International Journal of Computers and Informatics (IJCI)
Vol. (5), No. (8)
IJCI
August 2026
المجلة الدولية للحاسبات والمعلوماتية
الإصدار (5)، العدد (8)

Obstacle Avoidance by a Mobile Robot Using a Nuro-Genetic Controller System

Tefool H. O. Al-khafaji
College of Education for Pure Sciences, University of Babylon, Babylon, Iraq
pure.tefool.hussein@uobabylon.edu.iq

Ali Hasan Shaheed
College of Science for Women, University of Babylon, Babylon, Iraq
wsci.ali.hasan@uobabylon.edu.iq

Inas Imad Kadhim
College of Education for Pure Sciences, University of Babylon, Babylon, Iraq
edu622.enass.emad@uobabylon.edu.iq

Abstract

By applying Darwinian principles like natural selection and survival of the fittest, neural controllers can be developed that enable robots to act autonomously in unpredictable or unknown environments. This study examines the Genetic Algorithm (GA) as an optimization tool for mobile robot navigation. The controller's performance was evaluated on a standard obstacle-avoidance task, as well as a multi-task setup combining obstacle avoidance with homing. Experimental results confirm that neuro-genetic controller optimization significantly enhances both average population fitness and individual peak fitness, while demonstrating the feasibility of integrating cooperative functional modules to accomplish complex robotic goals.

Keywords: Neuro -Genetic Controller, Mobile Robots, Obstacle Avoidance.

1. Introduction and Literature Review

The main objective of the field of mobile robotics is to build general purpose intelligent robot systems that can adapt to their environments and successfully perform complex tasks in unpredictable and changeable environments. Adaptation is acquired through learning to behave correctly in different situations, and to cope with changes in the environment. Learning can be made using any known learning algorithm.

The field of mobile robotics has numerous contributions in the literature. All these contributions are tried to build either a new direction in building robot control systems or trying to enhance some aspects of control systems that previously have been proposed. There are examples to approaches used explicitly written programs to control a robot (see for example [1] [2]). It will be focused in this review on the controllers that exploiting learning on “Behavior Based Robotics” (robots that

concentrating on learning simple basic behaviors and learning to coordinate these behaviors to achieve a whole complex adaptive behavior).

The hybridization of Genetic Algorithms (GAs) with Artificial Neural Networks (ANNs) is mainly depends on evolving ANNs to act as controllers to a mobile robot. GAs are used to evolve appropriate internal parameters of robot neuro-controller. Each string in the population represents the parameters defining a neuro-controller for the robot. It is necessary to define a set of decoding rules for mapping the genotypic string into the neural network phenotype.

A series of experiments have been made at the endings of the 20th century and the beginnings of the 21st century to demonstrate the validation of Neuro-Genetic controller systems. These experiments concentrated on different aspects of neuro-genetic robotic systems. For example:

1. Evolving neuro-controller systems for navigation and performing tasks [3], [4].
2. Adapting controllers to different changes in environment [5], [6], [7] [8].
3. Studying the capability of evolutionary system to tackle complex tasks [9].
4. Presenting modular system architectures [10], [11].
5. Studying perception aspect in evolutionary system [12].
6. Researching the competition in co-evolutionary robotics [13].
7. Integration of sensory-motor information over time [14].
8. Investigating the evolution of language in evolutionary robotics [15].

For more details about evolutionary robotics see [16].

A more recent studies that combine GAs and ANNs like [17] that used a controller aided by a low computational cost vision system. The controller uses the vision system and the ANN to detect and recognize obstacles found in the robot's path. If the object is in the controller's knowledge bank a previously registered deviation solution is applied. Otherwise, the GA must optimize a new route alternative. Also, in [18], the GA controller is fed by the obstacles' distances and the turning angle generated by the sensory network of humanoid robot to generate the next best feasible solution as the initial output. The initial output from GA and the obstacles' distances are the input to 4-layer ANN that provide the desired output. The desired output represents the turning angle for the robot.

GAs are integrated with Deep Learning (DL) for mobile robots primarily to optimize network weights, tune hyper parameters, and fuse global path planning with local obstacle avoidance [19], [20], [21]. These hybrid approaches address limitations of traditional methods, such as slow convergence, local optima entrapment, and weak adaptability in dynamic environments [22]. For more comprehensive review about researches in mobile robotics, see [23], [24].

In the rest of the paper, the ANNs and GAs are briefly, the proposed main system is explained, the implementation and results are given, and finally discussions and conclusions are made.

2. ANNs and GAs

Neural networks and genetic algorithms are the main subjects that constitute the background that neuro-genetic mobile robot controllers stem from. ANNs are biologically motivated approaches to machine learning, inspired by ideas from neuroscience. An ANN is an information processing system that has certain performance characteristics in common with biological neural networks. ANNs have been developed as generalizations of mathematical models of human cognition or neural biology based on the assumptions that [25]:

1. Information processing occurs at many simple elements called neurons (units, cells, or nodes).
2. Signals are passed between neurons over connection links called synapses.
3. Each connection link has an associated weight, which, in a typical neural net, multiplies the signal transmitted.
4. Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.

ANNs provide a method of representing relationships that is quite different from Turing machines or computers with stored programs. An ANN is characterized by [25]:

1. Its pattern of connections between the neurons (called its architecture which classified to: single-layer NN, multi-layer NN and competitive-layer NN).
2. Its method of determining the weights on the connections (called its training or learning algorithm which contain three types: supervised, unsupervised and reinforcement learning).
3. Its activation function (which have many types like: binary step, binary sigmoid, …, etc).

A Genetic Algorithm (GA) is a population-based stochastic optimization technique inspired by the principles of natural selection and biological evolution. GAs search for high-quality solutions by maintaining a population of candidate solutions and iteratively applying selection, crossover, and mutation. Each candidate solution is represented as a chromosome, while a fitness function measures its quality. The general evolutionary process can be represented as [26], [27]:

$$P^{(t)} \xrightarrow{\text{Selection}} P_s^{(t)} \xrightarrow{\text{Crossover}} P_c^{(t)} \xrightarrow{\text{Mutation}} P_m^{(t)} \rightarrow P^{(t+1)}$$

Where $P^{(t)}$ denotes the population at generation t. The selection operator chooses chromosomes for reproduction according to their fitness. The most widely used selection strategies are roulette wheel selection, and tournament selection. Crossover combines genetic material from two parent chromosomes to produce offspring, and it occurs with probability $p_c \in [0, 1]$. Mutation introduces

random changes into chromosomes and helps maintain genetic diversity, and it occurs with probability p_m [28].

