \ge 3T_F/4 \end{cases}$$

where $B_{wheel}$ is the friction between the car wheels and the ground, assumed in our simulation = 0.2 N/m/s, while $C$ is a constant value reached by the wind force at the intended time interval assumed = 0.1 N.

Figure 13 shows the plots of ($R_y(R_x)$ and $y(x)$) generated from our controlled system after applying disturbance forces, with *RISE* = 20.0248 that looks large due to a final position error ≈ 0.124 m = 12.4 cm which is an acceptable value according to the safety margins considered for the reference point ($R$) at the start of the parking lot.

**Figure 9** The plots of $R_x(t)$ and $x(t)$

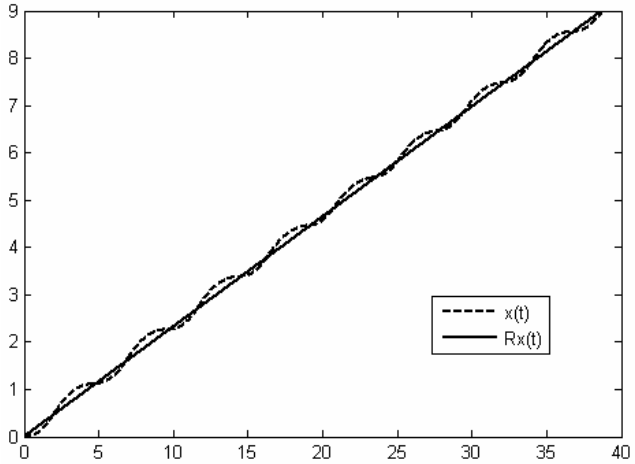



**Figure 10** The plots of $R_y(t)$ and $y(t)$

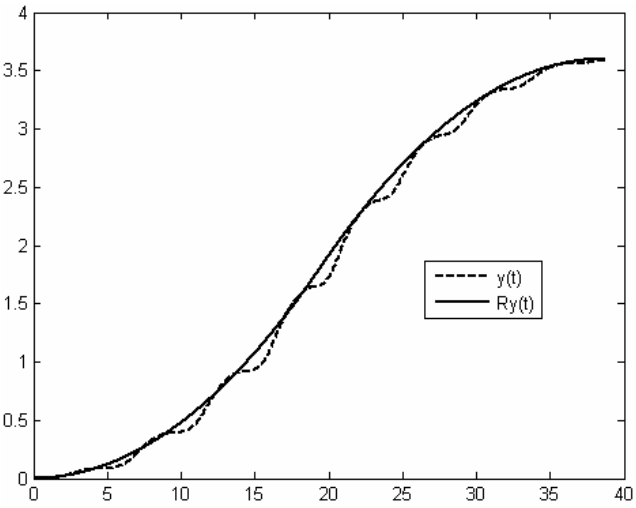



**Figure 11** The plots of $R_\theta(t)$ and $\theta(t)$

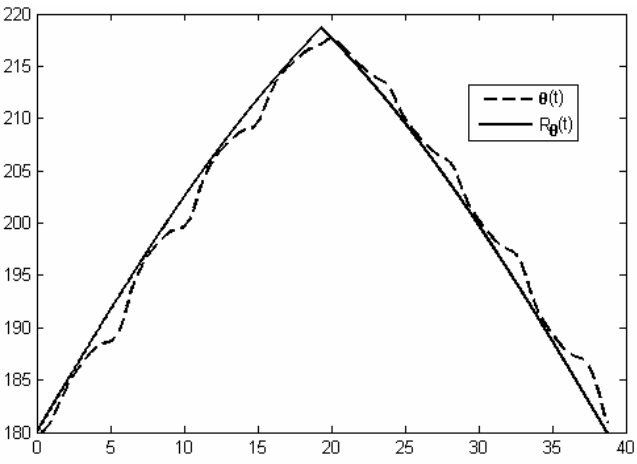


**Figure 12** The plots of $R_y(R_x)$ and $y(x)$

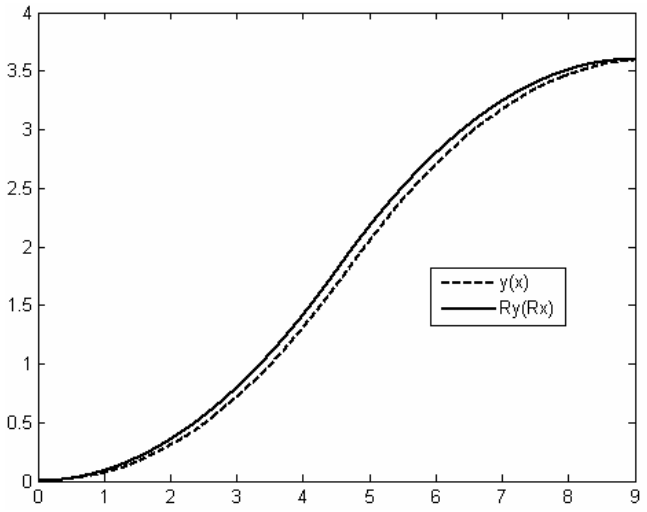
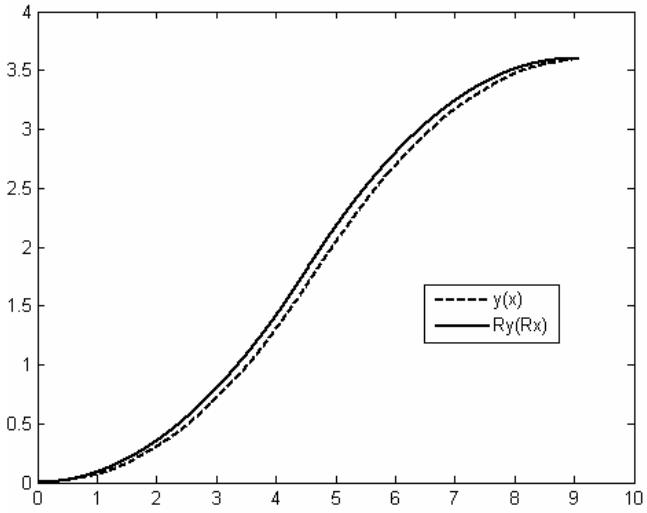



**Figure 13** The plots of $R_y(R_x)$ and $y(x)$ after applying disturbances




## 6 Conclusions and future work

SBFC has proven a splendid efficiency to solve the path tracking problem related to the car's column auto parking while applying it on a complex model with non-linear differential equations and realistic delays, i.e., the car's dynamic model; this result strongly encourages us to apply it practically on a realistic mechanical car model under more realistic disturbances.

## References

Abdelmoula, C., Chaari, F. and Masmoudi, M. (2009) 'Implementation of applications on a newly designed robot prototype: 'autonomous navigation and parallel parking'', Paper presented at the *2009 International Conference on Signals, Circuits and Systems*, pp.1–5.

Dao, M.Q. and Liu, K-Z. (2006) 'Development of A practical automatic parking technology for automobiles', Paper presented at the *45th IEEE Conference on Decision & Control*, 13–15 December, Manchester Grand Hyatt Hotel, San Diego, CA, USA, pp.1727–1732.

Hsieh, M.F. and Özguner, Ü. (2008) 'A parking algorithm for an autonomous vehicle', Paper presented at the *2008 IEEE Intelligent Vehicles Symposium*, 4–6 June, Eindhoven University of Technology, Eindhoven, The Netherlands, pp.1155–1160.

