Robotica (2014) volume 32, pp. 935–952. © Cambridge University Press 2013. The online version of this article is published within an Open Access environment subject to the conditions of the Creative Commons Attribution licence http://creativecommons.org/licenses/by/3.0/
doi:10.1017/S0263574713000982

A solution to the motion planning and control problem of a car-like robot via a single-layer perceptron

Avinesh Prasad, Bibhya Sharma* and Jito Vanualailai

School of Computing, Information & Mathematical Sciences, University of the South Pacific, Suva, Fiji

(Accepted October 25, 2013. First published online: December 13, 2013)

SUMMARY
This paper tackles the problem of motion planning and control of a car-like robot in an obstacle-ridden workspace. A kinematic model of the vehicle, governed by a homogeneous system of first-order differential equations, is used. A solution to the multi-tasking problem of target convergence, obstacle avoidance, and posture control is then proposed. The approach of solving the problem is two-fold. Firstly, a novel velocity algorithm is proposed to drive the car-like robot from its initial position to the target position. Secondly, a single layer artificial neural network is trained to avoid disc-shaped obstacles and provide corresponding weights, which are then used to develop a function for the steering angles. Thus, our method does not need *a priori* knowledge of the environment except for the goal position. With the help of the Direct Method of Lyapunov, it is shown that the proposed forms of the velocity and steering angle ensure point stability. For posture stability, we model the two parallel boundaries of a row-structured parking bay as continua of disk-shaped obstacles. Thus, our method is extendable to ensuring posture stability, which gives the desired final orientation. Computer simulations of the generated path are presented to illustrate the effectiveness of the method.

KEYWORDS: Car-like robot; Neural network; Single-layer perceptron; Motion planning; Posture control.

1. Introduction

Motion planning and control of autonomous nonholonomic robot has been an active area of research over the past two decades. Researchers over the years have used various methods such as artificial potential field (APF) method, graph search technique, and neural networks to tackle the multi-tasking problem of the autonomous nonholonomic robots.

The work of Khatib in ref. [1] is considered as the landmark result in motion planning and control of autonomous robot via APF method. Since then the potential method has created a huge interest among researchers and numerous algorithms based on this method have appeared in the literature.[2–6] Researchers have found APF useful due to easier analytic representation of system singularities and inequalities, better processing speed, its simplicity, and elegance.[7] However, this method inherits the problem of *traps* or local minima and work on the nonholonomic robots with continuous control laws have only managed to obtain a Lyapunov stable system,[7,8] invariably due to Brockett's Theorem.[9]

In the graph search techniques, collision-free trajectories are established by searching for graphs or maps formed out of straight lines via vertices of obstacles or patches of free space decomposed into geometrical primitives.[10–13] While the algorithms in this technique are elegant, they are computationally intensive and can suffer from the problem of *too close* or *too far*.[14] A commonly used algorithm known as the A* algorithm is often employed in the graph search technique for the avoidance of concave-shaped obstacles.

* Corresponding author. E-mail: sharma_b@usp.ac.fj

In other developments that are to some extent conceptually similar to ours, the form of the velocity or the steering angles of moving agents is modeled only after observing a certain process. For instance, in the work of Fajen *et al.*,[15] the function for the steering angles of a moving point mass is developed only after observing human walking. In our case, the steering angles are determined only after we trained a neural network to avoid obstacles and output corresponding values of the weights, which are used in the formula for the steering angle. Such an approach does not need *a priori* knowledge of the environment; hence, it is an on-line based control strategy. In the paper by Fajen *et al.*,[15] the history behind the development of such strategies is provided.

In recent years, the artificial neural network (ANN) has become a powerful tool for solving the motion planning and control problem of mobile robots. In literature, task planning, autonomous robot path planning, and position control can be easily formulated and solved through ANNs. In 1994, Rivals *et al.*[16] used the neural network technique to control the motion of a four-wheel-drive vehicle. The path and velocity were modeled via a supervised network, where the initial data were obtained by manually driving the vehicle. Izumi *et al.* in ref. [17] used a feedback error learning technique to control a nonholonomic mobile robot. The authors proposed a neuro interface-like control system using the concept of internal model control. However, there was no mention of targets or obstacles in the workspace. Janglova in ref. [18] dealt with a path planning and intelligent control of an autonomous robot in partially structured environment using two neural networks; one to determine the free space and the other to find a safe direction. Although the motion was controlled in an obstacle-ridden workspace, the authors have not considered the target convergence or the velocity of the robots. Naoui *et al.* in ref. [19] used nonconventional approaches as ANN in the internal model control of mobile car-like robot. A constant velocity was applied to the mobile robot and the steering angle was controlled using the ANN approach. Again, there was no mention of targets or obstacles in the workspace.

While neural network-based approach of controlling the motion of nonholonomic robots is commonly found in literature, the following aspects (to the author's knowledge) are lacking:

(a) a rigorous stability analysis of a system that depends on an ANN to provide the necessary parameters;
(b) the inclusion of a target when designing the velocity algorithm;
(c) an analysis of posture control of the nonholonomic robots; and
(d) an explicit formula of the steering obtained via an ANN.

It is only recent that the authors in ref. [20] deployed a neural network approach to control the motion of a point mass robot in an obstacle-ridden workspace and proved asymptotic stability of the system. In the paper, the authors designed a velocity algorithm that could drive the robot from its initial position to the target position and used a multi-layer perceptron (MLP) to control the direction of a point mass robot in the obstacle region. However, it was still not possible to obtain an explicit formula of the controllers.

In this paper, we extend the work carried out in ref. [20] by solving the multi-tasking problem of target convergence, obstacle avoidance, and attaining a desired final orientation of a car-like robot. While the velocity algorithm is adopted from ref. [20], we will model the steering angle of the car-like robot in the obstacle-ridden workspace using a single-layer perceptron (SLP) and derive an explicit formula of the steering angle in terms of specific inputs. In addition, this paper achieves a predetermined final orientation of the vehicle. This is done by constructing a virtual parking bay that surrounds the target, and boundary lines of the bay is avoided by treating them as artificial obstacles. Furthermore, the control laws proposed in this paper are continuous everywhere along the trajectory of the system and the resultant path is safest, smoothest, and the shortest one. This paper provides four major contributions:

(1) *Obstacle avoidance scheme*: We continuously consider all the obstacles in the workspace. This will give the motion planner a global view of the workspace and thus picking out the shortest and safest path among the obstacles.
(2) *Single-layer perceptron (SLP)*: While the velocity algorithm adopted from ref. [20] is sufficient to drive the robot to its target, the inclusion of obstacles garners the need to control the steering angle.
(3) *Training data*: The training data for the neural network are obtained using computer simulations where the initial path is traced by the user. The data are not obtained from real-life experiments,

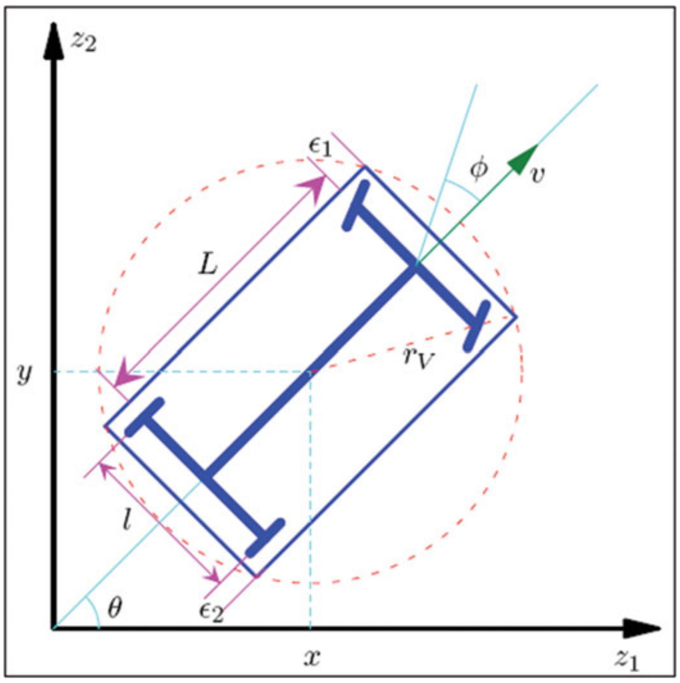

Fig. 1. (Colour online) A rear wheel driven vehicle with front wheel steering and steering angle ϕ (adopted from ref. [7]).

such as physically driving or maneuvering a real robot. Hence, the procedure in this research provides data in a more simplified and controlled way.

