Optimized path planning of an unmanned vehicle in an unknown environment using the PSO algorithm

V Tavoosi\textsuperscript{1,2}, J Marzbanrad\textsuperscript{1*}, M Golnavaz\textsuperscript{3}

1- School of Automotive Engineering, Iran University of Science and Technology, Tehran, Iran
2- Young Researchers and elite Club, Qazvin Branch, Islamic Azad University, Qazvin, Iran
3- Iran National Standardization Organization, Tehran, Iran

Abstract. Today, the use of drones has expanded, particularly in high-risk and/or inaccessible environments, or situations where the cost of human resource is high. One of the most important uses of these unmanned vehicles is in rescue fields where they carry instruments and resources, or transfer wounded people. One of the most important discussions in this regard is the issue of routing these cars in an unknown environment. The first step for the vehicle to start its mission is to drive around environmental barriers. Environmental barriers can be divided into two categories. The first category comprises barriers that can be located on a map using satellite imagery and known maps. The second category contains obstacles that the car may encounter while navigating the path and but which have not been anticipated. To solve this problem, this research uses the PSO method to optimize offline routing in an environment with a specified map. The vehicle then may encounter new obstacles when moving on the planned path and identify those obstacles using sensors. In this case, using the neural network algorithm, it can obtain an optimal pseudo-path to circumvent the obstacle. In fact, the issue is divided into two sections. The first issue is the optimal routing with the PSO method, and the second section tackles the problem of dealing with unplanned obstacles using the neural network algorithm.

1. Introduction

1.1. The issues and the necessity of the research

Mobile robots are widely used in various environments. Applications are diverse and include servicing robots for elderly people, automated guided vehicles for transporting equipment in different environments, as well as robots for the exploration of planets. One of their most important advantages is that moving robots are usually used in places where there is no easy access for humans, or where access endangers human health.

Therefore, the necessity of automatic navigation and transportation is keenly felt. In recent years, many efforts have been made to improve the automatic path planning of vehicles and robots. In this research, the goal is to estimate the path of an unmanned vehicle in an unknown environment. Using the proposed system, a robotic vehicle can plan the journey from an origin to a target point and drive the route in the shortest possible time, avoiding obstacles. Ideally it also can quickly respond to unexpected obstacles that are present in the environment.
1.2. Problem-solving methodology

Robot routing methods are sorted into two main categories, those being classical methods and heuristic methods. Classical methods include cell decomposition, potential field, sub-goal network and road map. Although these methods are simple, they require a lot of computer calculation, and may also generate errors in the presence of uncertainties. In the literature review section, the focus is primarily on heuristic methods including neural networks and PSO.

In this research, a random environment is designed and a map of the environment is available to the robot. The features of this environment are various features of a natural environment. For example, any kind of obstacle, sloping surfaces or ruggedness of the environment is indicated by use of a specific shape and colour. Then, a nature-inspired optimization method named the PSO algorithm is used to find the optimal path in this environment. Of course, in this research, it has been important also to consider possible obstacles that are not seen in the satellite map or that may appear suddenly in front of the robot. As the robot detects these obstacles by its sensors at a certain distance, and in the fastest time, it chooses an alternative route that does not encounter obstacles, in order to prevent a collision with the obstacle in its path.

In the second section, a review of earlier research is conducted with a focus on optimal path planning for drone robots. In the third section, the PSO algorithm chosen for optimization is described in detail. In section four, the proposed method has been simulated and evaluated in different conditions and environments, and the numerical results are discussed. In the clause five, a summary of the research is provided and suggestions for the continuation of the work are presented.

2. Review

In the following sub-sections the neural network and PSO methods and relevant recent works are reviewed.

2.1. Neural networks

In recent years, the neural network method has been widely used in solving robotic path planning problems. The application of this method is more evident when the relation between the input and the output is complex. For example:

1. Interpreting sensor data
2. Avoid obstacles
3. Route design.

In 2000, Yang and Meng [2] designed a route for a robot in a non-static environment, which, in addition to static obstacles and dynamic barriers, also considered a moving target point. The results of the study showed that the robot was moving away from obstacles toward the moving target. The dimensions of the robot, the obstacles and the goal were not considered in this simulation, and the speed of the barriers and robots were considered either. Therefore, there may be problems when this system is tried in the real world.