The basic idea behind neuro-genetic controllers for robotics goes as follows: an initial population of different artificial chromosomes, each encoding the control system (and sometimes the morphology) of a robot, are randomly created and put in the environment. Each robot (physical or simulated is then let free to act (move, look around, manipulate) according to genetically specified controller while its performance on various tasks is automatically evaluated. The fittest robots are allowed to reproduce (sexually or asexually) by generating copies of their genotypes with the addition of changes introduced by some genetic operators. This process is repeated for a number of generations until an individual is born which satisfies the performance criterion (fitness function) set by the experimenter [29].

1. The Main System:

The current work combines ANN with GA to construct a neuro-genetic controller for the mobile robot for the task of obstacle avoidance. This combination is chosen for several reasons:

- It does not need the complete specification of the environment, robot, and behavior. Instead, it needs only general indication about the level of performance (fitness) of the current controller.
- The robot is self-adapting and self-organizing from its (proximal) point of view, rather than from the designer's (distal) point of view.
- It has proven experimentally the successfulness of this approach when it is transferred between several platforms and between simulated and real robots. All it needed when transferring is to adapt (by continuing evolution for several additional generations) to the new conditions.
- It can form a basis to study embodied and grounded agents that taking into account the physical aspects of the robot and the environment.
- It is a good tool to study, realize, and test artificial life forms and strategies to both understanding natural life phenomena and discovering new computational solutions to different problems.

In this work genetic algorithm (GA) is used as evolutionary tool to evolve the neuro-controllers and evaluate their performance. Figure (1) clarify the main structure of the modular control system adopted by the current study. The system contains two main modules: the control system module and the robot simulator module. Figure 2 contains the algorithm that used in this work in detail.

The main task that the controllers tackle is obstacle avoidance. This task is necessary to be learned by any autonomous robot since the robot must navigate and explore the world surroundings while

performing its tasks. Also, the system is experimented when modulating two tasks: obstacle avoidance and homing.

An NN is used to implement the control system. NNs are decided to be used since they resistant to noise, which is massively present in the robot/ environment interactions, and they are able to generalize their ability in new situations. Another reason is that NN can easily exploit various forms of learning during its life time, and this learning process may help speed up the evolutionary process.

Figure 3 indicates the architecture of the proposed controller system. It consists of a neural net with eight input neurons and two output neurons and two adjustable weights biased units. Each input unit is fully connected to all output units. The eight sensors of the agent are connected to the input units of the neural net. The output units produce four possible outputs each of which represent an action the agent may take at each situation. The actions are: turn left, turn right, go forward, and go backward. Each of these actions is actually determining the speed of the two wheels of the agent depending on the configuration of the world at the current moment.

Each individual at the population is a controller which having the structure indicated in figure 3. The difference between one individual and other is in the weights that correspond to the connections between neurons. The vectors of weights that define individuals in the population are initialized randomly by assigning to each connection a small random real value. Each chromosome has an 18 gene represent a connection weight value between input neuron and output neuron. The population contains 100 individuals. At the beginning of the life of each one individual, the weights defining such individual is copied into a weight matrix. Each controller is then evaluated by letting it freely to locomote in the environment and to behave according to the situations faced and collect fitness. Adjusting weights may be applied or not depending on a decision made by the experimenter. Adjusting weights means that learning is used in conjunction with evolution; otherwise, the weights are only evolved.

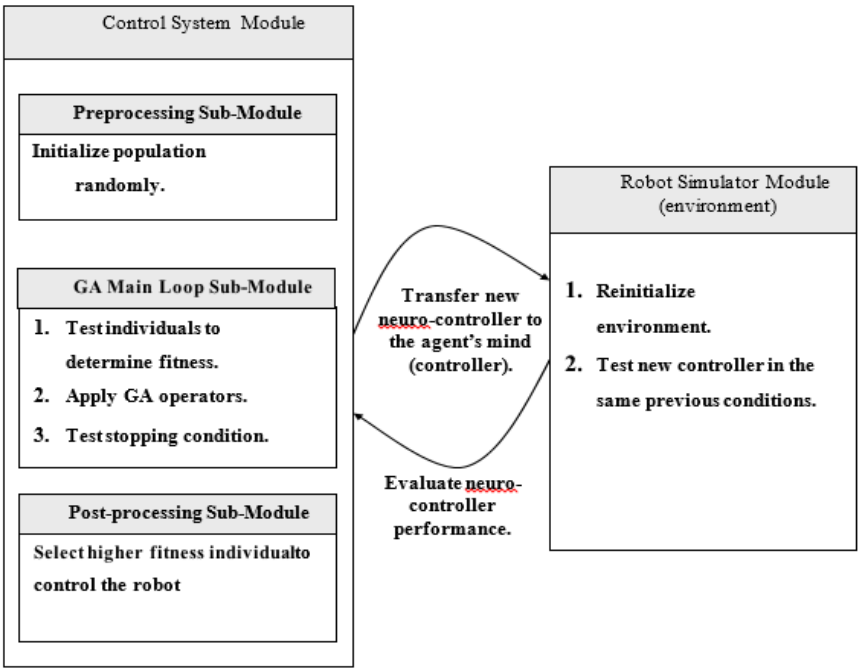

Figure (1): The general organization of the Neuro-genetic controller system

The next stage is evaluation which is very important in the life of each individual since its life or death is decided depending on the level of performance it exhibits during its life duration. This stage actually involves several sub stages: sensing world, taking action, and evaluation. Sensing world sub stage is made as indicated in the algorithm 1 discussed above. The structure of the environment at the current moment (from the agent's point of view) leads to yielding an activation vector that will be the input vector to the neural net and this net in turn decides the action that will be taken at the current moment.