(4) *Derive an explicit formula of the steering angle*: The use of a SLP to model the motion of the car-like robot enables us to derive an explicit formula of the steering angle as a function of specific inputs.

The remainder of the chapter is organized as follows. Section 2 considers the definition of the car-like robot and looks at its kinematic model. Section 3 considers the main objective followed by the motion planning and control of the car-like robot in the absence of obstacles in Section 4. The stability issues in the absence of obstacles are also addressed in this section. In Section 5, the motion of the car-like robot is controlled in the presence of fixed circular obstacles of random size and positions. Section 6 considers the problem of parking maneuverability and proposes a feasible solution using the minimum distance technique. Finally, in Section 7, the paper closes with a discussion on its contributions.

2. Vehicle Model

The rear wheel driven car-like vehicle model and the associated terminologies and notations are adopted from ref. [7]. Referring to Fig. 1, (x, y) denotes the center of mass (CoM) of the car, θ gives the car's orientation with respect to the z_1-axis, and ϕ gives the steering wheel's angle with respect to car's longitudinal axis. The configuration of the car is given by $(x, y, \theta, \phi) \in \mathbf{R}^4$, and its position is given as the point $(x, y) \in \mathbf{R}^2$. The role of the *clearance parameters* ε_1 and ε_2 will be highlighted in Section 5.

If L is the distance between the two axles and l the length of each axle, then the kinematic model of the car-like vehicle adopted from ref. [7] is

$$\left.\begin{aligned}
&\dot{x} = v\cos\theta - \tfrac{v}{2}\tan\phi\sin\theta,\\
&\dot{y} = v\sin\theta + \tfrac{v}{2}\tan\phi\cos\theta,\\
&\dot{\theta} = \tfrac{v}{L}\tan\phi,\\
&x_0 := x(0), \quad y_0 := y(0), \quad \theta_0 := \theta(0),
\end{aligned}\right\} \tag{1}$$

where v is the translational velocity. Hereafter, we shall use the vector notation $\mathbf{x}(t) = (x, y) \in \mathbf{R}^2$ to describe the position variables in Eq. (1). Moreover, v and ϕ are treated as controllers that will be designed in later sections.

3. Main Objective: Problem Statement

Let A be a car-like mobile robot, as shown in Fig. 1, moving in a constrained workspace *WS* on the $z_1 z_2$ plane. Let $FO_1, FO_2, \ldots, FO_q$ are stationary obstacles randomly placed in *WS*. Assume that both the geometry and the location of A and $FO_1, FO_2, \ldots, FO_q$ is *a priori* known. The problem statement is:

Given any initial position and orientation of A in WS, design the controllers v and ϕ so that A can converge to a goal position and orientation while satisfying all the nonholonomic constraints and avoiding collisions with FO_l for $l = 1, 2, \ldots, q$.

4. Motion Control in the Absence of Obstacles

In our motion planning problem, we have a designated target that is a disk of center (p_1, p_2) and radiusr_T. The target can be described as

$$\mathrm{T} = \left\{(z_1, z_2) \in \mathbf{R}^2 : (z_1 - p_1)^2 + (z_2 - p_2)^2 \le r_T^2\right\}.$$

We want the car-like mobile robot to start from an initial configuration, move towards its target, and converge at the center of the target.

4.1. Velocity algorithm

We first look at the controller v, which should drive A from its initial position to the target position and vanish once the robot reaches the target. Velocity algorithms in literature include mostly constant[18] (either maximum or optimal) velocities that truncate at the goal configuration. However, a sudden switch or truncation of the velocity to force the robot to stop will require infinite accelerations and in turn infinite torques and can also cause physical damage to the robot.[20] Thus, we adopt the velocity algorithm from ref. [20] where the authors developed a more practical algorithm that depend on the initial and final positions of the robot:

$$v(t) = |v_0| \frac{\|\mathbf{x}(t) - \mathbf{x}_e\|}{\|\mathbf{x}(0) - \mathbf{x}_e\|}, \tag{2}$$

where v_0 is the initial velocity of the robot at $t = 0$ and $\mathbf{x}_e = (p_1, p_2) \neq \mathbf{x}(0)$ is an equilibrium point of system (1). Note that v given by Eq. (2) is defined, continuous, and positive over the domain

$$D = \{\mathbf{x}(t) \in \mathbf{R}^2 : \mathbf{x}(0) \neq \mathbf{x}_\mathrm{e}\}.$$

We also note here that as $t \to \infty$, $v(t) \to 0$ since $\mathbf{x}(t) \to \mathbf{x}_e$.

4.2. Design of the steering angle

First we let

$$\xi(t) = \begin{cases} \mathrm{atan2}(p_2 - y(t), p_1 - x(t)) & \text{if } \mathbf{x}(t) \neq \mathbf{x}_e \\ \xi(t-1) & \text{if } \mathbf{x}(t) = \mathbf{x}_e \end{cases}$$

be the angular position of the target with respect to the current position of the car-like robot at time t. Before we design ϕ, we stipulate the following definition of heading:

Definition 1: When $\theta(t) = \xi(t)$, we say that the car-like robot is heading towards its designated target.

We now look at two simple scenarios in the absence of obstacles:

Scenario 1: When initially the car is heading towards its target.

Scenario 2: When initially the car is not heading towards its target.

Scenario 1: When $\theta(t) = \xi(t)$. In this scenario, when the car is heading towards its target, it is important that no turning occurs so that we can get an overall shortest path from the initial to the goal position. Thus, we define the steering angle as

$$\phi = 0. \tag{3}$$

In this case, we come up with the following claim.

Claim 1: *When $\theta(t) = \xi(t)$ and the velocity given by Eq. (2) is applied to the car-like robot, the robot will move straight towards its target with its speed slowing down and will converge to the target position.*

Proof. Note that if $\theta(t) = \xi(t)$, then $\phi(t) = 0$. We first show that under the given condition, the robot will move towards its target. Let $d = \sqrt{(p_1 - x(t))^2 + (p_2 - y(t))^2}$ be the distance between the target and the robot position at any time t. Then

$$\begin{aligned}
\dot{d} &= \frac{-1}{d}\left[(p_1 - x(t))\dot{x}(t) + (p_2 - y(t))\dot{y}(t)\right] \\
&= \frac{-1}{d}\left[(p_1 - x(t))v\cos\theta + (p_2 - y(t))v\sin\theta\right] \\
&= \frac{-v}{d}\left[\frac{(p_1 - x(t))^2}{d} + \frac{(p_2 - y(t))^2}{d}\right] \\
&= \frac{-v}{d^2}\left[(p_1 - x(t))^2 + (p_2 - y(t))^2\right] \\
&= \frac{-v}{d^2}d^2 = -v < 0,
\end{aligned}$$

which means that as the robot moves, d decreases; thus, the robot is moving towards the target. Secondly, to show that the motion of the robot gradually slows down, we note that

$$\dot{v} = \frac{|v_0|\,\dot{d}}{\|\mathbf{x}(0) - \mathbf{x}_e\|} = -\frac{|v_0|\,v}{\|\mathbf{x}(0) - \mathbf{x}_e\|} < 0,$$

which means that v decreases as the robot moves towards the target. Finally, to show that the robot converges to the target position, we see that at the center of the target,

$$v = |v_0|\,\frac{\|\mathbf{x}_e - \mathbf{x}_e\|}{\|\mathbf{x}(0) - \mathbf{x}_e\|} = 0.$$

Simulation 1: Figure 2 shows a simulation with defined initial and final positions of the car-like robot. Given that the car is initially pointing towards the target, with any initial position, the robot moves straight towards its designated target.