In 2010, Antonelo [3] solved the robot routing problem using a recursive neural network (RNN) method. In this method, two RNNs were used, the first to know the environment and the second for finding a path to the target point. Using this method in real-world environments has proven its success in facing an unknown environment.

In 2004, Janglova [4] used two neural networks to solve routing problems without dealing with obstacles. One of these networks was the principal component analysis (PCA) method and the multilayer perceptron (MLP). PCA is a linear learning method without a trainer, used to analyse complex inputs. Using the information the robot receives from ultrasound sensors, it can determine its distance from all environmental barriers and generate output $V_i$ as the free spaces that the robot can
use. The output of the first network, along with which indicates the position of the target, is used as inputs of the MLP neural network, and the output of the MLP is the path that the robot sends to the controller. This network has 18 input neurons, 20 neurons, and nine neurons. The advantage of this method is the convenience of applying the program and obtaining the input / output relationship, and its failure is also low speed and the number of high samples needed to train the network.

In 2013 Dezfolian[5], inspired by the method used in article [3], deployed this method to three robots that used different sensors and had different target points. All three robots succeeded in reaching the target. The advantage of using this method was the use of inexpensive sensors, and the biggest disadvantage the high volume of data needed to train the network (3000 patterns) for different scenarios.

2.2. Nature-inspired algorithms
In recent years, nature-inspired methods for robot routing have been widely considered. In the following sections, we will briefly summarize some of these methods, such as GA, PSO, and ASO. The reason for selecting these three methods is to understand the proper functioning of these methods according to various reports in the articles.

2.2.1. Genetic Algorithms (GA)
A genetic algorithm is a natural-gene-inspired optimization method that uses some of the evolutionary processes of genes, including natural selection, composition, and mutation. Genetic algorithm is a very suitable method for solving hybrid optimization problems.

Many studies have been carried out using the genetic algorithm to solve robot routing problems. In [6], a genetic algorithm based on information is used instead of the standard genetic algorithm to solve the problem. The proposed method is suitable for both static and dynamic environments. This algorithm was expanded in [7] for several mobile robots in a dynamic environment. When the robot’s work environment is simple or slow, the algorithm suggests a near-optimal solution. But in complex environments with fast dynamics, a path is always possible. A robust algorithm based on genetic algorithm has been used to prevent the robot being caught in U- and V-shaped obstacles. This algorithm also has a goal-oriented approach and thus avoids unnecessary searches. The function of this algorithm, both in the simulation environment and in the real environment, with two factors is, in the first place, the feasibility of the route, i.e., the failure to deal with obstacles, as well as finding the shortest path.

In [8] Oleiwi et al used a hybrid method based on genetic algorithm, A* algorithm and fuzzy logic. First, using the genetic algorithm and A*, an optimal path is obtained, and this path is given as input to the fuzzy logic controller to design a robot path. When new obstacles appear in the path, the fuzzy controller reduces the speed of the robot and helps the robot to prevent collisions. The function of this method was very suitable in an environment with dynamic barriers.

In the paper [9], a fuzzy logic-based approach and a genetic algorithm were proposed to follow the moving goals. In article [10], GA and PSO combined in optimal routing for automatic robots.

In [11], a novel neuro genetic approach was proposed to robotize a robot in an unknown environment. In this way, the genetic algorithm performs optimal routing in an unknown dynamic environment. The cost function consisted of three variables: avoidance of obstacles, route shortening, and routeability. In order to avoid falling into local livestock and preventing the convergence of parameters, Karami et al. [12] designed an adaptive selectivity algorithm that operated with a feeder that came from the environment. In the first stage, the primary population was randomized, then the path designed for each generation was optimized using the genetic algorithm. As a result, robot routing has significantly improved.
In short, the greatest problem of designing a path using the genetic algorithm is its inapplicability in dynamic environments. Genetic algorithms may converge before the population reaches puberty. To solve this problem, the combination of the genetic algorithm with other algorithms such as fuzzy, PSO, or etc., is used.

