Received: April 4, 2021. Revised: April 20, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Path Planning and Control Strategy Design for Mobile Robot Based on Hybrid Swarm Optimization Algorithm

Omar Abdul Razzaq Abdul Wahhab[1]* **Ahmed Sabah Al-Araji[1]**

[1]*Computer Engineering Department, University of Technology, Baghdad, Iraq*
* Corresponding author's Email: ce.19.08@grad.uotechnology.edu.iq

Abstract: This paper presents an improvement of the actual output trajectory tracking performance of a mobile robot based on convolutional neural network controller with off-line and on-line tuning Back-Propagation algorithms. The goals of this strategy are to find the optimal path to direct its movement and to design Convolutional Neural Network Trajectory Tracking (CNNTT) controller in order to control the nonlinear kinematics mobile robot system. Therefore, a hybrid swarm optimization algorithm uses for solving the two most important problems of path planning; the first is that the path must avoid collision with obstacles, and the second it must reduce the length of the path to a minimum. This paper will discuss the finding of the shortest path with the optimum cost function by using three optimizations' algorithms; Chaotic Particle Swarm Optimization (CPSO) algorithm, A-star algorithm, and a hybrid swarm optimization algorithm (ACPSO). The task of the proposed feedback (CNNTT) controller is to obtain precisely and quickly the robust left and right wheels velocity which are used to control the position and orientation of the mobile robot system. These algorithms are simulated by MATLAB in a fixed obstacles environment to show the effectiveness of the hybrid swarm optimization algorithm in terms of the minimum number of an evaluation function and the shortest path length as well as the results of the proposed method showing that the (CNNTT) controller is accurate in terms of the mobile robot follows the desired paths quickly through fast obtaining the (CNNTT) controller's parameters and a smooth linear wheels velocity actions are generated for mobile robot system with minimum number of cost-function evolutions that minimized the tracking error x-position around 4 cm and y-position around 2.5 cm and zero approximately orientation error as well as no oscillation in the responses. Finally, we confirm the effectiveness of the numerical simulation results of the proposed control strategy through comparison other types of controller simulation results.

Keywords: Mobile robot, Path planning, Chaotic particle swarm optimization (CPSO) algorithm, A-star algorithm, Fixed obstacles, Trajectory tracking, Convolutional neural network.

1. Introduction

Many commercial robots are now available such as robotic vacuum cleaners and lawn mowers [1] as well as mobile robots serve many practical purposes in real world applications such as industry, weather forecasting, mining, science, education, entertainment, security, and the military [2, 3]. So, a number of challenges must be addressed in order to improve the path planning and tracking mobile robot such as simultaneously localizing and mapping [4], object detection and tracking [5], the convergence of static and mobile surveillance systems [6]. On the other hand, when we are focusing on path planning problem of mobile robot and how to find the optimal or shortest path between two points while avoiding collision with obstacles. Many researchers have been studied this issue and tried to solve it by using optimization algorithms such as: in [7] explained Ant Colony Optimization (ACO) was used in the search process for the globally optimal direction, pheromone diffusion and geometric local optimization are combined that led to the ant scanning process and the present path pheromone diffuses in the direction of the possible field power, so ants prefer to aim for a higher fitness subspace, and the search space of the test pattern becomes smaller. Also, firefly algorithm

Received: April 4, 2021. Revised: April 20, 2021.

is discussed in [8] as swarm intelligence in order to achieve detailed and effective solutions, the current MO-FA deals with three separate aims. These aims are as follows: protection of the road, length of the path and smoothness of the route (related to the energy consumption). Ant and Bee colony optimizations are presented in [9] as a local search technique to optimize the feasible path discovered by a series of local procedures that generated a route for mobile robot depending on different optimization algorithms inspired by the organism's attitudes, fitness functions, and various constraints in the workspace. However, in [10] developed an accurate mapping of the trajectory of a moving robot using Particle Swarm Optimization PSO with radial foundation functions that described the working area of the moving mobile robots. In addition, many researchers have been designed control algorithms of trajectory tracking for mobile robots in order to solve the motion control and the platform can follow the desired path with minimum tracking pose error such as: a cognitive path planning with nonlinear neural networks like a PID controller is presented in [11] to find and track the best desired path for the mobile robot based on Particle Swarm Optimization (PSO) algorithm. In [12] proposed a genetic algorithm that gives a particular form of solution to the complex motion preparation problems of mobile robots in uncertain dynamic conditions dependent on action dynamics. In [13] neural network with Q-learning is used to address route planning issues in the generated solution. Also, nonlinear predictive neural network controller is explained in [14] for achievement trajectory tracking for mobile robot based on posture identifier model. Moreover, a feedforward neural controller and feedback kinematics controller are used for wheeled mobile robot to track different types of the desired trajectories that illustrated in [15]. In [16] efficiency was compared with the current Wavelet Neural Network (WNN) architecture of smart controllers for mobile robot route planning in an uncertain area. In the paper [17] back-stepping controller with on-line slice genetic algorithm is proposed for development path-tracking for real mobile robot. The problems definition of this work can be divided into two problems: the first problem that will encounter our work is generated optimal or near optimal desired path with achievement the two conditions; the first is that the path must avoid collision with obstacles, and the second it must reduce the length of the path to a minimum. The second problem that will encounter our work is designed motion controller of trajectory tracking for mobile robot because the mobile robot platform has the highly nonlinear kinematic model and time variant outputs states as well as it has under-actuated system. Therefore, to accurate and precise track the desired path in terms of minimum position and orientation errors, without oscillation and no overshoot in the pose of mobile robot system, we need to solve the problems definition. So, the motivation of this work is taken from [1, 11, 14, 17, 18]. The summarized contributions of this paper are to solve the problem statement through i) generated optimal or near optimal desired path with free collision and short path by using the proposed hybrid swarm algorithm (ACPSO) when compared to A-star algorithm and CPSO algorithm. ii) smooth with best values of two wheels velocities control actions that are generated using numerical simulation based on the proposed convolutional neural network trajectory tracking control algorithm. Therefore, the effectiveness of the proposed strategy leads to track and stabilize the mobile robot in the desired locations and orientations with minimum tracking error through control the pose of mobile robot system.

This paper is organized as follows: Section 2 explains the mathematical mobile robot kinematic model. Section 3 illustrates the proposed path-planning and control strategy design. In section 4 numerical simulation results are illustrated of the effectiveness and performance of the proposed (path planning algorithm and motion controller method). Finally, the conclusions for the proposed strategy are given in section 5.

2. Mobile robot kinematics model

The platform of the Wheeled Mobility Robot (WMR) is shown in Fig. 1, which has two wheels mounted on a parabolic shaft, and two multi-directional wheels are installed in front and rear of the platform that are carried by the mechanical structure and keeps the body stable during the mobile robot is motion and orientation [11].

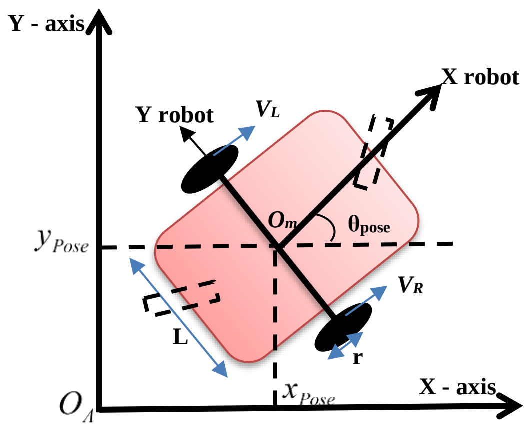


Figure. 1 Platform mobile robot

Received: April 4, 2021. Revised: April 20, 2021.