Theorem 1: *Let the controllers v and ϕ be as defined in Eqs. (2) and (3), respectively. If the car-like robot governed by system (1) is initially heading towards its target, then $\mathbf{x}_e$ is the only equilibrium point of system (1) and is globally asymptotically stable.*

Proof. We first show that $\mathbf{x}_e$ is the only equilibrium point of system (1). To see that note that when we solve the system

$$(\dot{x}, \dot{y}, \dot{\theta}) = (0, 0, 0),$$

we get only one solution $v = 0$, which implies that $\mathbf{x} = \mathbf{x}_e$. Next, to prove global asymptotic stability, consider the Lyapunov function

$$L(\mathbf{x}) = \tfrac{1}{2}\,\|\mathbf{x}(t) - \mathbf{x}_e\|^2\,,$$

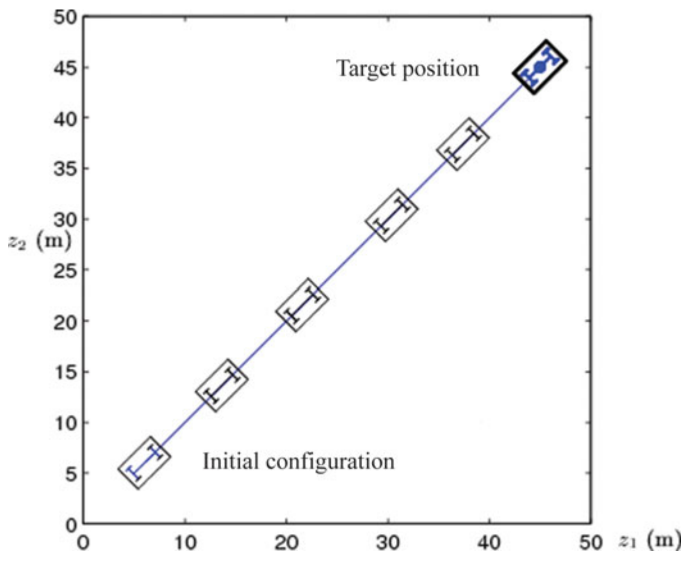

Fig. 2. (Colour online) Trajectory of the car-like robot with initial position (6, 6) and the target placed at (45, 45).

which is defined, continuous, positive, and radially unbounded over the domain $D = \{\mathbf{x}(t) \in \mathbf{R}^2 : \mathbf{x}(0) \neq \mathbf{x}_e\}$. Clearly, $L(\mathbf{x})$ has continuous first partial derivatives in the neighborhood $D$of the equilibrium point $\mathbf{x}_e$ of system (1). Moreover, in the region D, we see that $L(\mathbf{x}_e) = 0$ and $L(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{x}_e$. Now, the time-derivative of $L(\mathbf{x})$ along a trajectory of system (1) is given by

$$L(\mathbf{x}) = -v(t)\,\|\mathbf{x}(t) - \mathbf{x}_e\|\,,$$

where $v(t)$ is governed by Eq. (2). Again, it is clear that in the region D, $\dot{L}(\mathbf{x}_e) = 0$ and $\dot{L}(\mathbf{x}) < 0$ for all $\mathbf{x} \neq \mathbf{x}_e$. Hence, it can be concluded that the point $\mathbf{x}_e$ is a global asymptotic stable equilibrium point of system (1).

Scenario 2: When $\theta(t) \neq \xi(t)$: In a practical sense, $\theta(t)$ may not be the same as $\xi(t)$. To overcome this we propose a formula for calculating $\phi(t)$ in terms of $\xi(t)$ and $\theta(t)$. We also note here that the steering angle of the front wheel is bounded.[7] This leads to the following assumption:

Assumption 1: *The steering angle, $\phi(t)$, satisfies the inequality*

$$-70^\circ < \phi(t) < 70^\circ.$$

Since we want to express $\phi(t)$ in terms of $\xi(t)$ and $\theta(t)$, we need to enact the following rules:

Rule 1: If $\theta(t) < \xi(t)$, then a negative steering angle $\phi(t)$ should be applied, which in turn should increase until $\xi(t)$ and $\phi(t)$ become the same.

Rule 2: If $\theta(t) > \xi(t)$, then a positive steering angle $\phi(t)$ should be applied, which in turn should decrease until $\xi(t)$ and $\theta(t)$become the same.

Taking Assumption 1 and the above rules into account, we propose the following formula for calculating $\phi(t)$:

$$\phi(t) = \tfrac{7}{9}\tan^{-1}\left(\xi(t) - \theta(t)\right). \tag{4}$$

We make the following claim from our proposed formula:

Claim 2: *The proposed formula for $\phi(t)$ satisfies the inequality* $-70^\circ < \phi(t) < 70^\circ$.

Proof. Note that $-90^\circ < \tan^{-1}(\xi(t) - \theta(t)) < 90^\circ$. Multiplying through 7/9, we get $-70^\circ < \phi(t) < 70^\circ$.

Claim 3: *The controller $\phi(t)$ given by Eq. (4) will eventually reduce to Eq. (3).*

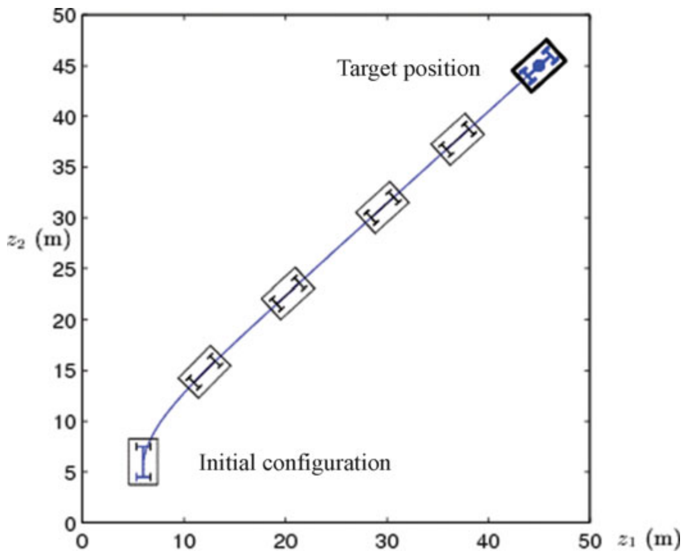

Fig. 3. (Colour online) Trajectory of the car-like robot with initial position (6, 6) and the target placed at (45, 45).

Proof. We need to show that $\phi(t) \to 0$ as $t \to +\infty$, or equivalently, we can show that $0 \in \phi(t)$and $\phi^2(t)$ is a decreasing function. Firstly, note that when $\theta(t) = \xi(t)$, we see that $\phi(t) = 0$, which implies that $0 \in \phi(t)$. Secondly, to show that $\phi^2(t)$ is a deceasing function, we consider two simple cases:

Case 1: When $\theta(t) < \xi(t)$. Then $\theta(t)< 0$ and in light of Rule 1, $\dot{\theta}(t)> 0$, so $\frac{d}{dt}(\phi^2(t)) = 2\phi(t)\dot{\phi}(t) < 0$.

Case 2: When $\theta(t) > \xi(t)$. In this case $\theta(t)> 0$ and in light of Rule 2, $\dot{\theta}(t)< 0$. Thus, $\frac{d}{dt}\left(\phi^2(t)\right) = 2\phi(t)\dot{\phi}(t) < 0$.