2.2.2. PSO

Similar to the genetic algorithm, the PSO algorithm also acts to produce a random population and then evaluates the cost of a population using a function. Of course, some of the features, such as the combination and mutation that existed in the genetic algorithm, disappear here. Particles update their speed with the help of personal experience and social experience. The PSO can also consider real numbers as particle, which is an advantage over GA, which should eventually make a conversion from binary codes to real numbers.

The PSO is inspired by the massive movement of birds and fish. This algorithm is a population-based probabilistic optimization technique. The PSO starts with a randomized approach and optimizes it continuously with optimization techniques. Ultimately, the optimal population is obtained by changing the composition of the population in the desired workspace.

Masehian and Sedighizadeh (2010) [13] solved the routing problem using the PSO algorithm that updates their position by using Equations 1 and 2 to reach the target position at each stage. Particle movement is affected by the cost function that evaluates each response. Figure 1 shows the left side of the updated particle \( j \) in step \( i \).

\[
\text{prtpos}_{j}^{i+1} = \text{prtpos}_{j}^{i} - 1 + \text{prtvel}_{j}^{i+1}
\]  

\[
\text{gbest}^{i} \quad \text{prtpos}_{j}^{i} \quad \text{pbest}_{j}^{i}
\]  

**Figure 1.** The status update of a particle in PSO. [8]
\[ p_{r t v e l}^i_j = \chi \left[ w. \ p_{r t v e l}^{i-1}_j + c_1 r_1 (p_{b e s t}^{j-1}_i - p_{r t p o s}^{j-1}_i) + c_2 r_2 (g_{b e s t}^{j-1}_i - p_{r t p o s}^{j-1}_i) \right] \] (2)

Where:
- \( p_{r t p o s}^{j} \) is the position of the particle \( j \) in the \( i+1 \) stage.
- \( p_{r t v e l}^{j} \) is the velocity of the particle \( j \) in the \( i+1 \) stage.
- \( p_{b e s t}^{j-1}_i \) is the best position of the particle \( j \) at stage \( i \).
- \( g_{b e s t}^{j-1}_i \) is the best position to stage \( i \).
- \( c_1 \) and \( c_2 \) are custom factors and \( r_1 \) and \( r_2 \) are random numbers between 0 and 1.
- \( \chi = 2 \sqrt{2 - \varphi - \sqrt{\varphi^2 - 4 \varphi}} \), \( \varphi = \varphi_1 + \varphi_2 > 4 \).

3. Formulation and problem-solving method

3.1. Introduction
In the previous section, explanations were given of the studies conducted in the field of optimal path planning methods for robots. In this section, the methods used to achieve the purpose of this study are fully described.

3.2. Offline path planning using the PSO optimization method
The PSO optimization method was introduced in 1995 by James Kennedy and Russell Eberhart [14]. They initially intended to use a combination of social models and existing social relationships to create a kind of computational intelligence that does not require special individual abilities. Their work led to the creation of a robust algorithm for optimization, called particle swarm optimization algorithm, or PSO. This method has been adapted to the collective function of animal groups, such as birds and fish.

In the PSO algorithm, there are a number of organisms, we call them particles and they are spun in the search space. Each particle calculates the value of the objective function in the position of the space in which it is located. Then, using the combination of its current location data and the best place previously provided, as well as the information of one or more particles of the best particles in the aggregate, it selects a direction for movement. After a collective move, one step of the algorithm ends. These steps are repeated several times until the desired answer is obtained.

In particle optimization algorithm, each particle has five properties. First of all, every particle in space has a position, and since it has a position, one can also calculate a measure corresponding to this position. For example, in our optimization problem, we have a cost function corresponding to each position, which we must minimize or maximize. So, for each particle next to its corresponding position, we also compute a cost function. In addition to the position, each particle has one direction of motion, which, because it is a vector number, is called the vector of the particle. Each particle also has the best position or the best memory experienced in the past, and consequently it also has the corresponding cost associated with that position.