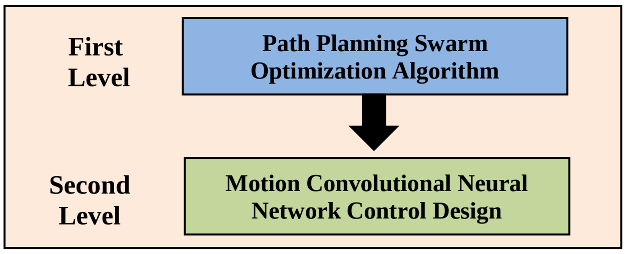

First Level
Path Planning Swarm Optimization Algorithm
Second Level
Motion Convolutional Neural Network Control Design

Figure. 2 Proposed strategy diagram

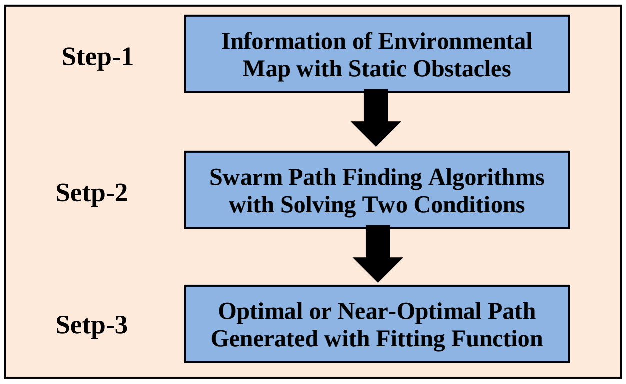

Step-1
Information of Environmental Map with Static Obstacles
Setp-2
Swarm Path Finding Algorithms with Solving Two Conditions
Setp-3
Optimal or Near-Optimal Path Generated with Fitting Function

Figure. 3 The proposed path planning strategy

Two independent analog Direct Current (DC) motors are actuated as the right and left wheel actuators of the wheeled robot for movement and platform steer. The point O_m is the location of the WMR center mass, the two drive wheels are connected to the center of the axis [17]. The kinematics equation of the mobile robot platform has the highly nonlinear and time variant outputs states as well as it has under-actuated model because the mobile robot model is multi-input multi-output system and the number of the input states are two states (left and right wheels velocities) but the output states are three based on its position in the global coordinate frame $[O_A, X_A, Y_A]$ and the pose surface are x_{pose} and y_{pose} are the coordinates of a point O_m and θ_{pose} is the mobile robot's direction angle measured from the X_A axis. Therefore, these three generalized coordinates can describe the configuration of the mobile robot. So, the computer simulation equation can be represented as follows [15, 17]:

$$x_{pose}(kT) = x_{pose}((k-1)T)) + \frac{((V_R(kT) + V_L\,(kT)) \times cos(\theta(kT)_{pose} \times T))}{2} \quad (1)$$

$$y_{pose}(kT) = y_{pose}((k-1)T)) + \frac{V_R(kT) + V_L(kT)) \times \sin(\theta(kT)_{pose} \times T)}{2} \quad (2)$$

$$\theta(kT)_{pose} = \theta((k-1)T)_{pose} + \frac{((V_R(kT) - V_L\,(kT))}{L} \quad (3)$$

Where, $V_R(kT)$ is denoted as right wheel velocity of the platform. $V_L(kT)$ is denoted as left wheel velocity of the platform. L is denoted as the length between the driving wheels of the platform. T is denoted as the sampling time of the numerical calculation.

3. Path planning and control strategy design

In this work, the proposed strategy consists of two levels as shown in Fig. 2.

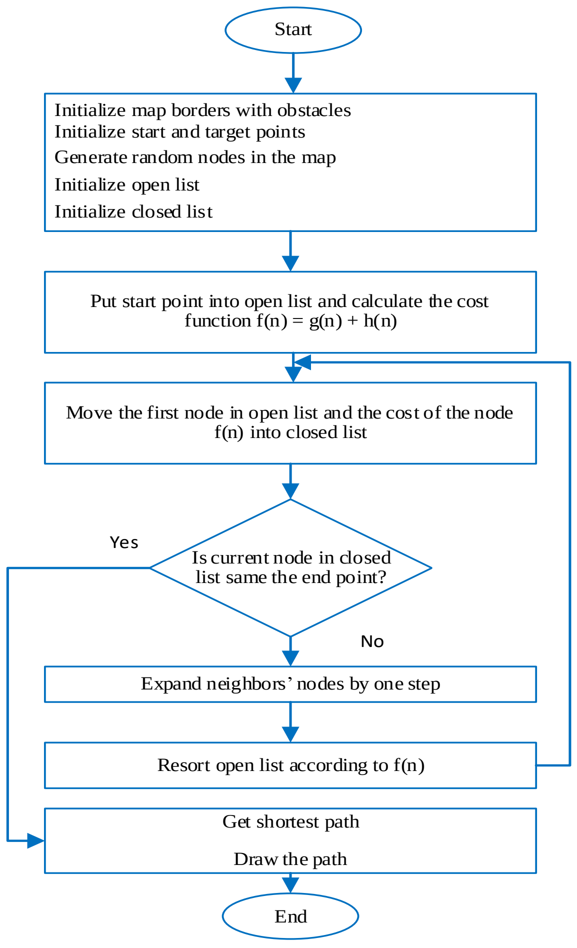
The first level is path planning for the mobile robot based on swarm optimization algorithm. The second level is motion control design for the mobile robot based on convolutional neural network.

3.1 Path planning swarm optimization algorithms

When we want to move a mobile robot between two points, the first problem that will encounter our work, the generated optimal or near optimal desired path with two conditions must be solved; the first condition is that the path must avoid collision with obstacles, and the second condition must reduce the length of the path to a minimum. In this work, this issue can be solved by using the proposed path planning strategy that consists of three steps, as shown in Fig. 3 and using three optimization algorithms; Chaotic Particle Swarm Optimization CPSO algorithm, A-star algorithm, and proposed hybrid swarm optimization algorithm ACPSO based on A-star algorithms and CPSO algorithm.

3.1.1. A-star path-planning algorithm

The A-star Algorithm knowns as a heuristic search algorithm, finds the optimal path by checking among all possible routes of a solution to problems with the minimal cost. It visits the nodes in the graph from the starting node to the target node. The prescriptive information about the properties of the issue is applied to guide its performance [19]. It is based on two standards algorithms, the first is the Dijkstra's algorithm, and the second is Greedy Best-First-Search's algorithm. Dijkstra's algorithm is designed to find the shortest path in a graph between two nodes. The algorithm visits the nodes in a graph one by one beginning from the starting point of the object [20]. The Greedy Best-First-Search's algorithm also keeps track of a frontier to locate the



Start
Initialize map borders with obstacles
Initialize start and target points
Generate random nodes in the map
Initialize open list
Initialize closed list
Put start point into open list and calculate the cost function f(n) = g(n) + h(n)
Move the first node in open list and the cost of the node f(n) into closed list
Is current node in closed list same the end point?
Yes
No
Expand neighbors' nodes by one step
Resort open list according to f(n)
Get shortest path
Draw the path
End

Figure. 4 Flowchart of A-star algorithm

target. This algorithm makes use of a heuristic function which determines approximately how far from the goal a particular node is. The Dijkstra's algorithm selects the node nearest to the starting point, while here the node closest to the goal is selected and given higher priority than those nodes which are far away. A-STAR balances between Dijkstra's algorithm by finding the shortest path without fail, *g(n)* and the Best-First-Search's algorithm by estimate the distance to the target, $h(n)$. In the main loop, the algorithm repeatedly checks which (n) vertex has the lowest value of $f(n)$ as in the evaluation Eqs. (4) and (5).

Fig. 4 demonstrates the flowchart of the A-star algorithm. Dijkstra's algorithm consumes time and resources to explore unsafe directions, while Greedy Best-First-Search always fails to find the shortest path to reach a goal. Therefore, Algorithm A-star uses both distances from the starting point and approximate distance to the target point to remove the limitations of these traditional algorithms by combining these two algorithms [21, 22].

$$f(n) = g(n) + h(n) \tag{4}$$

$$h(n) = \sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2} \tag{5}$$

Where, n is denoted by the current node, $g(n)$ is denoted by the cost distance function from starting point to the current node n, and $h(n)$ is denoted by the estimation minimum cost distance function from the current node to end point that is calculated by Eq. (5).

3.1.2. Chaotic particle swarm optimization path-planning algorithm

In general, PSO is an experimental community based on multi-point research technology that simulates the social behavior of a flock of birds, a school of fish, etc. [23]. Research begins with a set of research points called molecules because particles have a memory and they save part of their previous condition. The particles maintain their individuality in all cases, although they share the same point in the belief of space without limitations. The individuality and sociality are two randomly weighted factors that influenced the particle's movement. The definition of individuality is "the tendency to return to the particles best past situation" while sociality is defined as "the tendency to move towards the neighborhood's best previous situation". Each particle is encoded by a location vector (initially randomly chosen) and the position is updated using its velocity (randomly chosen at the beginning) in successive iterations. At each time step, PSO changes the speed of each particle to its optimum positions. Acceleration is measured in random terms, with separate random numbers are generated to accelerate to the best positions. The search by PSO algorithm is subject to stagnation due to early convergence so the search process should be diversified [24]. Chaos is introduced into the PSO to induce more randomness in the search for PSO [25]. A small error in particle position may make a big difference to their behavior for a long time and prevent them from getting trapped in some local optimal solution. After applying chaotic, Eq. (6), (7) and (8), each particle updates its velocity and position by using the Eq. (9) and (10) [11, 25].

$$\beta^{b+1} = \mu \times \beta^b(1 - \beta^b) \quad 0 \le \beta^1 \le 1 \tag{6}$$

$$W = W_{max} - [(W_{max} - W_{min}) \times (\frac{iter}{iter_{max}})] \tag{7}$$

$$W_{new} = W \times \beta^{b+1} \tag{8}$$

$$v(i,j)_a^{b+1} = W_{new} \times v(i,j)_a^b + c_1 r\left(pbest(i,j)_a^b - xy(i,j)_a^b\right) + c_2 r\left(gbest(i,j)^b - xy(i,j)_a^b\right) \tag{9}$$

$$xy(i,j)_a^{b+1} = xy(i,j)_a^b + v(i,j)_a^{b+1} \tag{10}$$

Where, a is denoted as the particle number in the total population, b is denoted as the iteration number, and (i,j) is denoted as coordinates number in x and y axis, respectively.