The discussion so far leads to the following theorem:

Theorem 2: *The equilibrium point* $\mathbf{x}_e$ *of system (1) is asymptotically stable provided the controllers* v *and* ϕ *are as defined in (2) and (4), respectively.*

Proof. From Claim 3, we see that $\phi(t)$ given in Eq. (4) eventually reduces to Eq. (3). Thus, Theorem 1 will hold and therefore we can conclude that the equilibrium point $\mathbf{x}_e$ of system (1) is asymptotically stable.

Simulation 2: To illustrate the effectiveness of the proposed formulas, we have generated the trajectory of the car-like robot from an initial position and orientation to the target position. In Fig. 3, the initial orientation of the car is 90°. With the steering angle given by Eq. (4), we see that the car-like robot converges nicely to the target.

5. Motion Control in the Presence of Fixed Obstacles

Let us fix $q > 0$ obstacles within the boundaries of the workspace. We assume that the lth obstacle is circular with center given as (o_{l1}, o_{l2}) and radius ro_l. We define the lth obstacle as

$$FO_l = \left\{(z_1, z_2) \in \mathbf{R}^2 : (z_1 - o_{l1})^2 + (z_2 - o_{l2})^2 \le ro_l^2\right\},$$

for $l = 1, 2, \ldots, q$.

To ensure that the entire vehicle safely steers pass any obstacle, we enclose the robot by the smallest circle, possible, of radius $r_V = \sqrt{(2\varepsilon_1 + L)^2 + (2\varepsilon_2 + l)^2}/2)$ as illustrated in Fig. 1, where ε_1 and ε_2 are the clearance parameters adopted from ref. [7].

Assumption 2: *There is sufficient free-space between any two stationary obstacles for the vehicle to steer through if warranted.*

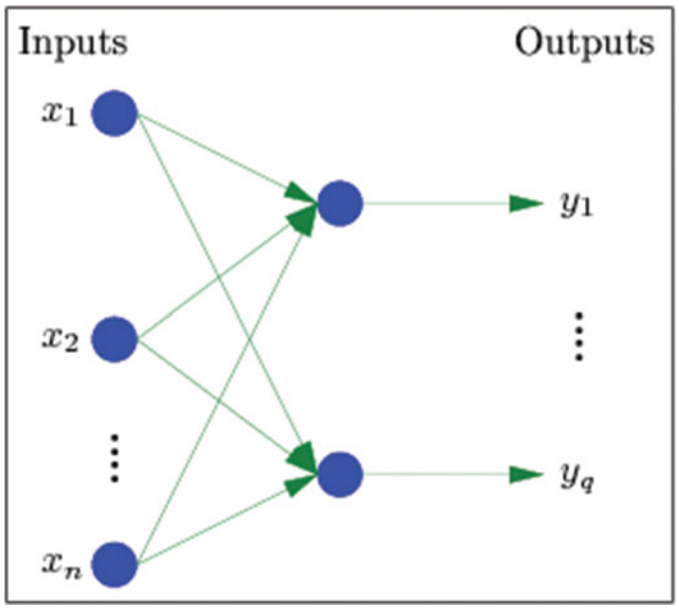

Fig. 4. (Colour online) Architecture of a standard feed-forward single layer ANN.

Remark. *Assumption 2 is inline with the work of Sharma et al. in ref. [7]. This is to ensure that the vehicle could fit into the free-space between two obstacles, in case one desires to steer the robot in between the two obstacles.*

5.1. An SLP-based model

Our method in the construction of a collision-free path for moving robot among obstacles is based on a feed-forward SLP. Figure 4 shows its general architecture.

A single layer ANN is made up of an input and an output layer. The input layer receives data from outside the network; the output layer sends data out of the network. The internal activity of a neuron is given as

$$\sum_{i=1}^{n} w_i x_i,$$

where w_i is called the weight associated with the input x_i.

We will use a supervised feed-forward SLP to model the steering angle ϕ so that the vehicle can avoid fixed obstacles that it may encounter within its path. Before we proceed, let's look at the following definition.

Definition 2. Let $d_{\max} > 0$ be a predefined scalar. All the points that lie in the open annulus $ro_l < (\sqrt{z_1 - o_{l1})^2 + (z_1 - o_{l2})^2} < ro_l + d_{\max}$ is said to be in the *sensing zone*. The sensing zone is precisely defined as

$$\mathbf{S} = \bigcup_{l=1}^{q} \left\{ (z_1, z_2) \in \mathbf{R}^2 : ro_l^2 < (z_1 - o_{l1})^2 + (z_2 - o_{l2})^2 < (ro_l + d_{\max})^2 \right\}.$$

Assumption 3: *The target position of the robot does not intersect with the sensing zone. This is,* $\mathbf{x}_e \notin \mathbf{S}$ *for all* $t \geq 0$.

Remark*: We shall see later that Assumption 3 will help us to construct an asymptotically stable system that will lead the robot from its initial position to a target position and remain there forever.*

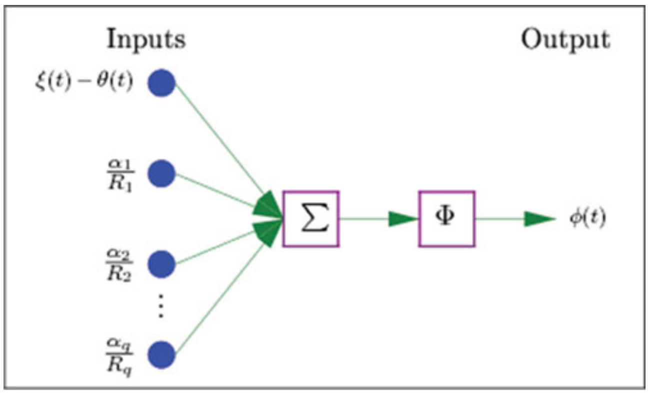

Fig. 5. (Colour online) The single-layer ANN used to model the steering angle $\phi(t)$.

Let $R_l = \sqrt{(x-o_{l1})^2+(y-o_{l2})^2}-(ro_l+r_V)$ be the distance from the robot to the circumference of the lth stationary obstacle. We make the following assumption:

Assumption 4: *The steering angle ϕ is inversely proportional to R_l.*

Remark. *Assumption 4 is justified since when the vehicle comes close to FO_l, the distance $R_l$will decrease. Thus, an increase in $|\phi|$ will enable the vehicle to effectively avoid FO_l.*

To model the motion of the car-like robot using the single-layer ANN, we enact the following rules:

Rule 1: In order to avoid collision with the stationary obstacles, R_l should be positive.

Rule 2: If the vehicle is approaching a stationary obstacle, it should change its direction when it enters the sensing zone.

Rule 3: When the vehicle enters the sensing zone, it should turn right or left whichever direction ensures shortest trajectory of the vehicle to its target.

Note that the size of the sensing zone is determined by $d_{\max}$. If $d_{\max}$ is large, then the robot will avoid the fixed obstacles from a greater distance. Likewise, if $d_{\max}$ is small, then the robot will avoid the fixed obstacles from a smaller distance. Thus, $d_{\max}$ is regarded as a *control* parameter in this paper.

Also keeping in mind, the main aim of the paper, i.e., the control laws must be continuous and should produce safe and shortest trajectory for the vehicle to follow and that ϕ is inversely proportional to R_l, we consider the inputs of the ANN as $x_0=\xi(t)-\theta(t)$, $x_1=\alpha_1/R_1$, $x_2=\alpha_2/R_2,\ldots,x_q=\alpha_q/R_q$, where

$$\alpha_l = \begin{cases} 0, & \text{if } R_l \ge d_{\max} \\ d_{\max}-R_l, & \text{if } R_l < d_{\max} \end{cases}.$$

The output is ϕ, which determines the steering angle.