The five properties of each particle in space are as follows:
- \( x^i \): position of the particle
- \( f(x^i) \): The target function corresponding to the position of the particle
- \( v^i \): the velocity of the particle
- \( x^{i,b e s t} \): The best experienced situation
- \( f(x^{i,b e s t}) \): The objective function of the best experienced situation

Assume that one of the particles is in the blue position in Figure 2, and the path of the particle motion is represented by the vector, as shown below. The dotted point is the best memory of the blue particle,
and the red dot is the best experience of the whole set. Now, using the equation, the particle can be moved to a new location so that the new position has the best cost function. For this purpose, the equation orders the particle to move in the same direction (black vector), part orders the particle to move in the direction of the best previous experience (grayscale vector line), and the part maps the particle in order to move towards the best of the whole experiment (red line vector). The final output is the sum of the current position of the particle with the particle velocity vector, which results in the new position of the particle and the vector of the new particle velocity, which is indicated in the figure with the green sign and the yellow vector. The Equations 3 and 4 describe the particle behaviour. This equations are rewritten form of Equations 1 and 2.

\[
\begin{align*}
    v_i[t + 1] &= w v_i[t] \\
    &+ c_1 r_1 (x_i^{best}[t] - x_i[t]) \\
    &+ c_2 r_2 (x_{gbest}[t] - x_i[t]) \\
    \end{align*}
\]

(3)

\[
    x_i[t + 1] = x_i[t] + v_i[t + 1]
\]

(4)

\[2\]

In Equation 3 the coefficients are very important and effective. The \(w\) is the coefficient of inertia, because this part of the equation actually indicates the tendency of the particle to maintain the direction of the previous move. This number should be less than one and is often selected between 0.4 and 0.9. The less inertia, the faster the convergence algorithm will be. \(r_1\) and \(r_2\) are random numbers with uniform distribution. \(c_1\) and \(c_2\) are also learning coefficients, the first is the personal learning coefficient and the second is the collective learning factor. Usually learning coefficients between 0 and 2 are chosen. Of course, these values can always change, and on some issues these changes can help to solve the problem faster and better.

Two very important concepts in the PSO algorithm are the concept of search and the concept of exploitation, which means the ability to generate new responses. The search algorithm searches for spaces that had not previously been searched for, and finds more optimal answers. In the exploitation capability, the algorithm searches for the neighboring point of interest and optimizes the previous answers. Increasing the \(w\)-factor is in favour of searching and reducing this coefficient is in the interests of exploitation. On the other hand, if the coefficients \(c_1\) and \(c_2\) are very large, the algorithm will go for search, and if they are too small, it will go for exploitation. Generally speaking, algorithms should be written in such a way that the search share is initially high and then the share of exploitation is gradually increased.

After the particle has arrived at the new position, the cost function must be recalculated and compared to the best of its own experience. If the particle has a better performance than its previous record, this new experience is replaced by previous experience, and in that case, the best overall record of the total must also be compared so that if it also performs better, the best record of the total will also be updated.
The steps to implement a PSO algorithm can be written as follows:
1. Create a primary population and assess it.
2. Determine the best personal memories and best collective memories.
3. Update speed and position and evaluate new responses.
4. If the conditions are not met, stop and return to step 2.
5. If the conditions are met, stop, the end of the algorithm.

Stop conditions can have different shapes, depending on the need for the problem. For example, if our optimal cost is a fixed number, we restrict the condition to this number, and when the cost function reaches this number, the problem is stopped. Or, a certain number of iteration or a specific execution time can be considered as a condition of stop. In the same way, the non-reducing rate in the cost function can be a stop condition at a predetermined amount for the number of steps or time. Or, the definition of the limited number of answers is defined as the condition for stopping the problem.

3.3. Formulation and definition of the problem
To solve the routing problem, before anything else it is necessary to define the environment in which the vehicle determines its path. There are several types of barrier in a natural environment. There is no way to cross some obstacles, such as mountains, valleys, deep rivers, etc. Some other obstacles can be crossed by the physical characteristics of the car, but passing through each has its own expense. For example, to cross the ground with a gradient of 20 degrees requires less energy than a 40-degree gradient. Or, for example, the vehicle uses more energy to drive in the earth’s paths than on asphalt. Therefore, the environment to be defined must have the following characteristics:
1. A random environment;
2. Some barriers are defined as too high cost and thus as barriers that cannot be passed through at all;
3. Another category of obstacles must also be defined at a lower and different cost.