Table 1 shows the parameters of CPSO that will be used in the simulation results.

Table 1. The choice of a parameter was considered to be the optimal choice.

Parameter	Definition with value
β^0	The initial value of deterministic β = 0.3
μ	The control parameter with a real value μ = 4
W	Inertia Weight
W_{min}	Minimum W = 0.3
W_{max}	Maximum W = 0.9
$iter$	Current iteration number (b)
$iter_{max}$	Maximum number of iterations
c_1, c_2	Coefficient of acceleration (1.25, 1.25)
V_a^b	The velocity of particle ath in iteration bth
$(xy)_a^b$	The position of particle ath in iteration bth
pbest_a	Best fitness values for particle ath
Gbest	Best fitness values for the whole swarm

```
Basic CPSO Procedure:
Step 1: Maximum iterations.
Step 2: Initialize particle.
Step 3: Each particle, checking fitness value, if the fitness value is better than the best fitness value (pbest) then set current value as new pbest.
Step 4: Each particle
    - Find the particle with the best fitness (gbest) in the particle neighbourhood.
    - Apply chaotic optimization algorithm eq. (6, 7, and 8).
    - According to the velocity equation (9) calculate particle velocity $v(i,j)$.
    - Apply the new velocity.
    - According to the position equation (10), update the particle position $xy(i,j)$.
    - Apply the new position.
Step 5: Repeat Step 3 until reach maximum iterations.
```

Figure 5. The proposed pseudocode of the CPSO algorithm.

3.1.3. Hybrid swarm optimization algorithm

In this work, the proposed hybrid swarm optimization algorithm is used to find the optimal or near optimal desired path for the mobile robot. It combined A-star algorithm with CPSO algorithm in order to find the shortest path to reach the target point

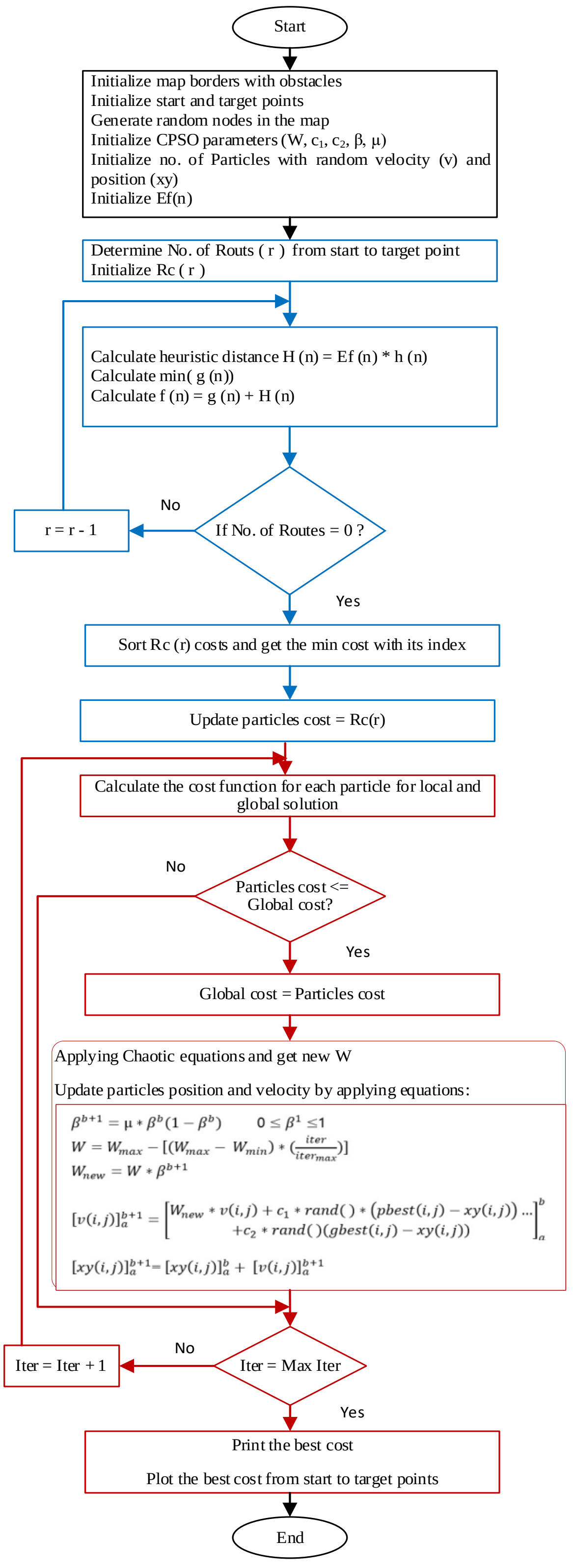

Start
Initialize map borders with obstacles
Initialize start and target points
Generate random nodes in the map
Initialize CPSO parameters (W, c1, c2, β, μ)
Initialize no. of Particles with random velocity (v) and position (xy)
Initialize Ef(n)
Determine No. of Routs (r) from start to target point
Initialize Rc (r)
Calculate heuristic distance H (n) = Ef (n) * h (n)
Calculate min(g (n))
Calculate f (n) = g (n) + H (n)
No
r = r - 1
If No. of Routes = 0 ?
Yes
Sort Rc (r) costs and get the min cost with its index
Update particles cost = Rc(r)
Calculate the cost function for each particle for local and global solution
No
Particles cost <= Global cost?
Yes
Global cost = Particles cost
Applying Chaotic equations and get new W
Update particles position and velocity by applying equations:
$\beta^{b+1} = \mu * \beta^b(1-\beta^b) \quad 0 \le \beta^1 \le 1$
$W = W_{max} - [(W_{max} - W_{min}) * (\frac{iter}{iter_{max}})]$
$W_{new} = W * \beta^{b+1}$
$[v(i,j)]_a^{b+1} = \begin{bmatrix} W_{new} * v(i,j) + c_1 * rand() * (pbest(i,j) - xy(i,j)) \dots \\ +c_2 * rand()(gbest(i,j) - xy(i,j)) \end{bmatrix}_a^b$
$[xy(i,j)]_a^{b+1} = [xy(i,j)]_a^b + [v(i,j)]_a^{b+1}$
No
Iter = Iter + 1
Iter = Max Iter
Yes
Print the best cost
Plot the best cost from start to target points
End

Figure. 6 The proposed hybrid ACPSO flowchart

Received: April 4, 2021. Revised: April 20, 2021.

in less time. The proposed hybrid flowchart algorithm is shown in Fig. 6. So, the steps of proposed hybrid ACPSO algorithm as follows:

Step1: Initialization step: generate random nodes (n) and find all possible routes (r) from source to destination point, then calculate the cost distance function for each route with A-star cost distance function as shows in Eq. (11) and (12).

$$H(n) = Ef(n) \times h(n) \qquad (11)$$

$$f(n) = g(n) + H(n) \qquad (12)$$

Where, $H(n)$ is denoted the enhancement heuristic function. $Ef(n)$ is denoted the enhancement random factor < 1.

Step2: Proposed a dynamic weight that is called enhanced factor (Ef) in the heuristic function which can minimize area search, where Ef is a random number less than 1, then save the routes cost into matrix called (Rc), after that sorting the matrix (Rc) to find the lowest cost and obtain its index; Finally, the costs of the (Rc) will be saved into the initialized particles and will evaluate the local and global cost distance function for best solution in the CPSO algorithm.

Step3: Iteration step: after getting the minimum global cost distance function, the main iteration starts and enforce all particles to update their velocity and position according to global cost distance function till end of the iteration.

3.2 Control strategy design

The proposed control strategy in this work is to solve the second problem that will encounter in the designed motion controller of trajectory tracking for mobile robot because the mobile robot platform has the highly nonlinear kinematics model and time variant outputs states as well as it has under-actuated system. Therefore, the proposed controller has ability for generating precisely and quickly the optimal left and right wheels velocities in order to track the desired paths equations with minimum tracking position error and without oscillation. The general structure of the on-line tuning control strategy of trajectory tracking for mobile robot is shown in Fig. 7.

So, the proposed structure of the Convolutional Neural Networks Trajectory Tracking (CNNTT) Controller for mobile robot is shown in Fig. 8.