Remark. *The function α_l serves two purposes here. Firstly, it ensures that the input is continuous everywhere (including on the boundaries of the sensing zone) and secondly it ensures that the turning is initiated once the vehicle enters the sensing zone.*

From the single-layer ANN shown in Fig. 5, we see that the output is given by

$$\phi(t) = \Phi\left(w_0(\xi-\theta)+\sum_{l=1}^{q}\frac{w_l\alpha_l}{R_l}\right),$$

where $\Phi(\cdot)$ is called the activation function and w_l is called the weight corresponding to the lth input. The value of w_l will be found by training the network. This will be illustrated in the next section.

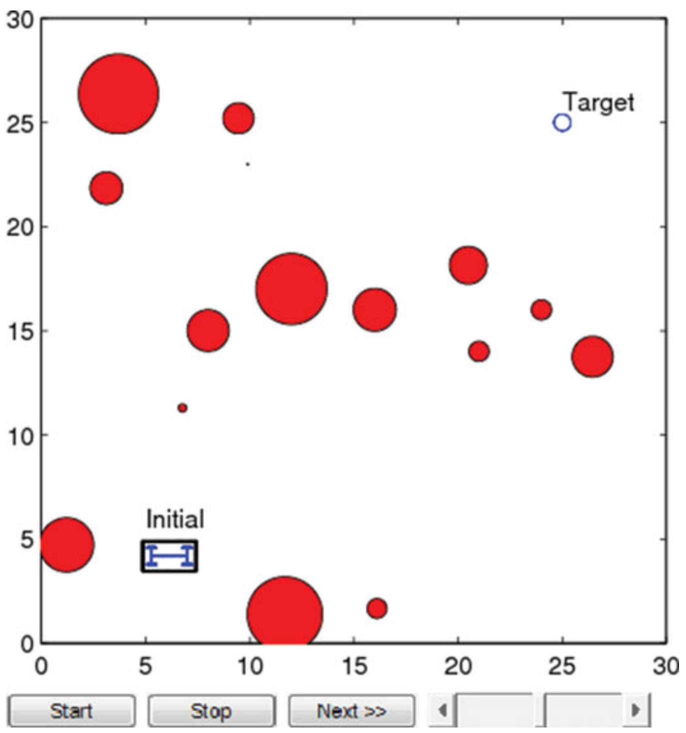

Fig. 6. (Colour online) Training environment.

5.2. Training the network

We have used supervised SLPs where the network will be trained using some known past data. The training will be done in two steps: (i) collecting the training data, and (ii) finding a set of weights that best fits the collected data satisfying the internal activities of the neurons.

Collecting the training data. The training data, which is a set of inputs and the corresponding outputs, were obtained using Matlab. The velocity algorithm given in Eq. (2) was applied to a car-like robot and its motion was manually controlled in the environment shown in Fig. 6.

We used the slider (bottom right of Fig. 6) to control the motion of the robot and guide it to its designated target. When the slider is kept at the center, the value of ϕ is zero so the robot moves straight (without turning) to the target. Moving the slider to the right will increase the value of ϕ and thus turning the robot to the right. Moving the slider to the left will decrease the value of ϕ so that robot turns to the left. Data from several scenarios were recorded; in each case the size and position of obstacle and the initial and target positions of the robot were chosen randomly.

Finding the best set of weights. Let us represent the collected set of data in a matrix form. Let A be an $m \times n$ matrix containing the input data. Here n is the number of inputs and m is the number of sets of collected data. Also let Y be the matrix containing the m sets of output data. Furthermore, if we let W be a matrix containing the weights, then our training problem simplifies to minimizing the quantity $\|AW - \Phi^{-1}(Y)\|^2$, where Φ is the activation function, which is the arctan activation function $\Phi(X) = \frac{7}{9}\tan^{-1}(X)$ in this paper.

The reasons for choosing this activation function are:

(1) $\Phi(X) = \frac{7}{9}\tan^{-1}(X)$ is a continuous and differentiable function on the interval $\left(-\frac{7\pi}{9}, \frac{7\pi}{9}\right)$. Thus, the output, the steering angle, will be continuous.
(2) The output of the network is the steering angle, ϕ. In nature, ϕ is restricted, say, between -70° (minimum turning angle) and 70° (minimum turning angle). Now since $-90^\circ < \tan^{-1}(X) < 90^\circ$, it follows that $-70^\circ < \phi < 70^\circ$.

Hence, our training problem further simplifies to minimizing

$$\left\|AW - \tfrac{9}{7}\tan(Y)\right\|^2.$$

One convenient way of solving this problem is by using the method of least squares where the solution can be obtained by solving the system of linear equation

$$(A^T A)W = \frac{9}{7}A^T \tan(Y).$$

Main result: Let w_l be the weight from the lth input neuron to the output neuron. Using the least squares method described above, it was found out that the weights follow the following pattern:

$$w_0 = 1;$$
$$w_l \approx \begin{cases} 1 \text{ when the vehicle avoids the } l\text{th obstacle} \\ \quad \text{from the left;} \\ -1 \text{ when the vehicle avoids the } l\text{th obstacle} \\ \quad \text{from the right.} \end{cases}$$

Now consider a straight line passing through (o_{l1}, o_{l2}) and (p_1, p_2) whose equation in the $z_1 z_2$ plane is given by $(z_2 - o_{l2})(o_{l1} - p_1) - (z_1 - o_{l1})(o_{l2} - p_2) = 0$. With the help of this equation, we define a function f_l, which is dependent on the obstacle's center and the robot's current and target position as:

$$f_l = (y - o_{l2})(o_{l1} - p_1) - (x - o_{l1})(o_{l2} - p_2).$$

We note here that when the vehicle avoids the lth obstacle from right, the value of f_l will be zero or positive. Similarly, when the vehicle avoids the lth obstacle from left, the value of f_l will be negative. We can therefore generalize the weights as follows:

$$w_0 = 1;$$
$$w_l = \begin{cases} 1 & \text{if} f_l < 0 \\ -1 & \text{if} f_l \geq 0 \end{cases}.$$

With these information, we have the following definition:

Definition 3: The steering angle $\phi(t)$ used in system (1) is depended on ξ, θ, R_l, f_l, and $d_{\max}$ according to the rule

$$\phi(t) = \frac{7}{9}\tan^{-1}\left(\xi - \theta + \sum_{l=1}^{q} \frac{w_l \alpha_l}{R_l}\right),$$

where

$$\alpha_l = \begin{cases} 0, & \text{if } R_l \geq d_{\max} \\ d_{\max} - R_l, & \text{if } R_l < d_{\max} \end{cases}$$

and $w_l = \begin{cases} 1 & \text{if } f_l < 0 \\ -1 & \text{if } f_l \geq 0 \end{cases}$. While the steering angle $\phi(t)$ is capable of deviating the motion of the vehicle away from the fixed obstacle to avoid collisions, safety of the robot is also dependent on the velocity. Commonly seen in literature, the planar robot should slow down on approach to a fixed obstacle.[6–8] Thus, we redefine the velocity as

$$v(t) = |v_0| \frac{\|\mathbf{x}(t) - \mathbf{x}_e\|}{\|\mathbf{x}(0) - \mathbf{x}_e\|} \prod_{l=1}^{q} \left(1 - \frac{\alpha_l}{d_{\max}}\right).$$

The inclusion of the factor $\prod_{l=1}^{q}(1 - \frac{\alpha_l}{d_{\max}})$ is to ensure that the robot would slow down as soon as it enters the sensing zone. We can conclude that the mobile car-like robot A whose motion is governed