In Figure 3, a sample map of the environment is displayed. Each geometric shape symbolizes a natural barrier or a particular natural feature. For example in this map:
Octagonal shapes: obstacles that cannot be crossed and cost 100 times the cost of crossing the normal path.
Seven-faced shapes: slopes between 20 and 40 degrees, with the cost of passing them 10 times the cost of crossing the normal path.
Square shapes: Slopes between 10 and 20 degrees are considered to be 7 times higher than normal paths.
Triangular shapes: The road ripples, which cost them 5 times more than ordinary tracks.
Pentagon shapes: rocky trails with a cost of crossing them 3 times more than ordinary trails.
The reason to define the costs is the abilities of the vehicle. To summarize its ability see Table 1. The vehicle is about 3 m in length, 2 m in width and 2 m in height.

Table 1. Vehicle abilities

| ability       | limits        |
|---------------|---------------|
| grading       | 100% or 45°   |
| obstacle      | 40 cm         |
| Road speed    | 40 km/h       |
| Off-road speed| 15 km/h       |
| Turning radius| 2 m           |
| Fording       | 1 m           |
Figure 3. A random created map

Now using the PSO algorithm, the optimal route is obtained through these obstacles. In order to get the path between the two points, there must actually be a curve between the two points, which is, of course so that the car avoids obstacles and reaches its destination at the lowest possible cost. For this purpose, there are several methods, the method used in this study is the use of spline.

Spline is a method for describing parametric curves. In this case, the PSO algorithm will adjust the spline parameters in order to create an optimal path between the initial and the endpoints. The spline method is to guess some points between two points and then creates a curve between these points. Given that an environment is a two-dimensional environment, there must be two functions in time, each of which is a spline, and then by eliminating the time from these two functions a path curve is achieved.

The most important part of the work is the definition of the cost function. In the cost function, the following are considered:
1. The path length is defined as a cost.
2. The obstacles that cannot be passed have huge costs.
3. Other obstacles, including slopes, unevenness of the path, etc. Due to the difficulty passing through them, the corresponding cost function is defined.
4. The radius of the possible turning radius is also a cost function.

Here the cost function is a type of constrained cost function. The length of the path is the main cost and the next three costs are constrained costs. There are several methods for combining the main cost with the binding, including the accumulative method, the multiplicative method, and the combination method. If \( z \) is the main cost function and \( v \) is a cost function, then the three methods above can be defined as follows:

Accumulative method: \( \tilde{Z} = Z + aV \)
Multiplication method: \( \tilde{Z} = Z(1 + \beta V) \)
Combination method: \( \tilde{Z} = Z(1 + \beta V) + aV \) or \( \tilde{Z} = (Z + aV)(1 + \beta V) \)

Here the multiplicative method is used to determine the overall cost function. The cost function is defined as:
Where \( L \) is the path length, \( \beta \) is a constant coefficient, and \( V \) is the cost function associated with the physical characteristics of the path defined as follows:

\[
V = (\text{Violation}_E) \times 10000 \\
+ (\text{Violation}_C) \times 10 \\
+ (\text{Violation}_S) \times 7 \\
+ (\text{Violation}_T) \times 5 \\
+ (\text{Violation}_F) \times 3 \\
+ (\text{Violation}_R) \times 5
\]

(6)

Where:
- \( \text{Violation}_E \): The obstacles that can’t be passed;
- \( \text{Violation}_C \): Slopes between 20 to 40 degrees;
- \( \text{Violation}_S \): Slopes between 10 to 20 degrees;
- \( \text{Violation}_T \): Off-road paths;
- \( \text{Violation}_F \): Rocky roads;
- \( \text{Violation}_R \): Turning radius less than 10 m.

### 3.4. Online path planning using the neural network approach

#### 3.4.1. Introduction

The neural network method is one of the most frequently-used methods for solving problems by artificial method. This method is inspired by the brain of intelligent beings, being fast and accurate, and works well in noise situations. The structure of each neural network is composed of a number of neurons, and these neurons are connected to one another with a network. Each neuron has a number of unique features:

1. Number of entries;
2. Number of outputs;
3. Neuron conversion function;
4. Neuron threshold.

In a neural network, information flows can be forward or recursive.

1. Leading flow: The flow of information is always forward and in the next layer and there is no possibility of receiving feedback from the forward layer.
2. Recursive: There is a possibility of receiving feedback from the layers ahead.