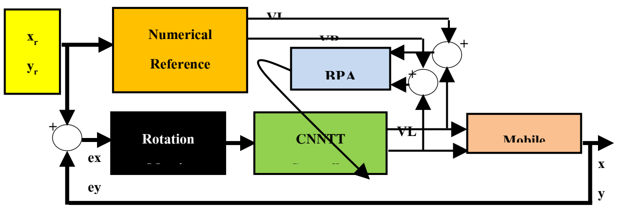


Figure. 7 General structure of on-line tuning control strategy

In this work, the control strategy is based on convolutional neural networks with back propagation algorithm that is used to learn the proposed trajectory tracking controller for mobile robot. Moreover, the off-line learning steps of proposed CNNTT controller can be described as follows:

Step1: Input Layer; the input layer of the CNNTT controller takes the extracted matrix from the optimal path equation (x_r, y_r, θ_r) and takes the values of the (VL_{ref}, VR_{ref}) from numerical reference velocities equation. So, the size of the data set can be described as two-dimensional matrix (I, J). Then the data set are divided into two sets, the first set is call learning set, and the second is called testing set.

Step2: Convolution Layer: a series of learnable filters make up the convolutional layer. These filters have a limited spatial size. The user selects the number of filters, with each filter learning to scan for a specific function in the current receptive region. We suggested a set of (F) number of filters that slide over the input matrix and produce 2D matrix as (I, F). To minimize the number of weights that will use to detect a pose of the mobile robot.

Step3: Bipolar Sigmoid Layer: The convolution layer's output is transferred through the sigmoid layer, which adds a non-linearity equation into the network and produces a stack of 2D activation arrays (I, F) with the relevant features in the original path matrix.

Step4: Pooling Layer: The pooling layer is used to minimize the number of weights in the model and can also assist with overfitting. The pooling process is carried out on each of the depth slices individually. In this work, we use the S that denoted to the length of the vectors so the new matrix will generate as (S, F).
Step5: Fully Connected Layer: after pooling layer, the matrix (S, F) will collect in one flat vector (S×F, 1) that is called fully connected network data set in order to learn the neural networks which consists of three layers (input layer, hidden layer and output layer) and it is used sigmoid activation function in the

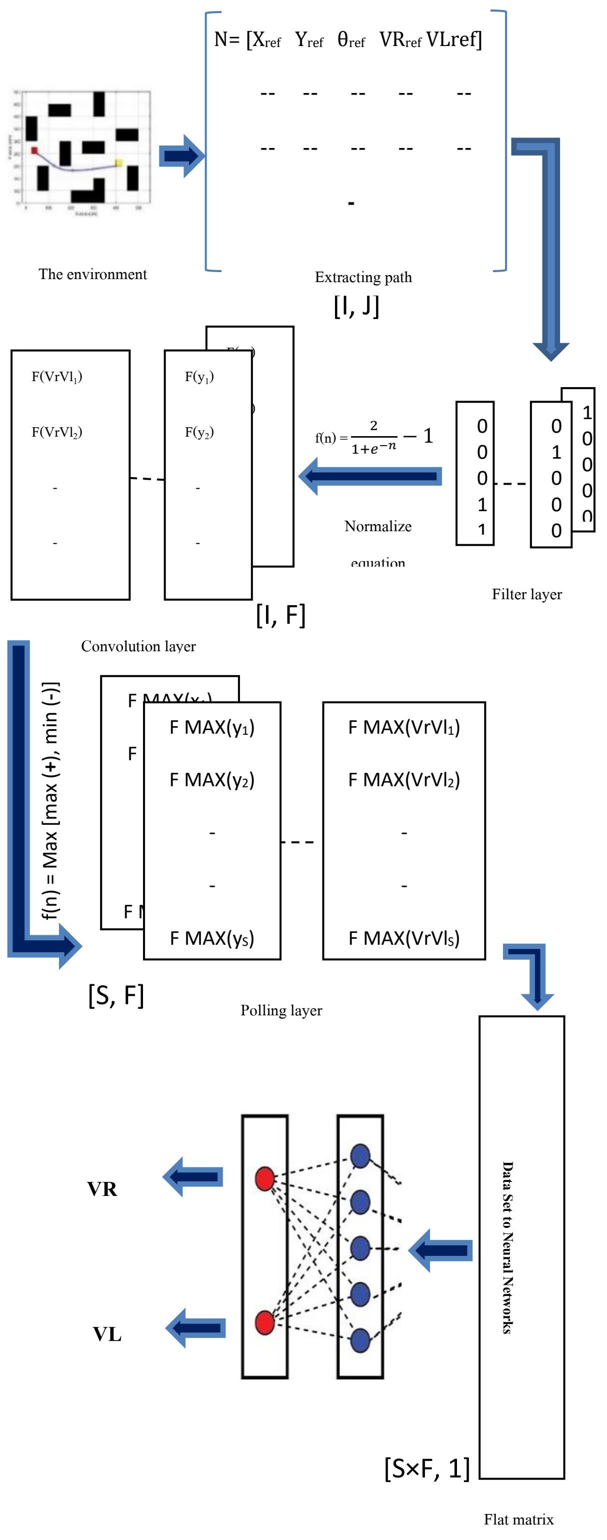


Figure. 8 The proposed structure of the (CNNTT) Controller for mobile robot

hidden layer and linear activation function in the output layer.

Step6: Back propagation algorithm is used for learning the neural network with mean square error cost function is used to determine the outputs neural network (left and right wheels velocities) are approximated equal to the reference velocity or not in order to update the weights of the CNNTT controller and increasing the number of epochs.

Step7: To investigate the neural network are excellent learned, a testing set is used and show the difference between the reference velocities and the left and right wheels velocities for the mobile robot.

4. Simulation results

The path-planning and control strategy are shown in Fig. 2 which are consisted of two steps, the first step is generated desired path with short length and free-navigation. The second step is generated best control action for the kinematics mobile robot model that is considered as under-actuated system with two inputs (left and right wheels velocities) and three states outputs (position (x,y) and orientation θ) with highly nonlinear and time variant behaviour system. The cart specification of the mobile robot has radius of wheel is 0.075 m and distance between wheels is 0.39 m wheel diameter. MATLAB (file.m) package is used to carry out the proposed strategy and solve numerical simulation kinematics mobile robot model as in Eq. (1) to Eq. (3) with sampling time is equal to 0. 1 sec. To investigate the efficiency of the proposed strategy at fixed obstacles environment with workspace [500 by 500] cm, as shown in Fig. 9 and to solve the problem definition of the free-navigation mobile robot as well as swiftly track the desired path with minimum position and orientation errors, and no oscillation in the output states of the mobile robot.

Different cases are used to confirm the mobile robot in both desired (position and orientation) are followed. The environment is populated by static obstacles and full information about the positions of all objects in the workspace are available. The task of finding this collision-free path is the responsibility of a path-generating algorithm, that we apply three

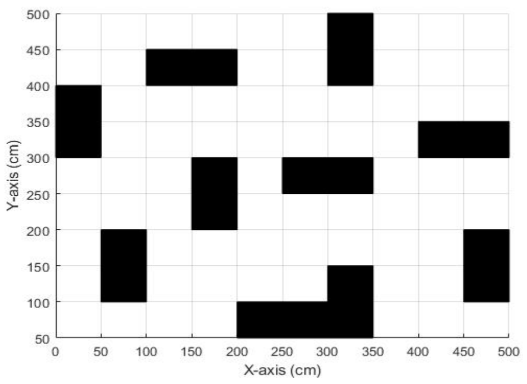


Figure. 9 The proposed environment with static obstacles

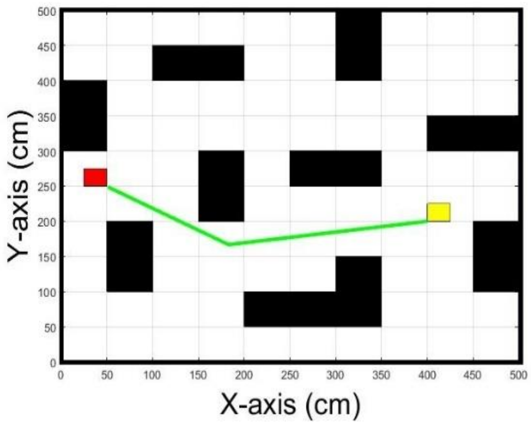
algorithms (A-star, CPSO, and Hybrid ACPSO) and compares them to find the optimum path with the best cost distance function and using computer which has specification as Intel Core i5 – 2450 M CPU 2.5 GHz with 8.00 GB RAM.

Case 1:

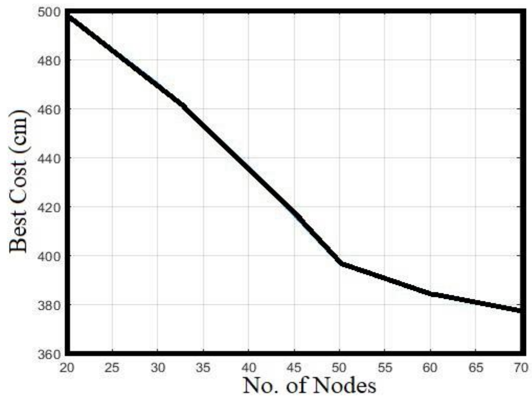
The initial position (50, 250) cm (red point) to the destination point (400, 200) cm (yellow point) as shown in Fig. 10-a when it applies A-star algorithm and obtained the distance cost function is equal to 374 cm as shows in Fig. 10 (b).