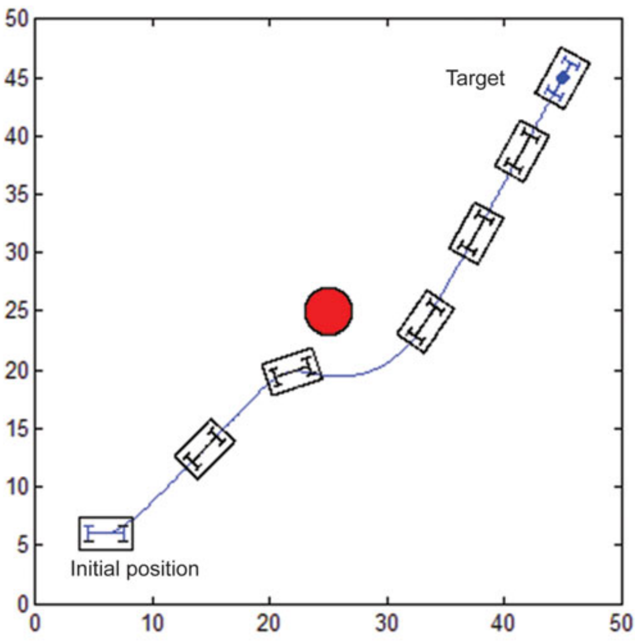

Fig. 7. (Colour online) Trajectory of the car-like robot with initial position (6, 6) and the target placed at (45, 45).

by the system (1) would move from an initial position to the target while avoiding collisions with fixed obstacles along its path if the controllers v and ϕ are defined as

$$\left.\begin{aligned} v(t) &= |v_0| \tfrac{\|\mathbf{x}(t)-\mathbf{x}_e\|}{\|\mathbf{x}(0)-\mathbf{x}_e\|} \prod_{l=1}^{q} \left(1 - \tfrac{\alpha_l}{d_{\max}}\right), \\ \phi(t) &= \tfrac{7}{9} \tan^{-1} \left(\xi - \theta + \sum_{l=1}^{q} \tfrac{w_l \alpha_l}{R_l}\right). \end{aligned}\right\} \quad (5)$$

We further note that the controllers are bounded and continuous at every point over the domain

$$D = \{\mathbf{x}(t) \in \mathbf{R}^2 : \mathbf{x}(0) \neq \mathbf{x}_\mathrm{e} \cap R_l > 0 \quad \text{for } l = 1, 2, \ldots, q\}.$$

Simulation 3: To illustrate the effectiveness of the proposed formulas, we have generated the trajectory of the car-like robot from an initial position to the target position in the presence of stationary circular obstacle(s).

Firstly, Fig. 7 shows the convergence of the car-like robot to its designated target in the presence of one obstacle.

Figure 8 shows explicitly the time evolution of the relevant nonlinear controllers (v and ϕ) along the trajectory of system (1). One can clearly notice the asymptotic convergence of these controllers at the final configuration implying the effectiveness of the new controllers.

Secondly, in Fig. 9, we have placed multiple stationary obstacles of random sizes and positions in the workspace. Again the car-like robot converges nicely to the target while avoiding the obstacles along its path.

Figure 10 shows the time evolution of the nonlinear controllers, v and ϕ. Looking at the velocity graph, we see that the planar robot slowed down on approach to the fixed obstacles. After it avoids collision with the fixed obstacles, it gained speed rapidly and then slowed down on approach to the target. Eventually, at the center of the target the velocity and the steering angle became zero.

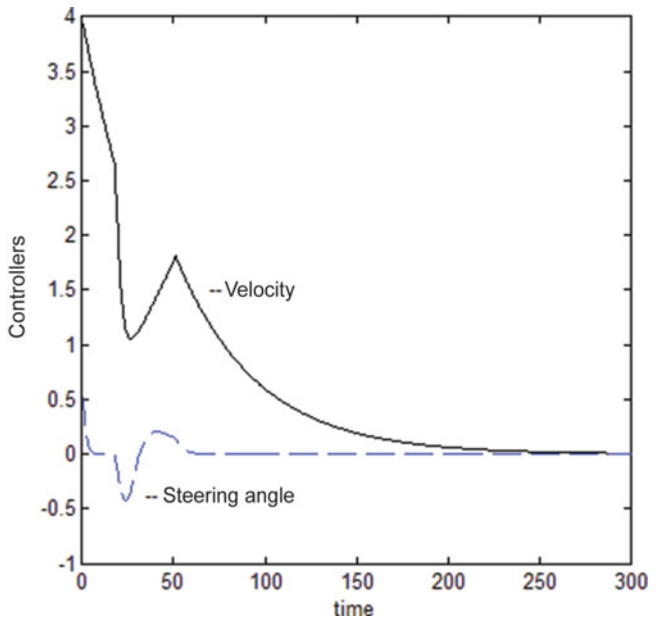

Fig. 8. (Colour online) Evolution of the controllers.

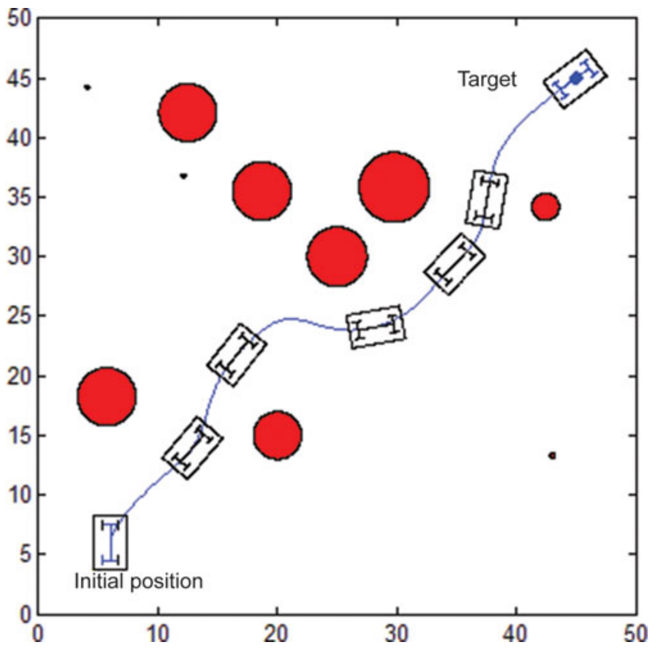

Fig. 9. (Colour online) Trajectory of the car-like robot with initial position (6, 6) and the target placed at (45, 45).

5.3. Stability analysis in the presence of obstacles

Theorem 3: *The equilibrium point* $\mathbf{x}_e$ *of system (1) is asymptotically stable provided the controllers, v and* ϕ*, are as defined in Eq. (5).*

Proof. With the injection of Assumption 2, we see that the robot will never be trapped in between two obstacles. Next, Assumption 3 ensures us that before the robot converges to its target, the controllers given by Eq. (5) will reduce to that given in Eqs. (2) and (4). Thus, Theorem 2 will hold and we can therefore conclude that system (1) will be asymptotically stable.

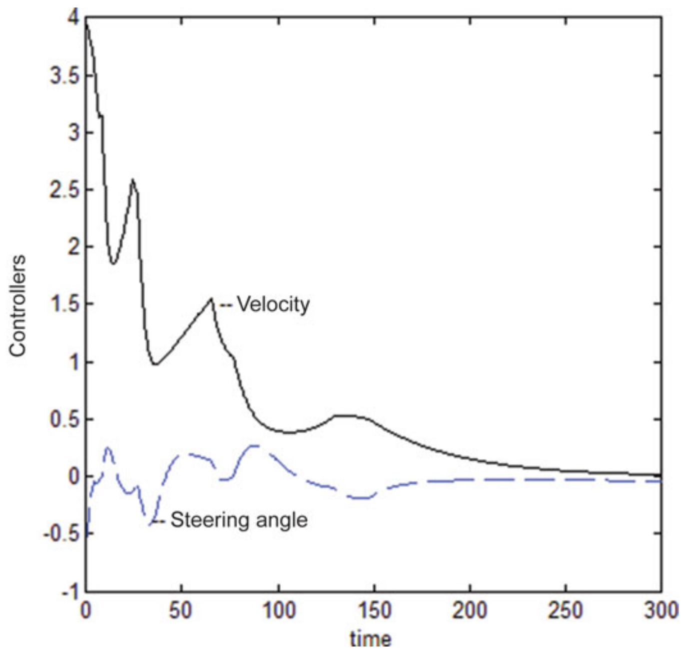

Fig. 10. (Colour online) Evolution of the controllers.