The neuro-controller receives the activation vector from the input neurons and at the each of the output neurons the following computation is occurred [25]:

$$net_j = \sum_{i=1}^{n} (w_{ij} \times x_i) \qquad (1)$$

where w_{ij} is the weight of the synapse connecting ith input neuron and jth output neuron; n is the number of input neurons, and x_i is the ith input value. The net_j is then used to determine the activation of the jth output neuron using the binary step activation function [25]:

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta \end{cases} \qquad (2)$$

The output of the two output neurons constitutes a two valued bina r which represents four possible actions the agent may take at each situation. These actions are:

Algorithm1: Evolutionary Control System.
Input: Robot Simulator, Control System;
Output: Adaptive Behavior;
Begin:

Input mode of operation;
Specify the no. of sensors;
If single behavior mode **then**
While no. of generations not reached
For all individuals in the population **do**

1. Map the individual's genotype into its NN phenotype;
2. Put the NN controller in a random position in the environment;
3. **Do While** not end of individual's life
   a. **If** no. of sensors < no. of input neurons
   Freeze non-used input neurons and their connections;
   b. **For** all sensors **do**
      i. Sense surroundings
      ii. If an obstacle detected
         1) Determine its distance, d, and angle, a;
         2) Specify sensor's activation value from the AMs;
      vi. Else
         sensor's activation value=0;
   c. **End** (for);
   d. **For** each output neuron, j, **do**

$$net_j = b_j + \sum_{i=0}^{n} w_{ij} \times x_i ;$$

   e. If $net_j \geq \theta$ $y_j = 1$ else $y_j = 0$;
   f. Determine the current action depending on vector y values;
   g. Apply current action;
   h. $f = f + v \times (\sqrt{1 - \Delta v}) \times (1 - xj)$;
   i. **If** current individual is osciliated or constant or spinning
      Kill current individual;
4. **End** (of individuals life);
5. Normalize fitness: *f(Ind)=(f × Life)/max life steps;*

End (of all individuals);
Sort population in descending order according to fitness;
Copy the best individual to the next generation population without change;
For each individual in the new population **do**

a. Select two individuals (x and y) from the upper τ individuals in the old population;
b. Recombine x and y with the EIR operator to yield offspring, z;
c. Mutate z with CM operator;
d. Add z to the new population;

End (of all generations);
Save the best individual as the controller to the given task;

Figure (2): The algorithm describing the complete operation of the proposed system

Else If Modular behavior mode, **then**

1. Get the neuro-controller that can avoid obstacles in the environment;
2. Get the experimental environment for the modular behavior;
3. **While** the robot alive **do**
   a. Check the level of energy of the battery;
   b. **If** low level energy
      Switch to modular behavior;
   c. **Else** Switch to obstacle avoidance behavior;
   d. **If** battery filled
      Switch to obstacle avoidance behavior;
   e. **Else**
   f. **While** the battery not filled **do**
      i. Sense surroundings;
      ii. **If** obstacle detected Avoid it;
      iii. Check the ambient light value returned by the sensors;
      iv. Direct the robot movement toward the area having greater light intensity;
      v. Check the value of the underneath sensor;
      vi. **If** changed Fill the battery;
   g. **End** (While);
4. **End** (while alive);

End (begin)

Figure (2): Continued

1. Go Forward: This action is taken when y_1=0 and y_2=0.
2. Go Backward: This action is taken when y_1=1 and y_2=0.
3. Turn Left: This action is taken when y_1=0 and y_2=1.
4. Turn Right: This action is taken when y_1=1 and y_2=1.

At the end of each action, the evaluation sub-stage begins where the fitness of each individual is updated according to the following formula, which is a modified version of Floreano and Mondada equation [6]:

$$fitness = fitness + v \times (\sqrt{1-\Delta v}) \times (1-xj) \qquad (3)$$

Where:

$$\Delta v = \frac{|vLeft + vRight|}{(2 \times vMax)} \qquad (0 \le \Delta v \le 1) \qquad (4)$$

$$v = \frac{|vleft| + |vright|}{(2 \times vMax)} \qquad (0 \le v \le 1) \qquad (5)$$

vLeft is the velocity of left motor, vRight: is the velocity of the right motor, vMax: is the highest velocity that the motors can move according to, and x_j is the highest activation value in the current vector of activation sensors. Usually, the values of the activation sensors are ranged between 0 and 1023, but in our case, they are normalized to be in the [0, 1] interval. v is a measure of the average rotation speed of the two wheels, Δv is the algebraic difference between the signed speed values of the two wheels (positive one direction and negative the other) transformed into positive values.

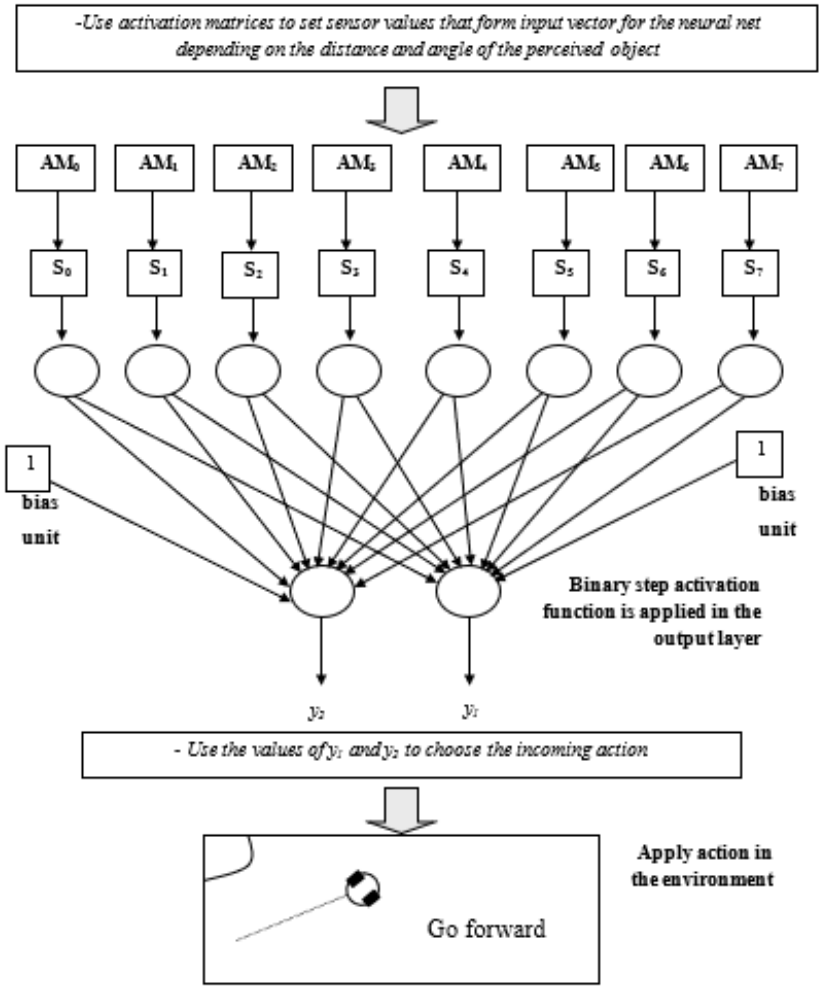

Figure 3: The structure of the controller system. Each of the sensors is connected to one of the input neurons and each output neuron is defining one possible action. AM_i is the ith activation matrix and S_i is the ith sensor activation.

