# A New Genetic Method for Mobile Robot Navigation

Behjat Frouzandeh, Seyyed Ehsan Mahmoudi, Ali Akhavan Bitaghsir and Alireza Marandi
Electrical and Computer Engineering Dep.
University of Tehran
IRAN

*Abstract:* - The obstacle avoidance and path planning is one of the most important problem in mobile robots, especially in dynamic environments which both target and obstacles are moving. Also the ideas and algorithms for controlling and decreasing the real robot error in path is another problem that must be solved. Usual methods have two separated parts, path planning with obstacle avoidance and auto-tuning motion control.

In this paper we combined these two parts, discuss about the pursuit idea in robot motion control and show the modification with genetic algorithm to achieve a method for navigation of a two wheeled mobile robot.

All ideas such pursuit and obstacle avoidance are taken from nature, and has been executed with genetic algorithm and fuzzy logic to train an intelligent robot in dynamic environment.

## 1 Introduction

An intelligent mobile robot is a type of autonomous mobile vehicle that is able to independently plan its own route and navigate through obstacles to reach a specified destination. There is no human input or intervention. Its potential applications range among civilian (automatic cars), industrial (co-operative mobile robots in factories), military (unmanned vehicles to destroy enemy target) and fun (soccer player robots) usages.

A simplified schematic model of a mobile robot controller is depicted in figure 1. The decision unit module acts as the brain of the robot deciding what to do, according to the robot's goal and also the robot perception from its environment. It then sends a planning request to the path planner. The path planner should determine the nearest route in terms of distance or time to the target which is free from collisions. The motion controller is responsible for following the suggested plan by the path planner, according to mobile robot's low level (physical) motion limitations as closely as possible (assuming that a path plan exists). In other words, the simplified controller model introduces three abstraction levels for the mobile robot navigation. In this paper we concentrate our discussion on the path planner and motion controller modules, and call these parts "the robot navigator". Many classical methods such as path velocity decomposition [6], incremental planning [6], probabilistic approach [14], cell decomposition [1], potential field [3] and other methods are proposed for path planning in static and dynamic environments. The cell decomposition algorithm computes explicitly the configuration space of the moving robot's environment, decompose the resulting free space into cells, and then search a route in the free cell graph. This technique is suitable for static and accessible environments (the environment must be completely accessible and the obstacles may not move). The artificial potential field is another method which is well used for path planning. However, it is computationally expensive, and still does not give distance-efficient solutions. These methods and the majority of other proposed methods are best suitable for static environments; however, they may result in poor performance for dynamic environments [1,2,3,4].

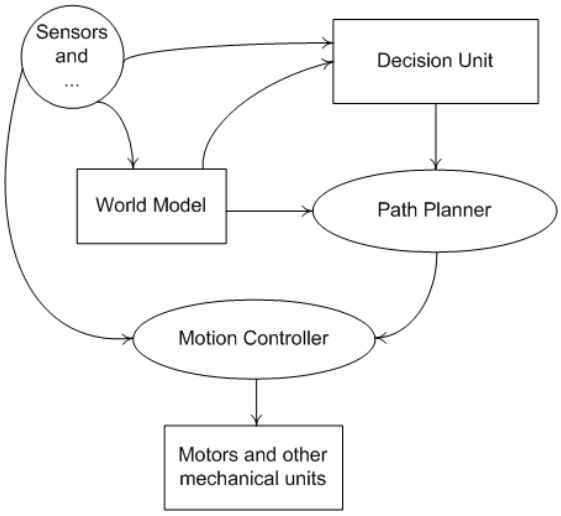


**Figure 1.** The simplified schematic model of a mobile robot controller.

In dynamic path planning problems, unlike static problems, the problem is not solvable by merely constructing a geometric path and then following this path because the proposed plan by the path planner module becomes invalid in short time (depending on the stochastic-ness of the environment) . However, some of the previous classical methods are still applicable with some modifications and extensions which are aimed at taking the effectiveness of the time dimension into account [1,9].