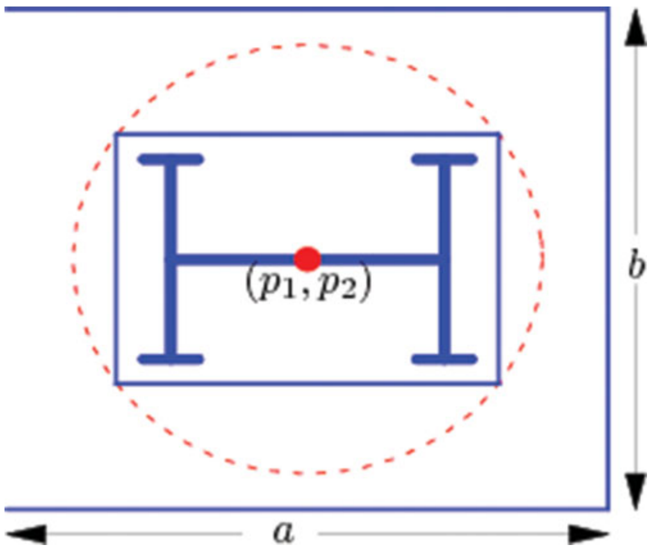

Fig. 11. (Colour online) Row-structured parking bay with length a and width b, adopted from ref. [7].

6. Posture Control

This section achieves a predetermined final orientation of the vehicle. This is done by constructing a virtual parking bay that surrounds the target and forcing the vehicle to park directly parallel to the boundary line of the virtual bay, hence forcing the desired final postures. Figure 11 shows the virtual parking bay with length a and width b.

In order to park the vehicle parallel to the boundary lines of the virtual parking bay, the vehicle needs to avoid the lines at all time $t > 0$.[7] For the vehicle to avoid any boundary line of the parking bay, we utilize the idea inspired by the work carried by Sharma in ref. [7] where the author designed the minimum distance technique in which the minimum distance from a robot to a line segment was calculated and the resultant closest point avoided. Avoidance of the closest point on a line segment simply affirms that the mobile robot avoids the whole line segment.

According to Sharma,[7] if the kth line segment in the z_1z_2 plane has initial coordinates (a_{k1}, b_{k1}) and final coordinate (a_{k2}, b_{k2}), then the point (x_k^*, y_k^*) on the kth line segment closest to the center of the vehicle is given by

$$(x_k^*, y_k^*) = (a_{k1} + \lambda_k(a_{k2} - a_{k1}), b_{k1} + \lambda_k(b_{k2} - b_{k1})),$$

where

$$\lambda_k = \frac{(x - a_{k1})(a_{k2} - a_{k1}) + (y - b_{k1})(b_{k2} - b_{k1})}{(a_{k2} - a_{k1})^2 + (b_{k2} - b_{k1})^2}.$$

If $\lambda_k \geq 1$, then we let $\lambda_k = 1$, in which case and if $\lambda_k \leq 0$, then we let $\lambda_k = 0$. Otherwise we accept the value of λ_k between 0 and 1.

Since avoidance of the whole line segment is equivalent to avoiding the point (x_k^*, y_k^*), we would consider (x_k^*, y_k^*) as a point obstacle and hence avoid it using the technique discussed in Section 5. As such, we let

$$D_k = \sqrt{(x - x_k^*)^2 + (y - y_k^*)^2} - r_v$$

$$g_k = (y - y_k^*)(x_k^* - p_1) - (x - x_k^*)(y_k^* - p_2)$$

$$\gamma_k = \begin{cases} 0, & \text{if } D_k \geq d_{\max} \\ d_{\max} - D_k, & \text{if } D_k < d_{\max} \end{cases}$$

$$\delta_k = \begin{cases} 1 & \text{if} g_k < 0 \\ -1 & \text{if} g_k \geq 0 \end{cases}.$$

For the vehicle to avoid the circular fixed obstacles and the point obstacles on the boundary lines of the parking bay, we modify the controllers v and ϕ from Section 5 as

$$\left.\begin{aligned} v(t) &= |v_0| \tfrac{\|\mathbf{x}(t)-\mathbf{x}_e\|}{\|\mathbf{x}(0)-\mathbf{x}_e\|} \prod_{l=1}^{q} \left(1 - \tfrac{\alpha_l}{d_{\max}}\right) \prod_{k=1}^{2} \left(1 - \tfrac{\gamma_k}{d_{\max}}\right), \\ \phi(t) &= \frac{7}{9} \tan^{-1} \left(\xi - \theta + \sum_{l=1}^{q} \frac{w_l \alpha_l}{R_l} + \sum_{k=1}^{2} \frac{\gamma_k \delta_k}{D_k} \right). \end{aligned}\right\} \tag{6}$$

Simulation 4: To illustrate the effectiveness of the proposed formulas, we have generated a trajectory of the car-like robot from some initial position to the target position as shown in Figs. 12 and 13. On the one hand, the car-like robot avoided all fixed obstacles along its path while on the other hand, the car parked correctly inside the virtual parking bay. In both the cases, the desired final orientations are 0 radians.

Figure 14 shows explicitly the time evolution of the relevant nonlinear controllers (v and ϕ) corresponding to the trajectories for Fig. 13. One can clearly notice the asymptotic convergence of these controllers at the final configuration implying the effectiveness of the control laws.

7. Concluding Remarks

This paper proposes a solution to the motion planning and control problem of a car-like robot. The multi-tasking problem of target convergence, obstacle avoidance, and parking maneuverability is successfully tackled in this paper. Moreover, mechanical singularities tagged to the system and the bounds on steering angle are carefully incorporated and reflected into the proposed formulas. Our method in the construction of a collision-free path in a workspace fixed with randomized multiple obstacles, of arbitrary sizes, is based on learning via the SLP. The SLP is used to determine the steering angle of the vehicle in the obstacle-ridden workspace.

The training data for the neural networks are obtained using computer simulations where the initial paths are traced by the user. The user plays the role of the supervisor to train the neural network. Several sets of data were obtained; in each case the size and position of obstacles and the initial and

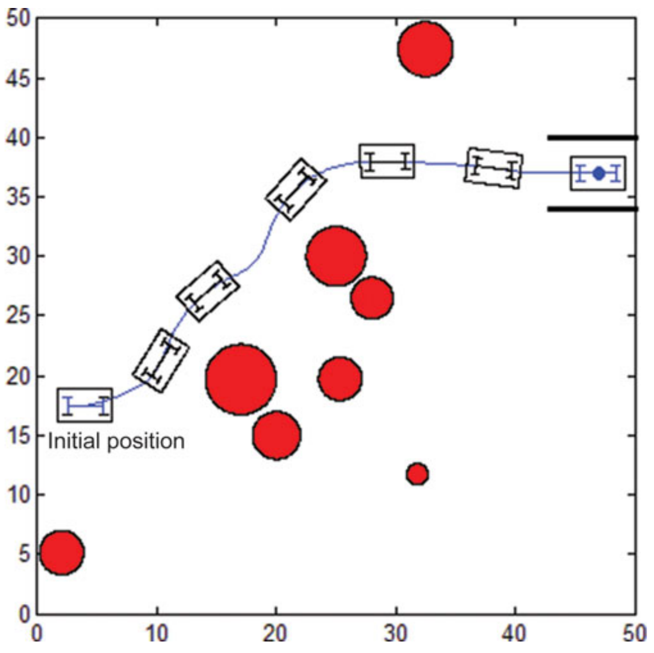

Fig. 12. (Colour online) Trajectory of the car-like robot from initial position (4, 17.5) to the target position (47, 37).