This fitness function is chosen because it encourages motion, straight direction and low sensor activation. Without the first component, a robot standing far from a wall would achieve maximum fitness. Without the second component in the fitness function, the maximum fitness could be easily achieved by fast spinning in the same place (wheels turning at maximum speed in opposite directions) far from walls. Without the third component, evolution would develop robots moving straight (frontally or backward) at maximum speed until they crashed against a wall.

At the end of individual's life, the fitness is averaged over all life steps to get the final evaluation of the individual such that $0 \leq \text{fitness} \leq 1$, as indicated in the algorithm.

The fitness function is considered to be an automatic way of evaluating individuals. This implies that the fitness function should compute only information that is available to the robot through its internal or external sensors. The fitness function indicated in equation 3 is an example of that, where it uses information that is available to the robot through its infra-red sensors and through its internal sensors of the state of the motors.

The Modular Mode:

To test the performance of the system when an additional task is performed in addition to obstacle avoidance, the task of homing is considered. To simulate this task the robot is equipped with a simulated battery that lasted after a pre-specified action steps (150 step). A simulated battery charger, having a different color, is put in the left upper corner under a light source. To charge the battery, the robot must pass through the recharging area. Additional sensor is put underneath the robot to measure the floor brightness. When the robot is happened to pass through the recharging area the battery is immediately filled and the age of the robot increased 150 actions.

A module is added whose task is to lead the robot to the recharging area when its energy minimized. A simple switching procedure is also added to switch between the obstacle avoidance mode to the modular mode. In the obstacle avoidance mode, the robot is navigating and avoid obstacles, while in the modular mode, which is selected when the energy in the simulated battery reached low levels, the robot directs its movement to the recharging area and at each step it decides to select either obstacle avoidance behavior or goal directed movement, depending on the sensation of the robot at that moment. The light source helps the robot to know the location of the recharging area, where a light following procedure is implemented to reach recharging area that sensed by the underneath sensor. When the underneath sensor receives a different value, the battery is recharged. The algorithm listed in figure 4 gives the outline of the modular mode behaviour.

System Implementation and Results

Table 1 indicates the parameters and settings that used in the experiments at the implementation phase.

When the system is traced during implementation, several types of individuals are noticed that exhibit the following undesired behaviors:

1. Spinning at the same location.
2. Oscillation between two adjacent cells.
3. Individuals that still constant in the same cell without changing direction or position.

These types of individuals are "killed" after a pre-specified number of life-steps (10 life-steps are specified) if they not improve their behavior. According to the fitness function used, individuals of type 1 and two are tend collect a good credit if they take the chance to "live" to the maximum number of life-steps. The "age" of individual is taken into account when evaluating its performance by multiplying its fitness by the following term:

$$\text{Fitness} = \text{fitness} \times (\text{age} / \text{maximum life-steps}) \qquad (6).$$

This decreases the fitness of short age individuals and increases the fitness of long age individuals

```
Algorithm 2: Modular mode;
Input: level of battery energy;
Output: switching behavior;
Begin:
        While life do
                Check battery level;
                If energy minimized
                 1. Switch to the modular mode;
                 2. While the battery not filled do
                        a. Sense surroundings;
                        b. If obstacle faced, avoid it;
                        c. Else direct the robot movement to the area having greater light intensity
                           measure;
                        d. Check the value of the underneath sensor;
                        e. If changed then fill the battery immediately;
                 3. End (do)
                Else
                 Switch to the obstacle avoidance mode;
        End(do)
End.
```

Figure (4): Modular mode algorithm

Table (1): Parameters and settings used in the experiments

GA Parameters	
size of population	100
length of chromosome	18
number of generations	100
maximum age of each individual	500
initial population gene initialization formula	(random (100)/10000)
number of elitist individuals (q)	1
percentage of the population best individuals that would contribute in the next generation (τ)	25%
selection type	truncation selection
recombination type	One-point crossover
mutation type	Random gene resetting
NN Parameters	
network type	single - layer feed- forward network
activation function type	binary step function
threshold (θ)	0

Three environments were used to test the system performance:

1. env1: A simple environment with only borders as obstacles and a lamp on the upper left corner (see figure 5).
2. env2: An environment having seven circular obstacles that made a line in the middle of the environment (see figure 6).
3. env3: An environment having six circular obstacles that made two lines in the environment (see figure 7).

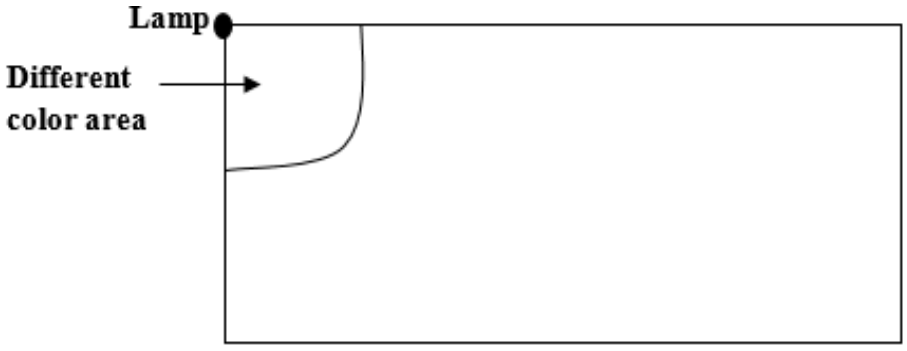

Figure (5): env1, a simple environment with borders only obstacles.

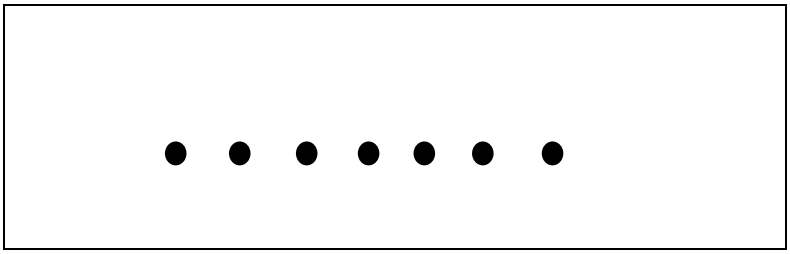

Figure (6): env2, an environment with 7 circular obstacles in addition to borders.