Each neuron has a special conversion function that changes the input signal and generates a new signal as output. In Figure 4, a number of converter functions that are used in neurons are displayed.
In Figure 5, signals emanating from a particular layer of neurons are combined with different coefficients and then entered into the next layer as input signals. In Equation 6, the relationship between these signals is shown.

\[ v_i = \sum_{i=0}^{i=P} w_{ji} \rightarrow y_i = f_j(v_j) \]  

(7.a)

\[ v_k = \sum_{j=0}^{j=P} w_{kj} \rightarrow y_k = f_k(v_k) \]  

(7.b)

Where \( f \) is the transfer function for each neuron. \( v \) is the input of next layer. \( y \) is the output for layers. \( i, j \) and \( k \) are layer numbers.

---

**Figure 4.** Some neuron transfer functions [15]

**Figure 5.** Sample neural network layers
In the method of training with the instructor, the examples given are presented to the system in such a way as to train the network with a coach. In a non-instructor training method, the test results are not given to the system, and the system should be trained with feedback from the environment. In the boosting system, the system does not respond perfectly, but if the system approaches the answer to this problem, then the answer to the grid is never given to the neural network.

Indeed, the coincidence in the training of a neural network is to adjust the coefficients of each of the neurons. To do so we used a method called Levenberg-Marquardt backpropagation training [16]. This method can be used for multilayer artificial neural networks.

3.4.2. Formulation and problem solving

Our goal in using the neural network approach is to make rapid decisions in the face of sudden obstacles in the vehicle’s path. In the previous section, we manually determined the robot’s path through the barriers we already knew about them. Now, with the assumption that the car starts to move along this path and faces obstacles along its path, it needs to instantly decide how to deal with these obstacles. Our goal in designing this neural network is that, when the sensors detect an obstacle in the vehicle path, this algorithm can obtain a quasi-optimal path from the right or the left of the barrier. The reason for the pseudo-optimal route is that, since the car must decide immediately, using an algorithm that finds the perfectly optimal route can be very slow and actually prevent the vehicle from being blocked before making a decision. The algorithm also decides to prevent rollover of the vehicle, due to the curvature radius of the new path by braking or not.

To solve this problem, a multi-layer neural network is simulated in MATLAB. This neural network has a hidden layer that has 20 neurons and the first layer is made up of 4 neurons. Also, the secret function of the transformation of the neurons is the softmax MATLAB function (see Figure 6), and the output function is also purelin. The algorithm selects 70% of the data as training data, 15% as test data and 15% as the verification data. The learning function used is trainlm, that uses Levenberg-Marquardt backpropagation training algorithm.

This part of the algorithm works so that a random obstacle is initially placed in the path of the vehicle. The vehicle recognizes the obstacle within a certain distance (10 m to 20 m) and then decides by neural network the optimal route to how to cross the obstacle. In fact, the input of the neural network is the cost of passing through the various paths, along with the minimum radius of the route path. Then, this algorithm chooses the path that has the lowest cost and according to the radius of the route, if the radius is less than 10, the braking instruction will be given to the vehicle. Actually, the output of this part of algorithm is the turning radius and the braking/acceleration value. For the purpose of this study the vehicle considered as a particle and the dynamic of the multi body is not concluded.

4. Discussion and numerical results

In the following figures, several scenarios have been tested and several simulations have been obtained. From resulting figures, it can be understood that the results are acceptable. Clearly, the PSO algorithm has tried to obtain the most optimal route through the barriers. The path optimality is related to the definition of the cost functions for each roughness or obstacle. So it’s not necessarily what
might be chosen at the first sight of a human being as the optimal route, but is equal to the optimal route chosen by the vehicle.

In the case of pass-through and unpredictable barriers, these barriers are not of the dynamic type, but rather fixed obstacles that are not already foreseen in the map. The vehicle only tries to avoid dealing with them, and the route chosen is not necessarily the optimal path.

Figure 7 and figure 9 show the offline routing of the vehicle. In both figures, the vehicle tries to pass through environmental barriers or avoid them to reach the target. As we described in section 3.3, some barriers have a huge cost, but the algorithm chose to pass them, to decrease the total cost.