Imae, J., Yoshimura, K., Zhai, G. and Kobayashi, T. (2009) 'Real-time optimization for parallel-parking control of four-wheeled vehicles', *International Journal of Modelling, Identification, and Control*, Vol. 6, No. 3 pp.255–262.

Inoue, T., Dao, M.Q. and Liu, K. (2004) 'Development of an auto-parking system with physical limitation', Paper presented at the *SICE 2004 Annual Conference*, Hokkaido Institute of Technology, Japan, pp.1015–1020.

Jallouli, M., Rekik, C., Chtourou, M. and Derbel, N. (2010) 'Optimised fuzzy logic controller for a mobile robot navigation', Paper presented at the *IJMIC 2010*, Vol. 9, No. 4, pp.400–408.

Lee, J-Y., Kim, M-S. and Lee, J-J. (2006) 'Design of fuzzy controller for car parking problem using evolutionary multi-objective optimization approach', Paper presented at the *IEEE ISIE 2006*, 9–12 July, Montréal, Québec, Canada, pp.329–334.

Li, T-H.S., Chen, C-Y. and Lim, K-C. (2010) 'Combination of fuzzy logic control and back propagation neural networks for the autonomous driving control of car-like mobile robot systems', Paper presented at the *SICE Annual Conference 2010*, 18–21 August, The Grand Hotel, Taipei, Taiwan.

Miah, Md.S. and Gueaieb, W. (2007) 'Intelligent parallel parking of a car-like mobile robot using RFID technology', Paper presented at the *ROSE 2007 – IEEE International Workshop on Robotic and Sensors Environments*, 12–13 October, Ottawa, Canada.

Moret, E.N. (2003) 'Dynamic modeling and control of a car-like robot', MSc thesis, Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA.

Paromtchik, I.E. and Laugier, C. (1997) 'Automatic parallel parking and returning to traffic maneuvers', Paper presented at the *IROS 97*, pp.V-21–V-23.

Wang, L-X. (1997) 'A course in fuzzy systems and control', in *Fuzzifiers and Defuzzifiers*, pp.105–112, Prentice Hall, USA.

Zhang, M., Shao, C., Li, F., Gan, Y. and Sun, J. (2006a) 'Evolving neural network classifiers and feature subset using artificial fish swarm', Paper presented at the *2006 IEEE International Conference on Mechatronics and Automation*, 25–28 June, Luoyang, China, pp.1598–1602.

Zhang, M., Shao, C., Li, M. and Sun, J. (2006b) 'Mining classification rule with artificial fish swarm', Paper presented at the *6th World Congress on Intelligent Control and Automation*, 21–23 June, Dalian, China, pp.5877–5881.