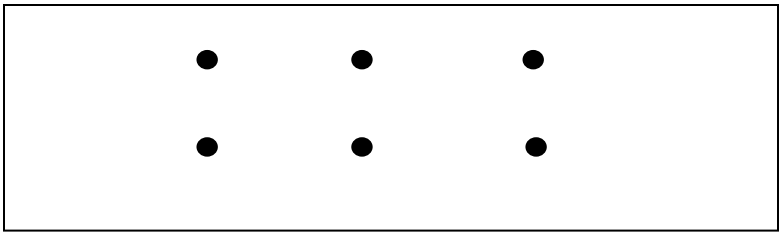

Figure (7): env3, an environment with 6 circular obstacles in addition to borders.

Table (2): Experimental layout

Obstacle Avoidance Mode (one task)					
Set1 experiments					
Experiment no.	Environment	Run replications	Life steps	No. of sensors	Corresponding figures
1	env1	4	500	8	8, a, b, c
2	env2	4	500	8	9, a, b, c
3	env3	4	500	8	10, a, b, c
Modular Mode (two tasks)					
Set2 experiments					
Experiment no.	Environment	Run replications	Life steps	No. of sensors	Corresponding figures
1	env1	4	500	8	11, a, b, c

Table 2 gives the layout of all experiments and their parameters and the corresponding figures and graphs. Several sets of experiments were performed, these are:

- **Set 1**: Investigating the feasibility and the degree of successfulness of the system in several different environments.
- **Set 2**: Investigating the effect of combining the obstacle avoidance task with another task (homing) on the behavior of the evolved individuals.

Set 1 Experiments (testing the performance of the system- 8 sensor robot

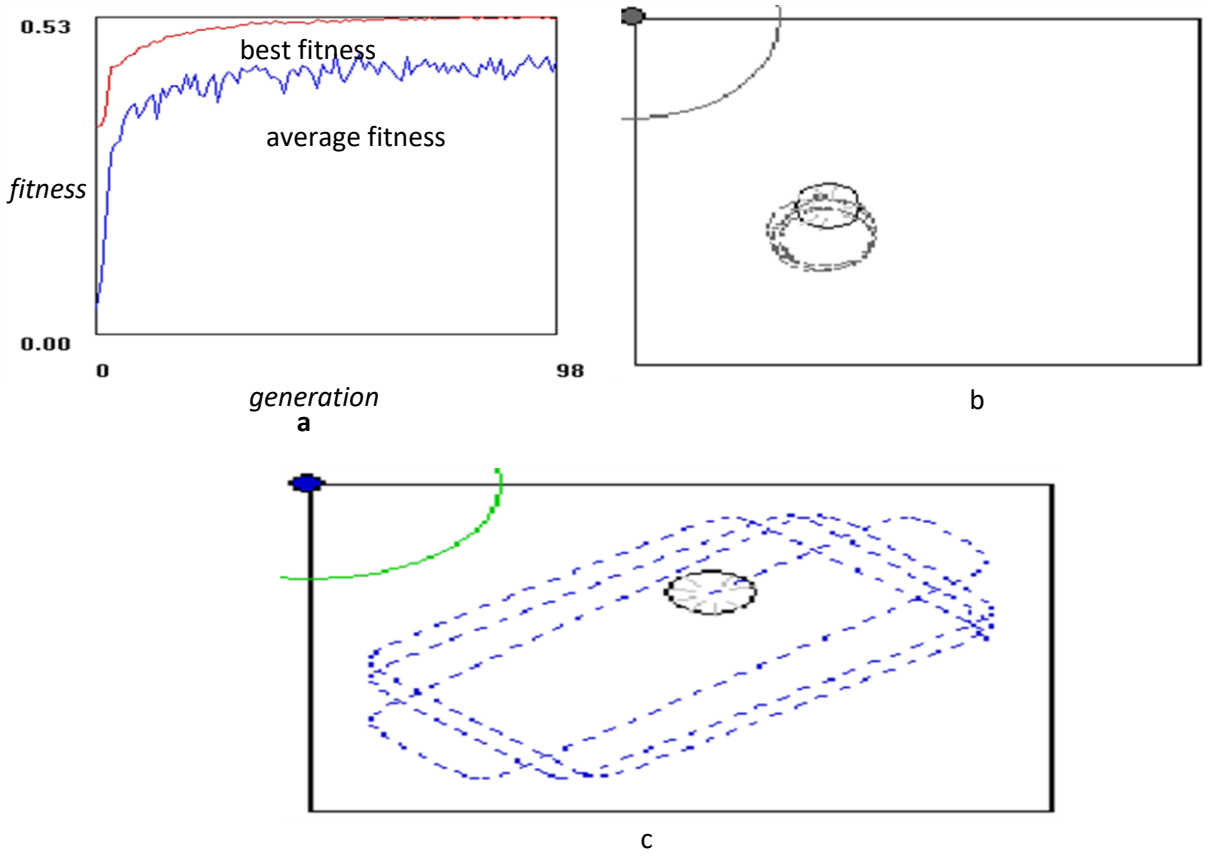

Figure (8): 8 sensor robot is trained to explore env1. a. the fitness of the best individual and the average population fitness across generations, b. the behavior of the best individual before evolution, and c. the behavior of the best individual after evolution.