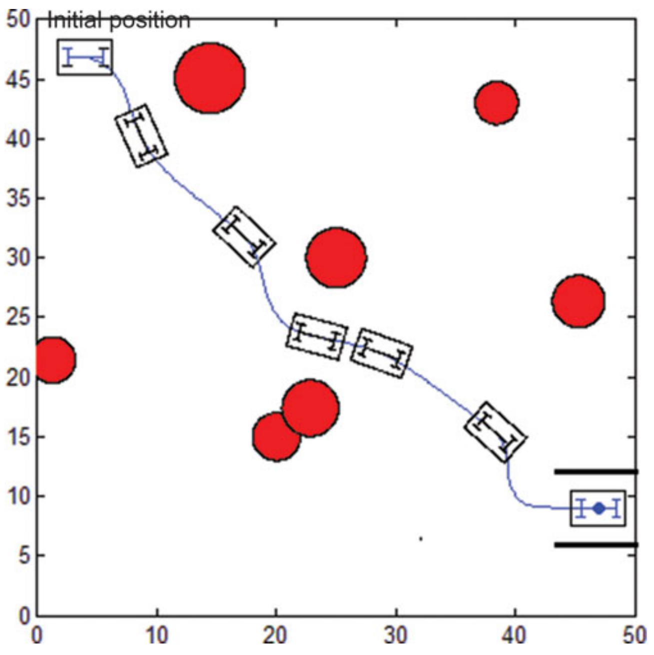

Fig. 13. (Colour online) Trajectory of the car-like robot with initial position (4, 46.9) and the target placed at (47, 9).

target positions were chosen randomly. The network was trained using these data by the least squares method and a new explicit formula for the steering angle in terms of the input was derived.

With the help of the proposed nonlinear controllers, we have achieved the point stabilization of the system. However, the posture stability is only achieved by forcing the vehicle to park correctly inside a virtual parking bay, thus giving the desired orientation at the target.

Finally, computer simulations of the generated path are presented to illustrate the effectiveness of our proposed method.

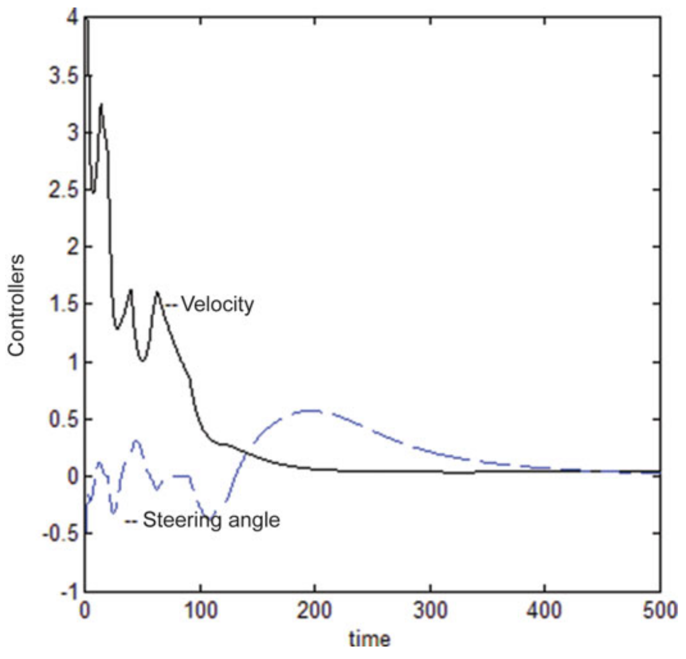

Fig. 14. (Colour online) Evolution of the controllers.

References

1. O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.* **7**(1), 90–98 (1986).
2. L.-F. Lee and V. Krovi, "A Standardized Testing-Ground for Artificial Potential-Field Based Motion Planning for Robot Collectives," *Proceedings of the Performance Metrics for Intelligent Systems Workshop*, Gaithersburg, MD, USA (Aug. 21–23, 2006).
3. Y. S. Nam, B. H. Lee and N. Y. Ko, "An Analytic Approach to Moving Obstacle Avoidance Using An Artificial Potential Field," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, PA, USA (Aug. 5–9, 1995) vol. 2, pp. 482–487.
4. E. Rimon, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.* **8**(5), 501–517 (1992).
5. P. Song and V. Kumar, "A Potential Field Based Approach to Multi-Robot Manipulation," *Proceedings of the IEEE International Conference on Robotics & Automation*, Washington, DC, USA (May 11–15, 2002).
6. J. Vanualailai, S. Nakagiri and J. Ha, "A solution to two dimension findpath problem," *Dyn. Stab. Syst.* **13**, 373–401 (1998).
7. B. Sharma, New Directions in the Applications of the Lyapunov-Based Control Scheme to the Findpath Problem *Ph.D. Thesis* (Suva, Fiji Islands: University of the South Pacific, 2008). Available at: http://www.staff.usp.ac.fj/∼sharma_b/Bibhya_PhD.pdf.
8. B. Sharma, J. Vanualailai and A. Prasad, "Formation control of a swarm of mobile manipulators," *Rocky Mt. J. Math.* **41**(3), 900–940 (2011).
9. R. W. Brockett, "Asymptotic Stability and Feedback Stabilisation," **In**: *Differential Geometry Control Theory* (Springer-Verlag, 1983) pp. 181–191.
10. K. Fujimura and H. Samet, "A hierarchical strategy for path planning amongst moving obstacles," *IEEE Trans. Robot. Autom.* **5**(1), 61–69 (1989).
11. R. A. Jarvis, "Growing polyhedral obstacles for collision free paths," *Aust. Comput. J.* **15**(3), 103–111 (1983).
12. S. Zeghloul, C. Helguera and G. Ramirez, "A local-based method for manipulators path planning, using sub-goals resulting from a local graph," *Robotica* **24**(5), 539–548 (2006).
13. D. Roy, "Study on the configuration space based algorithmic path planning of industrial robots in an unstructured congested three-dimensional space: An approach using visibility map," *J. Intell. Robot. Syst.* **43**(2–4), 111–145 (2005).
14. L. Edelstein-Keshet, "Mathematical Models of Swarming and Social Aggregation," *Proceedings of 2001 International Symposium on Nonlinear Theory and Its Applications*, Miyagi, Japan (Oct. 28–Nov. 1, 2001) pp. 1–7.
15. B. R. Fajen, W. H. Warren, S. Temizer and L. P. Kaelbling, "A dynamical model of visually-guided steering, obstacle avoidance, and route selection," *Int. J. Comput. Vis.* **54**(1–3), 13–34 (2003).

16. I. Rivals, D. Canas, L. Personnaz and G. Dreyfus, “Modeling and Control of Mobile Robots and Intelligent Vehicles by Neural Networks,” *IEEE Conference on Intelligent Vehicles*, Paris, France (Oct. 24–26, 1994) pp. 137–142.
17. K. Izumi, K. Watanabe, Y. Koga and K. Kiguchi, “Control of Nonholonomic Mobile Robots Using a Feedback Error Learning,” *SICE Annual Conference in Fukui*, Fukui University, Japan (Aug. 2003) pp. 2979–2984.
18. D. Janglova, “Neural networks in mobile robot motion,” *Int. J. Adv. Robot. Syst.* **1**(1), 15–22 (2004).
19. S. Naoui, D. Jarbi and M. Benrejeb, “Neural internal model control of a mobile robot,” *J. Autom. Syst. Eng.* **2**(3) (2008).Available at: http://jase.esrgroups.org/edition-2008.php.
20. B. Sharma, A. Prasad and J. Vanualailai, “A collision-free algorithm of a point-mass robot using neural networks,” *J. Artif. Intell.* **3**(1), 49–55 (2012).