In recent years, the application of learning techniques in mobile robot motion planning has been widely investigated by researchers. Modifying the classical methods of path planning became a popular subject and different learning techniques have also been exploited in order to improve the performance of such methods to use them in dynamic environments. Canny and Reif [12] showed the computational complexity of these methods in their research and proved that motion planning for a point robot in a two-dimensional plane with a bounded velocity is an NP-hard problem. However, these classes of solutions are still useful methods in this domain and many researches are attempting to achieve further findings in mobile motion planning by means of such methods [6].

The motion controller module is responsible for determining the physical configuration of the mobile robot by means of the suggested route plan (from the path planner module). The challenge of motion planner's task is reflected in the fact that although a mobile robot in the plane possesses three degrees of freedom of motion, it has to be controlled by only two control inputs[1] and under nonholonomic motion constraints. This task is a well-known problem of nonlinear control and there exist many classified solutions for it such as order reduction and Lyapunov approaches [1,6].

Another class of methods for robot navigation is those for which the controller merges the motion controller and path planning modules into one. The new module is responsible for setting directly the physical configuration of the robot based on the robot's goal and perception; this module does not compute a path plan explicitly anymore. As the controller is not responsible for constructing a geometric path in an abstract level and then following this path, this idea may give us higher simplicity in the design of the robot's controller; however, the robot's navigation to the target may not be optimum in terms of time and distance. Some previous works were done using this class of robot navigation methods, with and without learning; however, none of them focus on applying this class of methods in environments containing obstacles whether moving or not. A modification of this class of methods can be applied in dynamic environments for which predicting the robot's future path is not possible. Besides, in case of computational limitations, the method proposes low complexity for fast decision making. This paper, contributes a concrete representation of our new approach by proposing the application of genetic learning method in latter class of navigation methods; also, we verify the successful application of our proposed method to real robot navigation situations. In the next section, we derive the equations involving our approach, based on the kinematic model of the robot. Applying genetic learning algorithm to the domain in addition to the implementation phase is demonstrated in Section 3. The proposed method's results are showed and discussed in Section 4; and finally in section 5, we arrive at conclusion and discuss directions for future work.

## 2 Mobile Robot Navigator

The fundamental idea for the navigation method is inspired from wild animals when pursuiting their target. This control approach is so useful for reaching a static or moving target and is also applicable in rocket motion control. In this method the nonholonomic mobile robot attempts to change (adapt) its wheels' velocity to decrease its angle with the target at each decision point.

We consider a two wheel mobile robot with the kinematic model demonstrated in figure 2 and equations 1, 2 and 3.

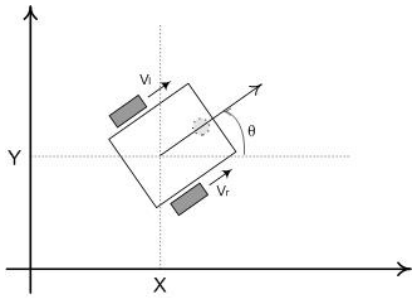


**Figure 2.** The schematic model of a nonholonomic mobile robot.

$$\frac{dx}{dt} = \frac{V_l + V_r}{2}\cos\theta \qquad (1)$$

[1] The robot's right and left wheels' velocity.

$$\frac{dy}{dt} = \frac{V_l + V_r}{2} \sin\theta \quad (2)$$

$$\omega = \frac{d\theta}{dt} = \frac{V_r - V_l}{d} \quad (3)$$

$$V = \frac{V_l + V_r}{2} \quad (4)$$

Where $V_l$ and $V_r$ are the robot's left and right wheel's velocities, respectively; $d$ is the distance between two wheels and $\theta$ is the angle between the robot's direction and $x$ axis.