Secondly, applying CPSO algorithm to find the shortest path in known environments and using the number of iterations is 30 as shown in Fig. 11 (a). Then the cost distance function based on CPSO algorithm is obtained 363 cm as shown in Fig. 11 (b).

Thirdly, applying the proposed hybrid ACPSO algorithm to find the shortest path in known environments as shown in Fig. 9 with the number of iterations is equal to 15 as shown in Fig. 12 (a). The value of the proposed hybrid ACPSO cost distance function is equal to 357 cm as shown in Fig. 12 (b).

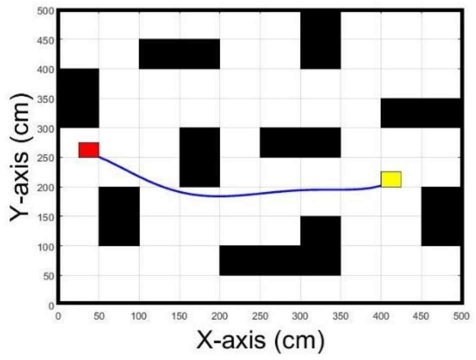


(a)

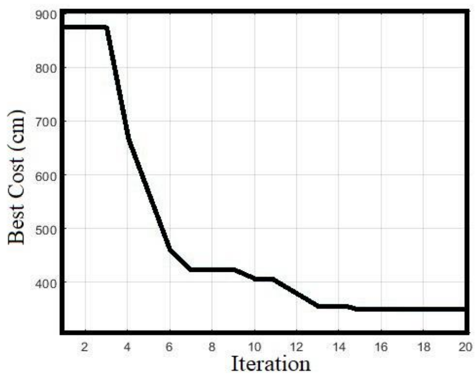


(b)

Figure. 10 The A-star algorithm for case 1: (a) path planning and (b) best distance cost function

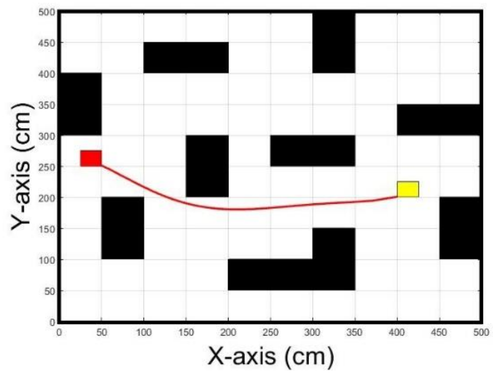


(a)

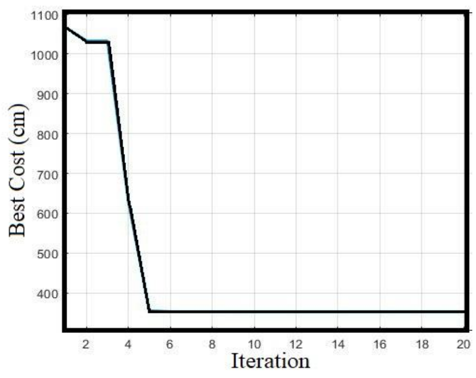


(b)

Figure. 11 The CPSO algorithm for case 1: (a) path planning and (b) best distance cost function



(a)



(b)

Figure. 12 The proposed hybrid ACPSO algorithm case 1: (a) path planning and (b) best distance cost function

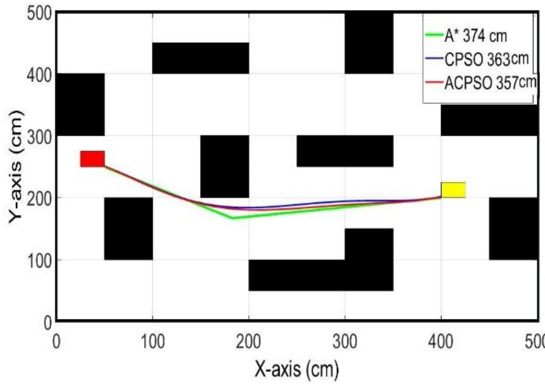


Figure. 13 All path planning in case 1

Table 2. A-star algorithm results

No. of Nodes	Nodes of the Route	Cost (cm)	Spent time (sec)
20	4	498	0.956
30	4	470	1.359
40	4	436	1.794
50	3	397	3.951
60	3	384	5.578
70	**3**	**374**	7.782

Table 3. CPSO algorithm results

Iteration	Cost (cm)	Spent time (sec)
5	495	1.502
10	453	2.608
15	377	3.775
20	373	5.070
25	370	6.272
30	**363**	7.377

Table 4. ACPSO algorithm results

Iteration	Nodes of the Route	Cost (cm)	Spent time (sec)
5	3	470	2.747
10	3	406	3.829
15	3	**357**	4.985
20	**3**	**357**	-
25	**3**	**357**	-
30	**3**	**357**	-

In addition, when comparing between A-star, CPSO and ACPSO algorithms as in Fig. 13 and Tables 2, 3 and 4 respectivtly.

We show the hybrid ACPSO algorithm is much better than A-star and CPSO Algorithms in terms of the number of nodes is only three, the number of the iterations is fifteen, path length is 357 cm and the execution time is 4.98 sec because of the hybrid ACPSO algorithm is used the best nodes that are generated from A-star algorithm and used these nodes as initial values of the particles in the CPSO algorithm with Spline interpolation technique is used to find the short and smooth desired path.

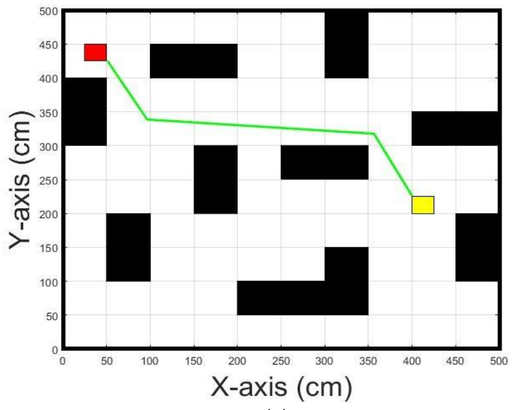
Case 2:

The initial position of the mobile robot at (50, 425) cm (red point) to the destination point (400, 225) cm (yellow point) as shown in Fig. 14 (a). Applying A-star algorithm and the value of the cost distance function is 459 cm, as shows in Fig. 14 (b).

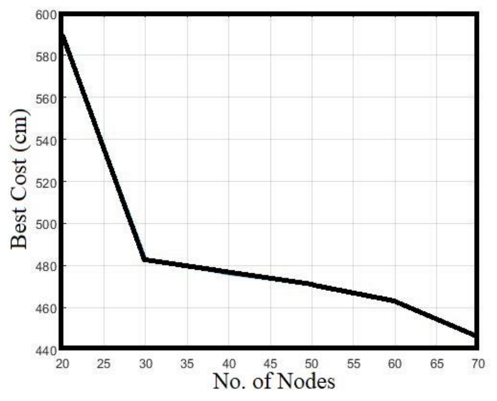
Secondly, Fig. 15 (a) shows the path planning of the mobile robot after applying CPSO algorithm. We found CPSO cost distance function is equal to 443 cm, as shown in Fig. 15 (b).

Finally, applying hybrid ACPSO algorithm to the same environment in case 2, the path planning is generated as shown in Fig. 16 (a) with cost distance function is equal to 387 cm, as shown in Fig. 16 (b).

In order to invistigate the effectiveness of the hybrid swarm algorithm, the three algorithms (A-star, CPSO and ACPSO) are applied to the environment case 2 as shown in the Fig. 17. The path planning that is generated from the proposed hybrid algorithm was smooth and the shortest path from strating point to target point when we comparted with A-star and CPSO algorthims.