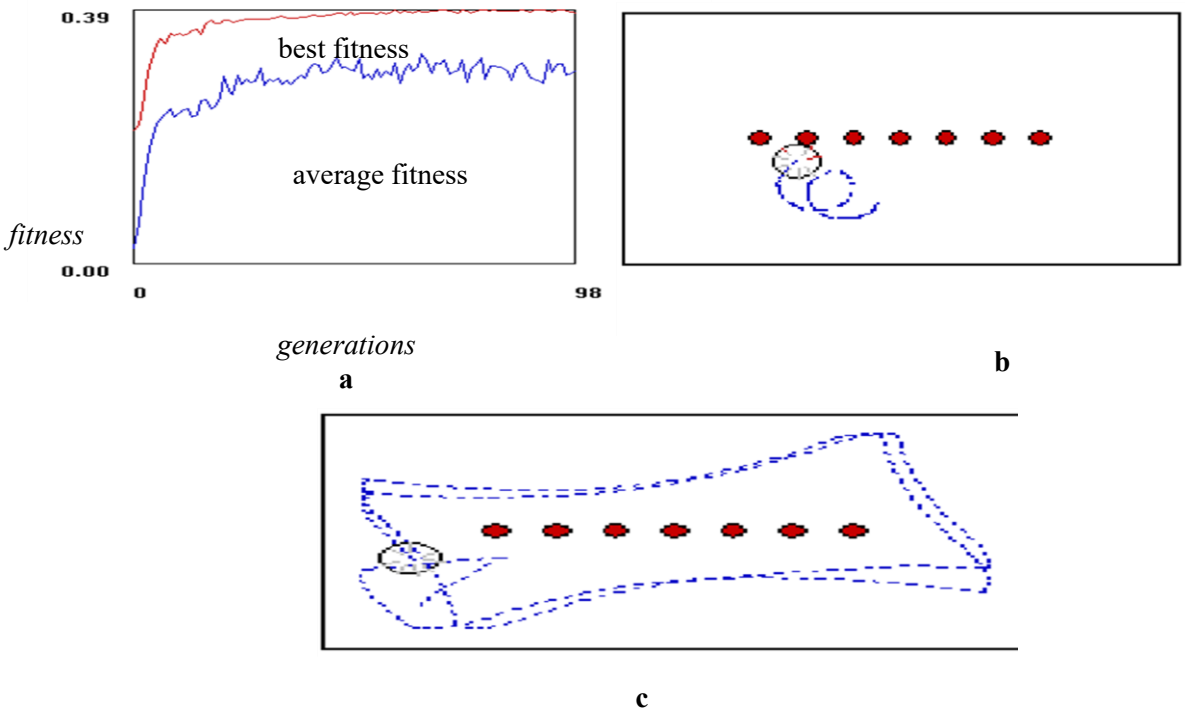

Figure (9): 8 sensor robot is trained to explore *env2*. a. the fitness of the best individual and the average population fitness across generations, b. the behavior of the best individual before evolution, and c. the behavior of the best individual after evolution.

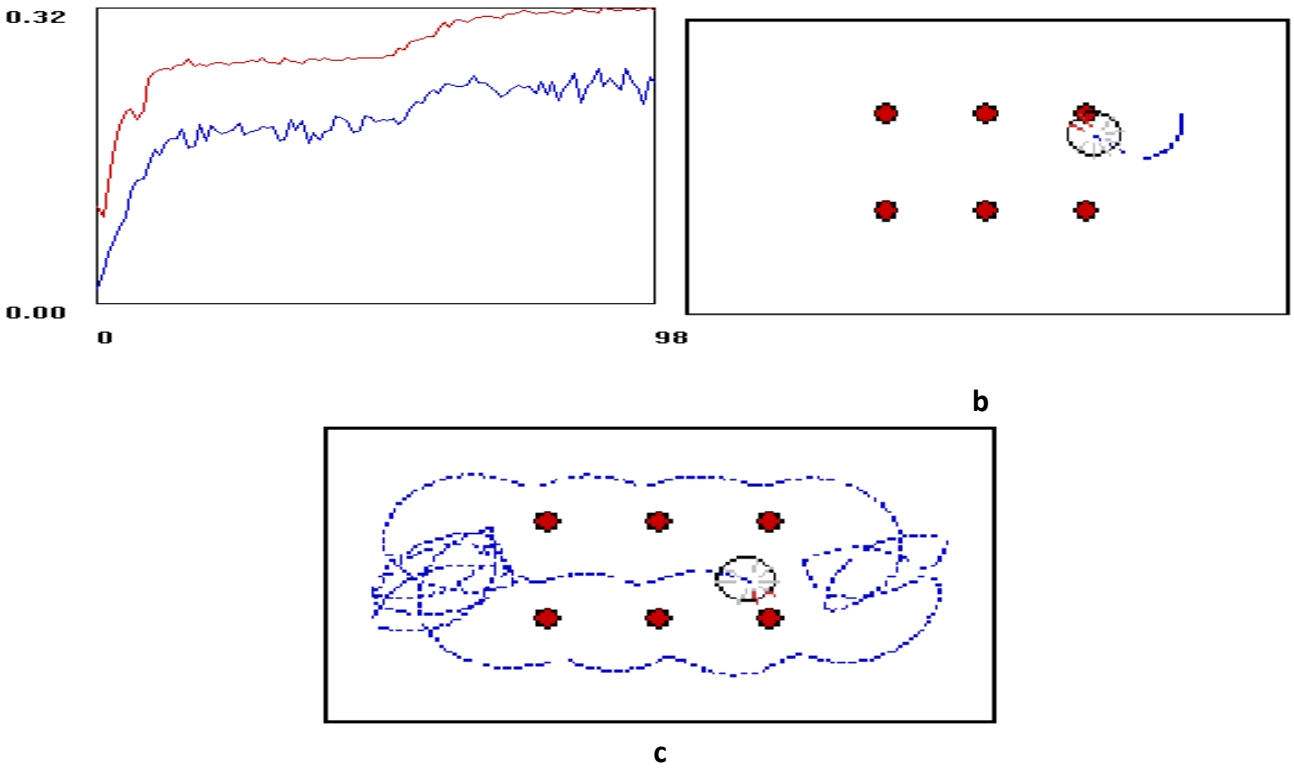

Figure (10): 8 sensor robot is trained to explore *env3*. a. the fitness of the best individual and the average population fitness across generations, b. the behavior of the best individual before evolution, and c. the behavior of the best individual after evolution.