The robot's goal is to reach the target at the end of the experiment; the robot's angle ($\theta$) at the target position may be arbitrary. The robot starts at a position ($x, y, \theta$), moving based on its wheels' velocity, and finally reach its target ($x', y', -$). In order to reach the target by an exponential rate[2], the desired values for $\omega$ and $V$ must be:

$$\omega = K_1 \times \textbf{\textit{AngleError}} \quad (5)$$

$$V = K_2 \times \textbf{\textit{DistanceError}} \quad (6)$$

Where ***DistanceError*** is the distance between the center of robot and target; and ***AngleError*** is the difference between the angle of robot's direction and the angle of the connecting line between the center of robot and the target. The solution to these equations is as follows:

$$V_l = K_d \times \textbf{\textit{DistanceError}} + K_a \times \textbf{\textit{AngleError}} \quad (7)$$

$$V_r = K_d \times \textbf{\textit{DistanceError}} - K_a \times \textbf{\textit{AngleError}} \quad (8)$$

These equations show that the robot tries to decrease its angle error to the target while decreasing its distance error to it. The resulting path's properties derived by these equations depends on the coefficients $K_a$ and $K_d$.

For developing this idea, it seems better to use a table of selective (according to the situation) coefficients instead of always using the same coefficients (constants). So, in order to have a good training, we split the distance error variable range into 7 disjoint intervals and angle error to 11. As it is shown in figure 3 and 4, as the angle (distance) error increases, the intervals become longer; this is because the criticality of situations regarding lower angle (distance) errors is higher in comparison to higher values.

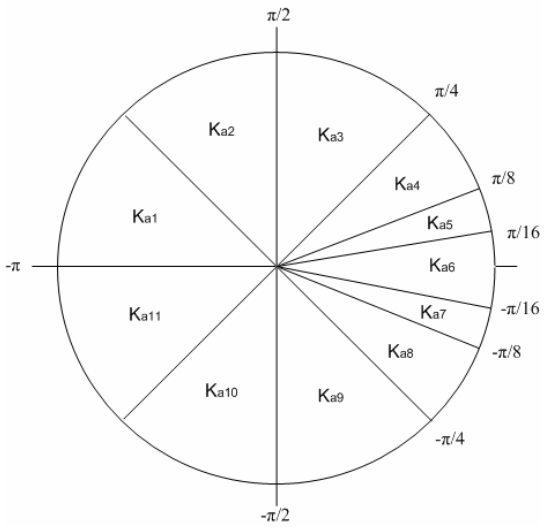


**Figure 3.** Angle error slides.

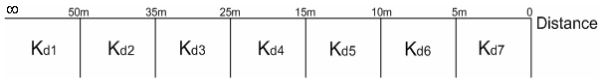


**Figure 4.** Distance error intervals.

## 2.1 Obstacle Avoidance

Now that the mobile is able is to reach the target, it should avoid the obstacles as well. For obstacle avoidance, the obstacles will be avoided locally by the robot. In other words, the robot will avoid the obstacle just after obstacle perception. For this purpose, two approached are possible. The first approach is to keep the motion equations body (equations 7, 8) unchanged, and only changing the coefficients ($K_a$ and $K_d$) appropriately when an obstacle is observed by the robot. Consequently, the appropriate values for $K_a$ and $K_d$ at each time also depends on the relative position of the robot to the newly observed obstacle. However, we used another approach for which we add new terms to the motion equations; in this approach, we approximate the robot's future path and calculate the minimum distance between the robot's center and the obstacle ($H$) in this path. Figure 4 and equations 9-12 demonstrates the calculations for evaluating $H$.

[2] The rate of distance and angle error's convergence to zero.