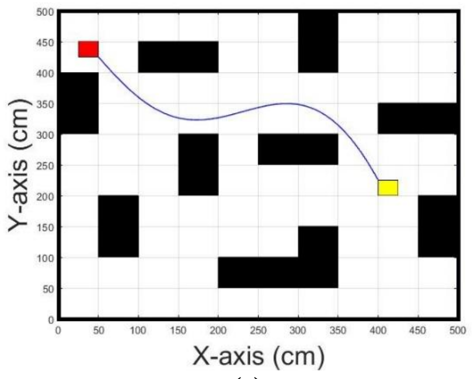


(a)



(b)

Figure. 14 The A-star algorithm for case 2: (a) path planning and (b) best distance cost function


(a)

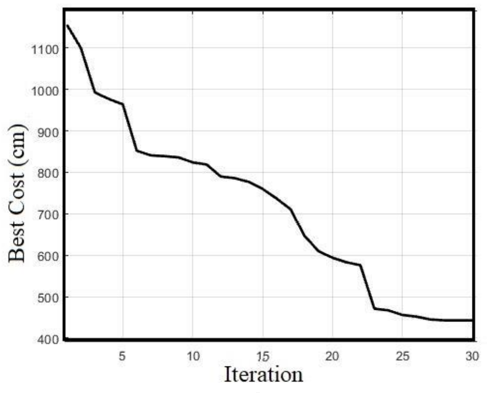
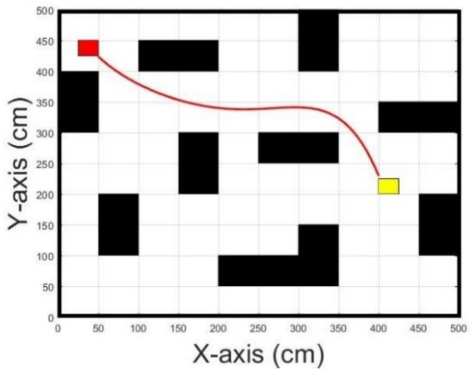

(b)

Figure. 15 The CPSO algorithm for case 2: (a) path planning and (b) best distance cost function


(a)

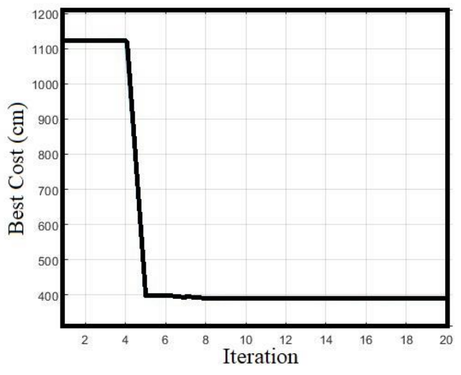

(b)

Figure. 16 The proposed hybrid ACPSO algorithm case 2: (a) path planning and (b) best distance cost function

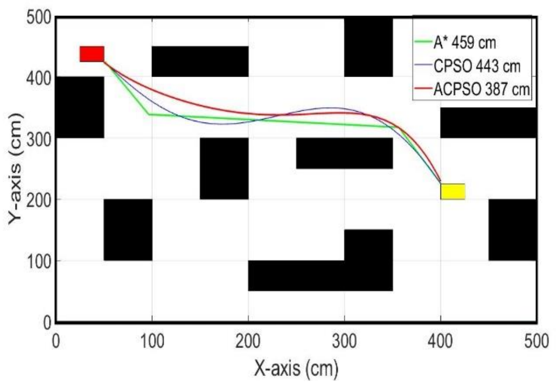


Figure. 17 All path planning in case 2.

Table 5. A-star algorithm results

No. of Nodes	Nodes of the Route	Cost (cm)	Spent time (sec)
20	5	590	1.453
30	4	483	1.879
40	4	476	3.458
50	4	471	5.781
60	4	468	6.673
70	**4**	**459**	8.256

Table 6. CPSO algorithm results

Iteration	Cost (cm)	Spent time (sec)
5	2504	1.482
10	977	2.616
15	847	3.790
20	760	4.911
25	550	6.335
30	**443**	7.82001

Table 7. ACPSO algorithm results

Iteration	Nodes of the Route	Cost (cm)	Spent time (sec)
5	4	530	2.364
10	4	420	3.516
15	4	389	5.050
20	**4**	**387**	6.230
25	**4**	**387**	-
30	**4**	**387**	-

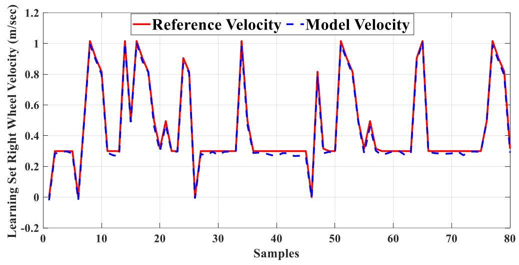
In addition, when comparing between A-star, CPSO and ACPSO algorithms as in Fig. 17 and Tables 5, 6 and 7 respectivtly, we show the hybrid ACPSO algorithm is much better than A-star and CPSO Algorithms in terms of the number of nodes is only four, the number of the iterations is twenty, path length is 387 cm and the execution time is 6.23 sec.

Based on the fitting function, we obtained the reference path equation as in Eq. (13) and (14) for the case 1 and case 2 respectivitly, as the optimal path that depends on the hybrid ACPSO algorithm.

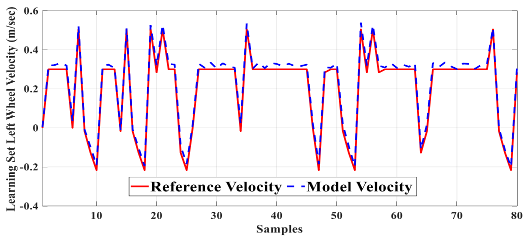
$$y(x) = 2.085e-10 \times x5 - 2.373e-07 \times x^4 + 9.516e-05 \times x^3 - 0.014 \times x^2 + 0.198 \times x + 266.410 \quad (13)$$

$$y(x) = -3.115e-10 \times x5 + 2.697e-07 \times x^4 - 9.276e-05 \times x^3 + 0.018 \times x^2 - 2.455 \times x + 510.797 \quad (14)$$

Then applying the proposed structure as in Fig. 8 on the maps based on desired paths as in Eq. (13) and (14), the two-dimensional matrix (I, J) is equal to (160, 5) respectively, with sixteen proposed filters (F=16) for obtaining the data set matrix is (160, 16) in the convolution layer then divided into two sets; learning set matrix is (80, 16) and testing set matrix is also (80, 16). In the pooling layer where S is equal to 5 (using only five maximum values in the matrix (80, 16)) so it becomes data matrix (5, 16) than in the fully connected layer the data set becomes as only one vector (5×16, 1) then carrying out the proposed CNNTT controller as in Fig. 7 that consists of [5:11:2] nodes as follows: five nodes in the input layer and one hidden layer that has eleven nodes with nonlinear activation function, and the output layer has two nodes with linear activation function. Based on Back Propagation Algorithm (BPA) and after 1000 epoch, the model CNNTT controller has ability to generate left and right wheels velocities as the same reference velocity as shown in Fig. 18-a and b.



(a)



(b)

Figure. 18 Learning set: (a) righ wheel velocity and (b) left wheel velocity

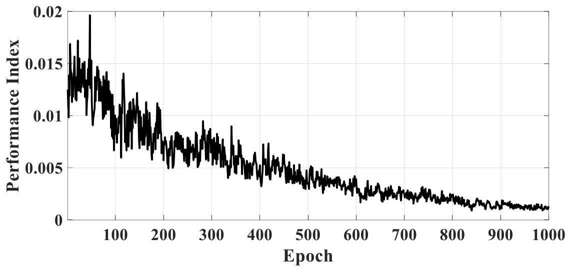


Figure. 19 best response of performance index for the CNNTT controller in the learning cycle.

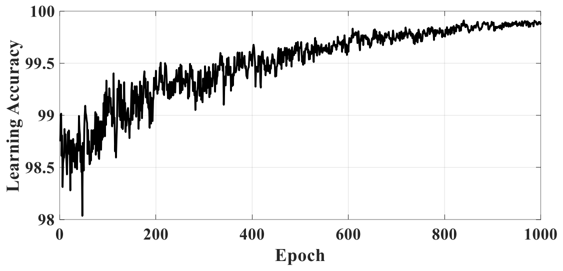
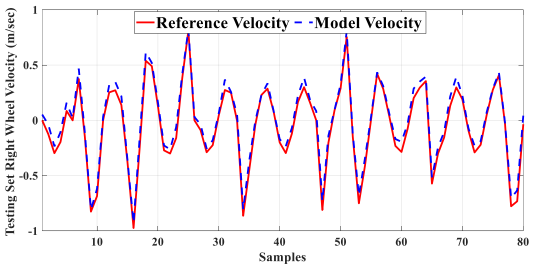


Figure. 20 The accuracy of the CNNTT controller learned.

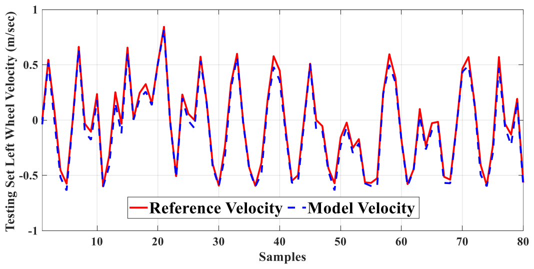
Figure 19 shows the response of the performance index of the learning of the CNNTT controller during 1000 epoch which it reaches to small value that it is less than 0.005.

To show the accuracy of the learning the CNNTT controller that reaches to over 99.5% at 1000 epoch an in Fig. 20.