Set 2 Experiments (testing the performance of the system -two tasks modulated):

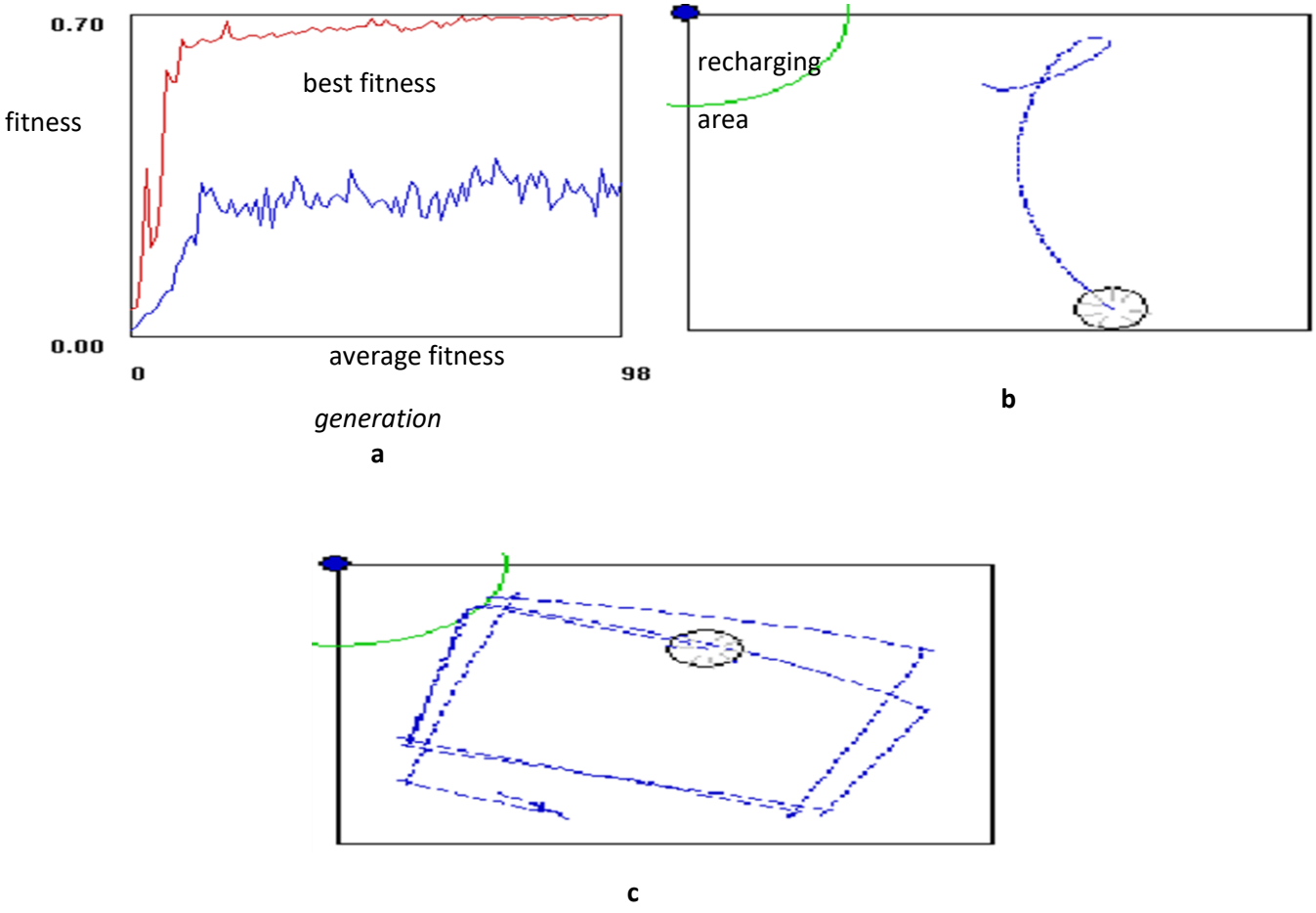

Figure (11): 8 sensor robot is trained to explore *env1* and to recharge the simulated battery. a. the fitness of the best individual and the average population fitness across generations, b. the behavior of the best individual before evolution. c. the behavior of the best individual after evolution.

Four run replications were used for each experiment and the best individual from the four runs is tested in the environment. 500 life steps for each individual are performed during evaluation stage.

Discussions and Conclusions

1. **Variability of Evolutionary Runs:**

   Because evolutionary algorithms are stochastic, independent runs on the same problem may produce different solutions and performance levels. Therefore, conducting the evolutionary process in simulation is advantageous, as it allows multiple runs of the same experiment to be performed and facilitates a more reliable assessment of the system's performance.

2. **Importance of Simulation:**

   Simulation provides an effective environment for determining and refining the main parameters of neuro-genetic controller systems before implementation. These parameters include the neural network architecture, initial weight matrix, environmental configuration, and the type and number of sensors. This makes it possible to investigate different configurations efficiently and identify suitable settings for the evolutionary process.

3. **Fitness Function Design:**

   The design of the fitness function is a critical factor in obtaining the desired robot behavior. The fitness function defined in Equation (3) incorporates three complementary components that promote forward motion, straight-line movement, and obstacle avoidance. The first component, (V), rewards higher wheel speeds; the second component, (\cdots), promotes straight movement; and the third component, ((1-x_i)), encourages the robot to maintain a safe distance from obstacles. The combination of these components is essential for producing effective autonomous navigation behavior.

4. **Evaluation of Evolutionary Performance:**

   The fitness function provides an objective measure for evaluating the performance of the evolutionary system. Across the experiments, both the average population fitness and the fitness of the best individual generally improved over successive generations, indicating progressive adaptation of the population. In addition to this quantitative assessment, the evolved controllers can be evaluated qualitatively by examining the trajectories and navigation behaviors generated by the best individuals.

Set 1 Experiments:

5. The results of Set 1 demonstrate that the proposed neuro-genetic controller can successfully evolve individuals capable of exhibiting the desired navigation behavior, particularly obstacle avoidance.
6. The use of eight infrared sensors provides sufficient information about the robot's surrounding environment, enabling the evolved controller to make appropriate navigation decisions.
7. The comparison between the robot's behavior before and after evolution demonstrates that the fitness function defined in Equation (3) is effective for automatically evaluating and guiding the evolution of candidate controllers toward the desired behavior.
8. The evolutionary process successfully generated effective individuals across the different environments. In particular, Environment 3 (Env3) demonstrates the capability of the evolved controller to identify a feasible trajectory through the passage between two groups of obstacles, highlighting the adaptability of the evolutionary approach.
9. The results also show a progressive improvement in population fitness throughout the evolutionary process. As the number of generations increases, the algorithm is able to discover individuals with increasingly better performance, demonstrating the effectiveness of evolutionary optimization in developing autonomous navigation controllers.

Set 2 Experiments:

10. Set 2 demonstrates the ability of the evolutionary system to integrate and modulate multiple behaviors within a single controller.
11. The results show that the evolutionary process can discover individuals capable of performing autonomous navigation while periodically directing the robot toward the designated recharging area. This demonstrates that the evolved controller can coordinate navigation and battery-recharging behavior within the same control system.