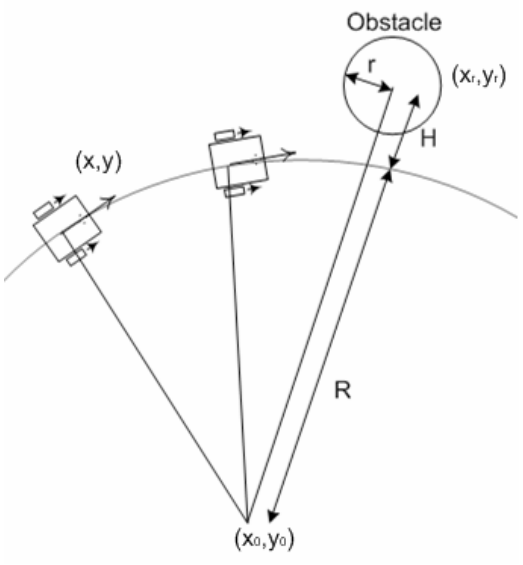


**Figure 4.** The process of calculating $H$ . ( $x_0$ and $y_0$ are the dimensions of rotation center and R is radius of rotation. )

$$R = \frac{V}{\omega} \tag{9}$$

$$x_r = x + R\sin(\theta) \tag{10}$$

$$y_r = y - R\sin(\theta) \tag{11}$$

$$H = \sqrt{(x_0 - x_r)^2 + (y_0 - y_r)^2} - R \tag{12}$$

In some special occasions R is infinitive which occurs when $V_l = V_r$; in such case, robot's path is a straight line so $H$ is the distance of this line from the obstacle.

Finally, the following equations will take $H$ into account for robot's obstacle avoidance while getting closer to the target.

its fitness. This penalty makes the algorithm to choose gains such they avoid obstacles.

Algorithm is simulated in a random area with random sized obstacles. Obstacles are considered circles. The simulation is done 20 times for each robot from random positions and the final fitness is the sum of each fitness function.

The next step is to code each state to a binary chromosome. Each gain is a real number less than one and is coded as 32-bit precision binary number. So each of the chromosome contains 11 $K_a$ gains,7

$$V_l = K_d \times \textbf{\textit{DistanceError}} + K_a \times \textbf{\textit{AngleError}} + \frac{K_o \times H}{r'} \tag{13}$$

$$V_r = K_d \times \textbf{\textit{DistanceError}} + K_a \times \textbf{\textit{AngleError}} - \frac{K_o \times H}{r'} \tag{14}$$

Where $r' = r + \frac{d}{2}$ and $r$ is the obstacle radius and $d$ is the distance between robot's wheels. (Sign of the $H$ is important and should be calculated in relation to $V_l - V_r$) . This approach first estimates the curve (as a circle) on which the robot will move in its path; if it passes through the obstacle it tries to change its direction to avoid the obstacle.

We set 10 different situations (intervals) for choosing a suitable $K_o$ in attention to our $H$ and $r'$. The $K_o$ value table is shown in Figure 5. For the training part we will exploit genetic algorithm to learn the optimum values for these 8 coefficients in addition to the previous 18 (11 + 7) coefficients.

| H | -∞ | -r'* | -0.75r' | -0.5r' | -0.25r' | 0 | 0.25r' | 0.5r' | 0.75r' | r' | ∞ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Ko | 0 | Ko1 | Ko2 | Ko3 | Ko4 | Ko5 | Ko6 | Ko7 | Ko8 | | 0 |

* r'=r+d/2

**Figure 5.** $K_o$ values in terms of $H$ intervals.

## 3 Genetic Implementation

Fitness function used for this problem is based on two parameters. First, the time it is taken to reach the target. If the robot takes too long to reach target (doesn't reach target in a limited time) a large number is considered as fitness. Another parameter is a penalty for obstacle collisions. Robot is allowed to pass through the obstacle but for each time step which the robot is in obstacles area, fix number is added to its fitness. This penalty makes the algorithm to choose gains such they avoid obstacles.

$K_d$ and 8 $K_0$ gains which makes each chromosome 832 bit binary string. The detail of how these genes affect the performance of robot is not important to genetic algorithm.

Used Genetic operators are crossover and mutation. Application of these operators is by roulette-wheel selection strategy, which fitter individuals have more chance for reproduction and the chance for other individuals to be used is not zero.