To confirm the proposed controller has excellent learning in the all regions based on the desired paths in the two cases. Another set of the data (testing set)



(a)



(b)

Figure. 21 Testing set: (a) righ wheel velocity (b) left wheel velocity

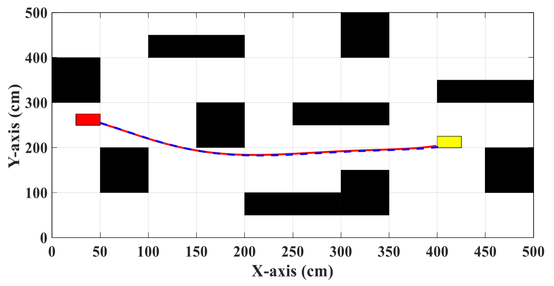


Figure. 22 The 2D simulation desired path case 1 where (red line) and actual mobile robot path (dash blue line)

one vector (80,1) is applied to the CNNTT controller model to demonstrate the left and right wheels velocities set as in Fig. 21 a and b, respectively. These responses of left and right wheels velocities of the CNNTT controller outputs have very small error w.r.t the reference velocities that means the learning set has rich input signal to excite all regions of the proposed controller model and did not occur the over-learning problem during 1000 epoch then the proposed controller is ready to track the different types of the desired paths.

In case I, the mobile robot has initial position (x_o, y_o) = (50, 250) cm respectively. Fig. 22 shows 2D simulation desired path based on Eq. (12) as in red colour with the output of the kinematics mobile robot model outputs (x, y) at blue colour.

The actual output of the mobile robot (x, y) has fast and without oscillation during 350 sample for tracking the desired path with minimum errors in the x-position 4 cm and y-position 1.5 cm in Figs. 23 (a) and (b) as well as the orientation error of the mobile robot as in Fig. 23 (c).

The output response of the proposed CNNTT controller is shown in Fig. 24 that represents the fast and smooth control action responses of left and right wheels linear velocities of the mobile robot and did not exceed 1 m/sec and no saturation state that makes swiftly and successfully desired path tracking. Fig. 25 shows the best response of the linear velocity and angular velocity of the platform mobile robot.

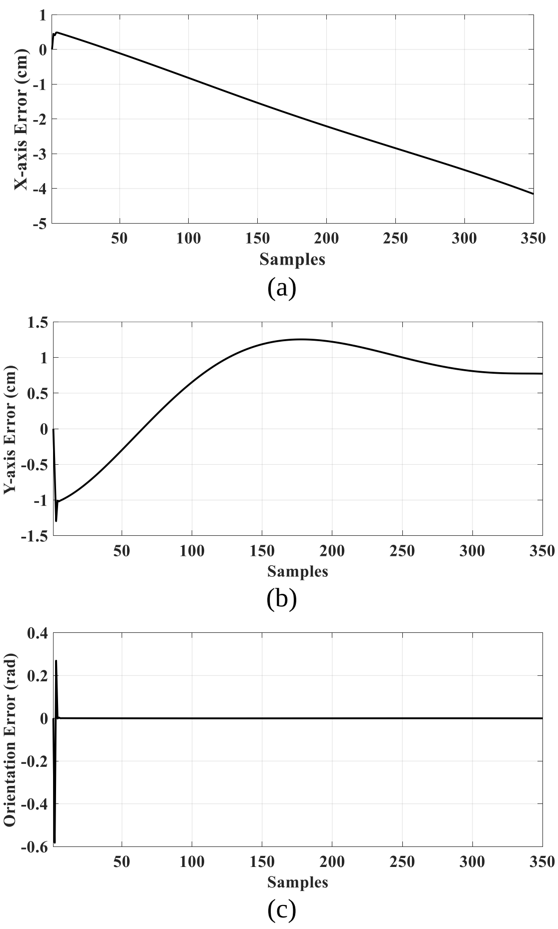


Figure. 23 The response of the tracking error during 350 sample in case 1: (a) in the x-axis, (b) in the y-axis, and (c) orientation error

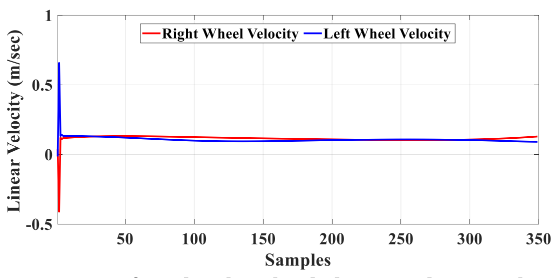


Figure. 24 Left and right wheels linear velocities that are generated from the proposed CNNTT controller in case 1

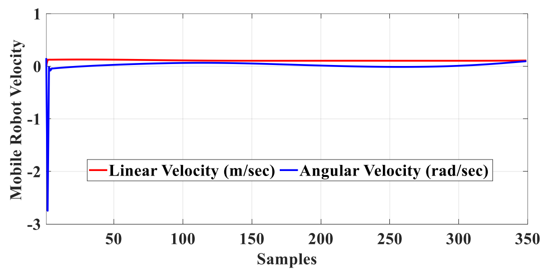


Figure. 25 linear velocity and angular velocity of the platform mobile robot in case 1

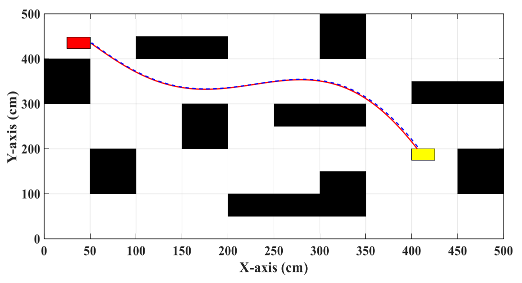
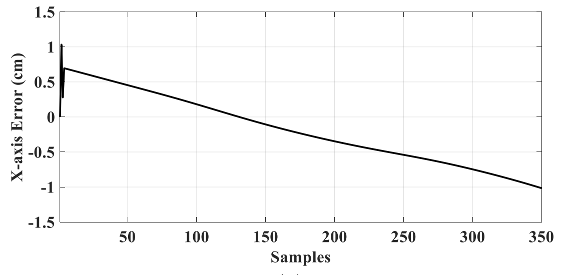


Figure. 26 The 2D simulation desired path case 2 where (red line) and actual mobile robot path (dash blue line)

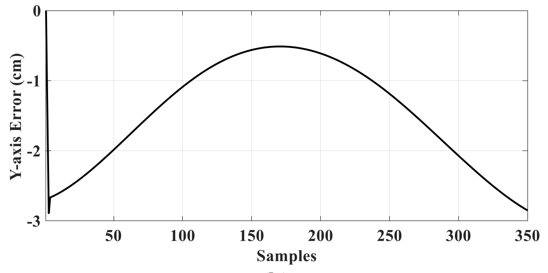
The actual output of the mobile robot (x, y) has fast and without oscillation during 350 sample for tracking the desired path with minimum errors in the x-position and y-position are equal 1.5 cm and 3 cm respectively as in Fig. 27 (a) and b. As well as the orientation of the mobile robot as in Fig. 27 (c).

Fig. 28 shows the fast and smooth control action responses that generated from the proposed CNNTT controller as the left and right wheels linear velocities of the mobile robot. As well as the maximum value of the linear wheel velocity did not exceed 1 m/sec and no saturation state that makes pure-rolling and non-slipping for tracking the desired path. The swift response of linear and angular velocity of the platform mobile robot during 350 sample is shown in Fig. 29.

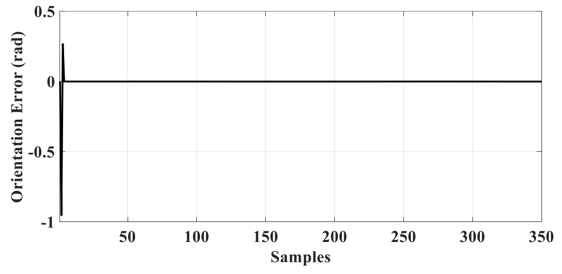
To confirm the effectiveness of this proposed work, we are compared the numerical simulation results of the proposed CNNTT controller with other types of controller results that are taken from [11, 14, 16] as shown in Table 8 in terms of the maximum tracking error in the position (x-axis and y-axis) of



(a)



(b)



(c)

Figure. 27 The response of the tracking error during 350 sample in case 2: (a) in the x-axis, (b) in the y-axis, and (c) orientation error.