Future research could focus on developing multi-objective fitness functions that simultaneously consider navigation efficiency, obstacle avoidance, energy consumption, and path length. Another aspect can be studied is to examine different network structures and learning mechanisms to improve adaptability and generalization of evolved controllers. Also, the experiments can be extended to dynamic and more complex environments containing moving obstacles and changing environmental conditions.

References

1. M. A. Al-Bayati, Genetic Algorithm Based Path Planner, Ph.D. dissertation, Dept. Comput. Sci. Inf. Syst., Univ. Technology, Baghdad, Iraq, 1998.
2. J. R. Koza, Genetic Programming: on the Programming of Computers by Means of Natural Selection. Cambridge, MA, USA: MIT Press, 1992.
3. S. Nolfi, D. Floreano, O. Miglino, and F. Mondada, "How to evolve autonomous robots: Different approaches in evolutionary robotics," in Proc. 4th Int. Conf. Artif. Life, Cambridge, MA, USA: MIT Press, 1994.
4. O. Miglino, H. H. Lund, and S. Nolfi, "Evolving mobile robots in simulated and real environments," Artificial Life, vol. 2, no. 4, pp. 417–434, 1995.
5. J. Urzelai and D. Floreano, "Evolutionary robotics: Coping with environmental change," in Proc. Genetic and Evolutionary Computation Conf. (GECCO), 2000.
6. D. Floreano and F. Mondada, "Evolution of plastic neurocontrollers for situated agents," in From Animals to Animats 4: Proc. Int. Conf. Simulation of Adaptive Behavior, P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. Wilson, Eds. Cambridge, MA, USA: MIT Press, 1996, pp. 402–410.
7. D. Floreano and J. Urzelai, "Evolutionary robots with on-line self-organization and behavioral fitness," Neural Networks, vol. 13, nos. 4–5, pp. 431–443, 2000.
8. D. Floreano, "Evolutionary re-adaptation of neurocontrollers in changing environments," in Proc. Conf. Intell. Technol., vol. II, P. Sncak, Ed. Kosice, Slovakia: Efa Press, 1996, pp. 9–20.
9. S. Nolfi, "Evolving non-trivial behaviors on real robots: A garbage collecting robot," Robotics and Autonomous Systems, vol. 22, pp. 213–234, 1997.
10. S. Nolfi, "Using emergent modularity to develop control systems for mobile robots," Adaptive Behavior, vol. 5, pp. 343–363, 1997.
11. R. Calabretta, S. Nolfi, D. Parisi, and G. P. Wagner, "Duplication of modules facilitates the evolution of functional specialization," Artificial Life, vol. 6, no. 1, pp. 69–84, 2000.
12. D. Floreano and F. Mondada, "Active perception, navigation, homing, and grasping: An autonomous perspective," in Proc. From Perception to Action Conf., pp. 122–133, Sep. 1994.
13. D. Floreano and S. Nolfi, "God save the red queen! Competition in coevolutionary robotics," in Proc. 2nd Int. Conf. Genetic Programming, J. Koza, K. Deb, M. Dorigo, D. Fogel, M. Garzon, H. Iba, and R. L. Riolo, Eds. San Mateo, CA, USA: Morgan Kaufmann, 1997.
14. S. Nolfi and D. Marocco, "Evolving robots able to integrate sensory-motor information over time," Theory in Biosciences, vol. 120, pp. 287–310, 2001.

15. D. Marocco, A. Cangelosi, and S. Nolfi, "The role of social and cognitive factors in the emergence of communication: Experiments in evolutionary robotics," Philosophical Transactions of the Royal Society B: Biological Sciences, vol. 361, no. 1811, pp. 2397–2421, 2003.
16. S. Nolfi, J. Bongard, P. Husbands, and D. Floreano, "Evolutionary robotics," in Springer Handbook of Robotics, 2nd ed., B. Siciliano and O. Khatib, Eds. Cham, Switzerland: Springer, 2016, pp. 2035–2067.
17. D. R. Bruno, N. Marranghello, F. S. Osorio, and A. S. Pereira, "Neurogenetic algorithm applied to route planning for autonomous mobile robots," in Proc. 2018 Int. Joint Conf. Neural Networks (IJCNN), 2018.
18. A. K. Rath, D. R. Parhi, H. C. Das, P. B. Kumar, and M. K. Mahto, "Design of a hybrid controller using genetic algorithm and neural network for path planning of a humanoid robot," International Journal of Intelligent Unmanned Systems, vol. 9, no. 3, pp. 169–177, 2020.
19. R. Younis, M. T. Ejaz, and S. Davarzani, "Optimizing mobile robot navigation: A genetic algorithm and convolutional neural network approach," in Proc. 2024 Int. Conf. Electr., Commun. Comput. Eng. (ICECCE), pp. 1–6, Oct. 2024.
20. Y. Shi, G. Zhang, and M. Kong, "Path planning of welding robot based on deep learning," Paladyn, vol. 15, no. 1, Art. No. 20240002, 2024.
21. Z. Zhu, S. Wang, and T. Wang, "Optimizing robotic mobile fulfillment systems for order picking based on deep reinforcement learning," Sensors, vol. 24, no. 14, Art. no. 4713, 2024.
22. Z. Zhang, H. Yang, X. Bai, S. Zhang, and C. Xu, "The path planning of mobile robots based on an improved genetic algorithm," Applied Sciences, vol. 15, no. 7, Art. no. 3700, 2025.
23. A. Waga, S. Benhlima, A. Bekri, J. Abdouni, and F. Z. Saber, "A survey on autonomous navigation for mobile robots: From traditional techniques to deep learning and large language models," Journal of King Saud University Computer and Information Sciences, vol. 37, no. 7, Art. No. 198, 2025.
24. K. Katona, H. A. Neamah, and P. Korondi, "Obstacle avoidance and path planning methods for autonomous navigation of mobile robot," Sensors, vol. 24, no. 11, Art. no. 3573, 2024.
25. L. Fausett, Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Upper Saddle River, NJ, USA: Prentice Hall, 1994.
26. J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, 2nd ed. Cambridge, MA, USA: MIT Press, 1992.

27. I. De Falco, "Nonlinear system identification by means of evolutionary optimized neural networks," in Genetic Algorithms and Evolution Strategy in Engineering and Computer Science, D. Quagliarella, J. Periaux, C. Poloni, and G. Winter, Eds. Chichester, U.K.: John Wiley & Sons, 1998.
28. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA, USA: Addison-Wesley, 1989.
29. S. Nolfi and D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Cambridge, MA, USA: MIT Press, 2000.