Crossover is the process of choosing two individuals and making new chromosomes from them.

Crossover operation chosen for this purpose is multiple-point crossover. Because of the long chromosome that is used (Single point crossover changes the population slowly). Breaking points are choused randomly within 832 possible positions in two chromosomes.

Mutation is process of choosing one individual and changing its chromosome randomly. The change is in the form of changing a bit in chromosome. Mutation is for exploring the search space. Mutation is included as an operator to recover a gene that may have been lost in the population. It is not the primary operator in the algorithm, which is reflected in the low probability (0.01) of mutation.

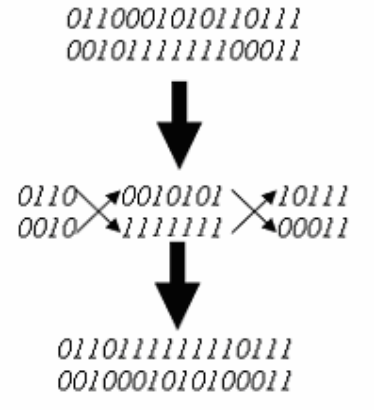


**Figure 6.** The crossover process.

The population size used was 50 while number of generations was 200.

## 4 Simulation Results

Figure 7 shows the convergence of individuals fitness function after simulation.

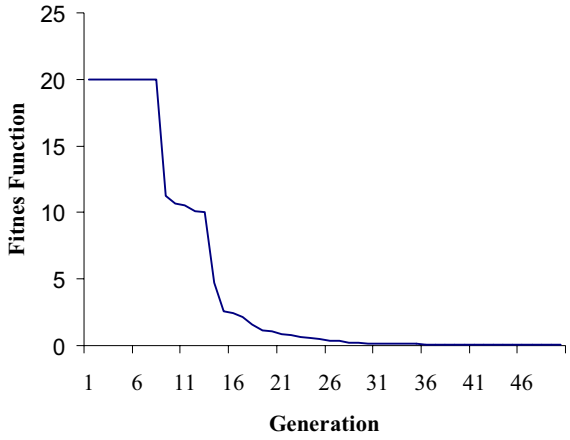


**Figure 7.** Convergence of fitness function.

In this Section, two scenarios of robot navigation are discussed. In the first scenario there's no obstacle in the environment; meaning that, we want to test only the target reaching ability of the robot. Figure 8 shows the simulation result for the fittest individual in an environment with no obstacle. As it is shown, the robot will smoothly adjust its wheels' velocity to reach the target. The robot has positioned on a random point of the field. Since there is no obstacle on the field, the path curve has a uniform shape.

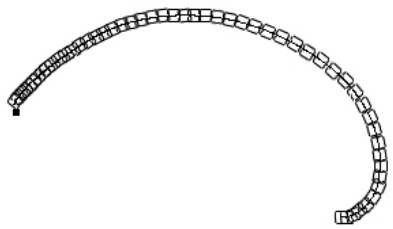

**Figure 8.** Simulation result in an environment with no obstacle; the black square is the target.

The second scenario considers environments containing several obstacles.

Figure 8 shows the simulation of the best individual after the production of generations from a random position. The initial angle of robot is 0 and its initial position is (20,-1) and target position is (0, 0). From the previous arguments, it is expected that as the robot approaches an obstacles it changes its direction to avoid collision.

As it is clear in the picture, the path to the target is not the optimum one, because the obstacles are observed and considered locally by the robot. The points in which the robot has changed the direction are the points that the obstacle enters the robot's detection area. This has made the path a bit non-smooth.