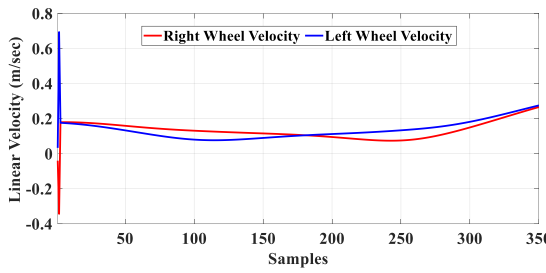


Figure. 28 Left and right wheels linear velocities that are generated from the proposed CNNTT controller in case 2

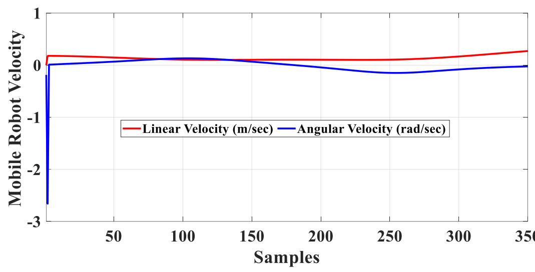


Figure. 29 linear velocity and angular velocity of the platform mobile robot in case 2

Table 8. Comparison simulation results between the proposed CNNTT controller and other controller types

Type of Controller	**Type of Performance Index with minimum value obtained**	**Maximum Error in x-axis obtained**	**Maximum Error in y-axis obtained**
[11] Nonlinear NN like PID controller	On-Line MSE=0.009	6 cm	2.5 cm
[14] Nonlinear Predictive Controller	Off-Line MSE=0.035	5 cm	4 cm
[16] Nonlinear Back Stepping Controller	On-Line MSE=14	5 cm	± 2.5 cm
Proposed Controller	Off-Line MSE= 0.005	4 cm	2.5 cm

the path tracking for mobile robot was obtained as well as the minimum value of the MSE performance index was obtained for learning these controllers.

5. Conclusions

In this paper, an off-line and on-line tuning CNNTT controller has been designed and simulated for mobile robot system and tracking desired path equation that generated from proposed hybrid swarm optimization ACPSO algorithm using MATLAB simulation package. The mobile robot platform has highly nonlinear multi-input multi-output system, therefore, the proposed path planning and control strategy has excellent ability for solving the problem statement of the path planning and trajectory tracking for mobile robot in term of the following:

- Optimal or near optimal smooth desried path equation is generated based on hybrid swarm ACPSO algorithm.
- Reducing the path length, the number of the iterations, the evaluation function and the execution time of the processor unit during generated the desired path.
- Best and smooth value of left and right wheel's velocities control actions that were generated for under-actuated system states and nonlinear kinematics behaviour of the mobile robot platform model.
- The mobile robot is excellent tracking the desired path and reached the target point successful without oscillation.
- The maximum span tracking pose error reached approximately 4 cm in x-axis and 2.5 cm in y-axis.
- The on-line tuning weights of the proposed CNNTT controller based on BPA leads to generate smooth velocity actions without spikes and no saturation state in each wheels or platform velocities that make precision tracking of the desired path equation.

So, we hope the experimental works of the proposed path planning algorithm and control strategy for mobile robot system and will be implemented in the future work.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Omar Abdul Razzaq Abdul Wahhab and Ahmed Sabah Al-Araji developed convolutional neural network trajectory tracking controller for mobile robot model and verified the proposed controller. Omar Abdul Razzaq Abdul Wahhab explained the swarm optimization algorithms for path planning. Ahmed Sabah Al-Araji presented the kinematics mobile model. Both authors discussed the numerical simulation results contributed to the final work.

References

[1] J-C. Liao, S-H. Chen, Z-Y. Zhuang, B-W. Wu, and Y-J. Chen, "Designing and Manufacturing of Automatic Robotic Lawn Mower", *Processes*, Vol. 9, No. 358, pp. 1-21, 2021.

[2] R. Dikairono, S. D. Purwanto, and T. A. Sardjono, "CNN-Based Self Localization Using Visual Modelling of a Gyrocompass Line Mark and Omni-Vision Image for a Wheeled Soccer Robot Application", *International Journal of Intelligent Engineering & Systems*, Vol. 13, No. 6, pp. 442-453, 2020.

[3] S. K. Malu and J. Majumdar, "Kinematics, Localization and Control of Differential Drive Mobile Robot". *Global Journal of Researches in Engineering: Robotics & Nano-Tech*, Vol. 14, No. 1, pp. 1-7, 2014.

[4] D. Di Paola, A. Milella, G. Cicirelli, and A. Distante, "An Autonomous Mobile Robotic System for Surveillance of Indoor Environments", *International Journal of Advanced Robotic Systems (IJARS)*, Vol. 7, pp. 19-26, 2010.

[5] C. C. Tseng, C. L. Lin, B. Y. Shih, C. Y. Chen, "SIP-Enabled Surveillance Patrol Robot", *Robotic and Computer-Integrated Manufacturing*, Vol. 29, pp. 394–399, 2013.

[6] Y. L. Liao, and K. L. Su, "Multi-Robot-Based Intelligent Security System", *Artificial Life* and *Robotics*, Vol. 16, pp. 137–141, 2011.

[7] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An Improved Ant Colony Algorithm for Robot Path Planning", *Soft computing*, Vol. 21, No. 19, pp. 5829–5839, 2017.

[8] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz, and N. Pavón, "Solving the Multi-Objective Path Planning Problem in Mobile Robotics with a Firefly-Based Approach", *Soft computing*, Vol. 21, No. 4, pp. 949–964, 2017.

[9] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile Robot Path Planning Using Artificial Bee Colony and Evolutionary Programming", *Applied Soft Computing*, Vol. 30, pp. 319-328, 2015.

[10] N. A. Daniel, A. A. Gallegos, C. Lopez-Franco, and A. Y. Alanis, "Smooth Global and Local Path Planning for Mobile Robot Using Particle Swarm Optimization, Radial Basis Functions, Splines and Bézier Curves", In: *Proc. of International Conf. On Evolutionary Computation (CEC)*, pp. 175-182, 2014.

[11] S. Al-Araji, A. K. Ahmed, and K. E. Dagher, "A Cognition Path Planning with a Nonlinear Controller Design for Wheeled Mobile Robot Based on an Intelligent Algorithm", *Journal of Engineering*, Vol. 25, No. 1, pp. 64-83, 2019.

[12] R. k. Panda and B.B. Choudhury "An Effective Path Planning of Mobile Robot Using Genetic

Algorithm", In: *Proc. of International Conf. On Computational Intelligence & Communication Technology*, pp. 287-291, 2015.

[13] M. Duguleana and G. Mogan, "Neural Networks-Based Reinforcement Learning for Mobile Robot's Obstacle Avoidance", *Expert Systems with Applications*, Vol. 62, pp. 104–115, 2016.

[14] A. S. Al-Araji, M. F. Abbod, H. S. Al-Raweshidy, "Applying Posture Identifier in Designing an Adaptive Nonlinear Predictive Controller for Nonholonomic Mobile Robot", *Neurocomputing*, Vol. 99, pp. 543-554, 2013.

[15] M. Asif, M. J. Khan, M. Rehan, and M. Safwan," Feedforward and Feedback Kinematics Controller for Wheeled Mobile robot Trajectory Tracking", *Journal of Automation and Control Engineering*, Vol. 3, No. 3, pp. 178–182, 2015.

[16] S. Ghosh, P. P. Kumar, and D. R. Parhi, "Performance Comparison of Novel WNN Approach with RBFNN in the Navigation of Autonomous Mobile Robotic Agent", *Serbian journal of electrical engineering*, Vol. 13, No. 2, pp. 239–263, 2016.

[17] A. Al-Araji, "Development of Kinematic Path-Tracking Controller Design for Real Mobile Robot via Back-stepping Slice Genetic Robust Algorithm Technique", Arabian Journal for Science and Engineering, Vol. 39, No. 12, pp. 8825-8835, 2014.

[18] M. Fuad, T. Agustinah, and D. Purwanto, "Collision Avoidance of Multi Modal Moving Objects for Mobile Robot Using Hybrid Velocity Obstacles", *International Journal of Intelligent Engineering & Systems*, Vol. 13, No. 3, pp. 407-421, 2020.

[19] A. K. Patil, S. S. Patil, P. Manickam, "Identification of Lung Cancer Related Genes Using Enhanced Floyd Warshall Algorithm in a Protein-to-Protein Interaction Network", *International Journal of Intelligent Engineering & Systems*, Vol. 11, No. 3, pp. 215-222, 2018.

[20] S. Ahmed, R. F. Ibrahim, and H. A. Hefny, "Mobile-Based Routes Network Analysis for Emergency Response Using an Enhanced Dijkstra's Algorithm and AHP". *International Journal of Intelligent Engineering & Systems*, Vol. 11, No. 6, pp. 252-260, 2018.

[21] S. M. Persson and I. Sharf, "Sampling-Based A* Algorithm for Robot Path-Planning", *International Journal of Robotics Research*, Vol. 33, No. 13, pp. 1683-1708, 2014.

[22] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path Planning with Modified A Star Algorithm for a Mobile Robot", *Procedia Engineering*, Vol. 96, pp. 59-69, 2014.

[23] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", In: *Proc. of International Conf. On Neural Networks*, pp. 1942-1948, 1995.

[24] T. T. Mac, C. Copot, D. T. Tran, R. Keyser, "Heuristic Approaches in Robot Path Planning: A survey", *Robotics and Autonomous Systems*, Vol. 86, pp. 13–28, 2016.

[25] H. N. Abdullah, "An Improvement in LQR Controller Design based on Modified Chaotic Particle Swarm Optimization and Model Order Reduction", *International Journal of Intelligent Engineering & Systems*, Vol. 14, No. 1, pp. 157-168, 2021.