In Figures 8 and 10, the black point is an unknown barrier that is suddenly revealed and the thin line is the pseudo optimal path, that is planned by the robot in the moment.
5. Conclusion
To conclude, in this research two path planning methods are used. The first one is the PSO. It is used for the known environment, to plan an optimal path. The second method is a neural network. The neurones have been trained to plan a pseudo-optimal path in a short time, to help the vehicle bypass the new obstacle. Both algorithms show reasonable results in simulation and helped the vehicle to find a way and reach its goal.

As all of this research was carried out in a simulated environment, it is proposed to replicate it with a real map at first, and then apply those logics to a real robotic vehicle in the real world.

6. References

[1] Thi Thoa Mac, Cosmin Copot, Duc Trung Tran, Robin De Keyser, Heuristic approaches in robot path planning: A survey, *Robotics and Autonomous Systems*, Volume 86 (2016), pp. 13-28,

[2] S.X. Yang, M. Meng, An efficient neural network approach to dynamic robot motion planning, *Neural Netw.* 138 (2000) 143–148
[3] E.A. Antonelo, B. Schrauwen, Supervised learning of internal models for autonomous goal-oriented robot navigation using reservoir computing, in: IEEE International Conference on Robotics and Automation, ICRA, Alaska, USA, 2010, pp. 2959–2964.

[4] D. Janglova, Neural networks in mobile robot motion, Int. J. Adv. Rob. Syst. 1 (2004) 15–22.

[5] S.H. Dezfulian, D. Wu, I.S. Ahmad, A generalized neural network approach to mobile robot navigation and obstacle avoidance, in: Intelligent Autonomous Systems 12, in: AISC, vol. 193, 2013, pp. 25–42.

[6] Y. Hu, S.X. Yang, A knowledge based genetic algorithm for path planning of a mobile robot, in: IEEE International Conference on Robotics and Automation, ICRA, New Orleans, LA, USA, 2004, pp. 4350–4355.

[7] S.X. Yang, Y. Hu, M.Q.H. Meng, Knowledge based GA for path planning of multiple mobile robots in dynamic environments, in: IEEE Conference on Robotics, Automation and Mechatronics, Bangkok, Thailand, 2006, pp. 1–6.

[8] B.K. Oleiwi, R. Jarrah, H. Roth, B.I. Kazem, Multi objective optimization of trajectory planning of non-holonomic mobile robot in dynamic environment using enhanced GA by fuzzy motion control and A*, in: Neural Networks and Artificial Intelligence, in: Communications in Computer and Information Science, vol. 440, 2014, pp. 34–49

[9] B. Karim, Q. Zhu, Genetic fuzzy Logic control technique for a mobile robot tracking a moving target, Int. J. Comput. Sci. Issues 10 (1) (2013) 607–613.

[10] H.C. Huang, C.C. Tsai, Global path planning for autonomous robot navigation using hybrid meta heuristic GA–PSO algorithm, in: IEEE SICE Annual Conference, SICE, Tokyo, Japan, 2011, pp. 1338–1343.

[11] S.M.R. Farshchi, S.A.N. Hoseini, F. Mohammadi, A novel implementation of Gfuzzy logic controller algorithm on mobile robot motion planning problem, Comput. Inf. Sci. 4 (2) (2011) 102–114.

[12] A.K. Karami, M. Hasanzadeh, An adaptive genetic algorithm for robot motion planning in 2D complex environments, Comput. Electr. Eng. (2015) 1–13.

[13] E. Masehian, D. Sedighizadeh, Multi–objective PSO and NPSO based algorithms for robot path planning, Adv. Electr. Comput. Eng. 10 (4) (2010) 69–76.

[14] J. Kennedy, R Eberhart, Particle swarm Optimization, Conference: Neural Networks, 1995. Proceedings., IEEE International Conference Volume 4, December 1995.DOI: 10.1109/ICCNN.1995.488968

[15] N. Agami, A. Atiya, M. Saleh, H. El-Shishiny, A neural network based dynamic forecasting model for trend impact analysis, Technological Forecasting and Social Change, Volume 76, Issue 7, 2009, Pages 952-962.

[16] C. Lv et al., Levenberg–Marquardt Backpropagation Training of Multilayer Neural Networks for State Estimation of a Safety-Critical Cyber-Physical System, in IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3436-3446, Aug. 2018.