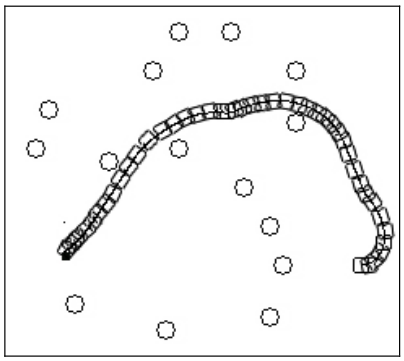

**Figure 9.** The fittest individual after running the genetic method in an environment with several obstacles.

## 5 Conclusion

In this paper, we have proposed a new method for mobile robot navigation concerning the application of genetic algorithm for determining appropriate coefficients in the motion equations of the robot. The experimentations verified the successful result of the proposed method in different environments. Also, the results of experiments are hope-giving to complete these ideas for achieving a more optimized and robust method for an auto-tuning mobile robot controller in static and dynamic environments. The learning algorithms' settings may not be the same in different environments and the designer should decide upon it, based on the environment's characteristics.

A potential future research direction is to change the tables of the equation coefficients into fuzzy functions in order to approve the performance and smoothness of the resulting path. Another direction is to tune the learning algorithm for faster convergence into an appropriate path.

Another potential work is to apply genetic programming [8] to the domain in order to calculate a formula for wheels velocities directly; however it has its own complications.

*References*


[1] Latombe J.C. "Robot Motion Planning"Kluwer Academic Publishers,1991.

[2] H. Noborio, T. Nauiwa, and S. Arimoto, " feasible motion-planning algorithm for a mobile robot in a quadtree representation" in proceedings of th IEEE International Conference on Robotics and Automation, 1993, pp.327-332.

[3] J.O. Kim and P.K. Khosla, "Real-time obstacle avoidance using harmonic potential functions", IEEE Transaction on Robotics and Automation, Vol.8, No.3, pp.338-349, June 1992.

[4] T.J. Pan and R.C Luo, "Motion planning for mobile robots in a dynamic environment with moving obstacles", Proceedings of the IEEE International Conference on Robotics and Automation, 1990, pp.578-583.

[5] M. Walker, C.H. Messom, "A comparison of genetic programming and genetic algorithms for auto-tuning mobile robot motion control", Proceedings of The First IEEE International Workshop on Robotics and Automation, Jan 2002, pp.507 -509.

[6] D.K. Partihar, K. Deb, A. Ghosh, "A genetic-fuzzy approach for mobile robot navigation among moving obstacles", International journal of Reasoning, vol 20, 1999, pp.145-172.

[7] Lefeber. A. A. J. "Tracking Control of Nonlinear Mechanical Systems", PHD Thesis, Universitate Twente, 2000.

[8] J. R. Kozza, "Genetic Programming: on the Programming of Computers by Means of Natural Selection", The MIT Press, 1992.

[9] T. C. Chin and X. M. Qi, " Integrated Genetic Algorithm Based Optimal Fuzzy Logic Contrller Design", Proceeding of the Fourth International Conference on Control Automation, Robotics and Vision, 1996, pp. 563-567.

[10] C.H. Leung and A.M.S. Zalzala, "A GA Solution for the Motion Control of Wheeled Robotic System in Dynamic Environment" International Conference on Control '94. Volume 1, Mar 1994, pp: 760 -764.

[11] Wei-Ming Lee, Han-Pang Huan, "Stablization of Nonholonomic Mobile Rbots by a GA-Based Fuzzy Sliding Mode Control" Fuzzy Systems Symposium, 1996. 'Soft Computing in Intelligent Systems and Information Processing'., Proceedings of the 1996 Asian pp: 388 -393.

[12] J. Canny, J. Reif, "New lower bound techniques for robot motion planning problems", Proceedings 27th IEEE symposium on Foundations of Computer Science, 1987, pp:49-60.

[13] T. Shibata,T. Fukuda,;K. Tanie, "Fuzzy Critic for Intelligent Planning by Genetic Algorithm" Workshop Proceedings. IEEE 2nd International Workshop on Design and Operations of Intelligent Factories, 27-29 Sept. 1993, pp 78-85.

[14] C.F. Olson,"Probabilistic sel-localizatin for mobile ronot", IEEE transaction on Robotics and Automation, vol 16, 2000.