GENETIC ALGORITHM-BASED NAVIGATION STRATEGY FOR MOBILE TERRESTRIAL ROBOTS

Átila V. F. M. de Oliveira*, Marcelo A. C. Fernandes*

*Department of Computer Engineering and Automation – DCA
Federal University of Rio Grande do Norte – UFRN
Natal, RN, Brazil

Emails: avfmo.engcomp@gmail.com, mfernandes@dca.ufrn.br

Abstract— This article proposes a new dynamic planning navigation strategy for use with mobile terrestrial robots. The strategy was applied to situations in which the environment and obstacles were unknown. After each displacement event, the robot replanned its route using a control algorithm that minimized the distance to the target and maximized the distance between the obstacles. Using a spatial localization sensor and a set of distance sensors, the proposed navigation strategy was able to dynamically plan optimum routes that were free of collisions. Simulations performed using different types of environment demonstrated that the technique offers a high degree of flexibility and robustness, and validated its potential use in real applications involving mobile terrestrial robots.

Keywords— Genetic Algorithms, Mobile Robots, Atonomous Navigation.

1 Introduction

The evolution of robotics research in the last fifteen years has had an important influence on human activities in many sectors of society. In the case of mobile robotics, the main objective is to develop autonomous intelligent robots that are able to effectively plan their movements in static or dynamic environments that are unknown (Siegwart et al., 2011).

Methods involving Artificial Intelligence (AI) are amongst the techniques that can be used for this purpose, and there have been several reports of new navigation strategies showing a high level of precision in terms of avoidance of obstacles and achievement of optimum routes. In the area of AI, many navigation techniques employ fuzzy logic, artificial neural networks (ANNs), genetic algorithms (GAs), or combinations of these procedures (Martinez-Soto et al., 2012) (Juang and Chang, 2011). Techniques based on GAs have been widely used due to their robustness, and in most cases have been able to provide optimum routes in both static and dynamic environments (Ismail AL-Taharwa and Al-Weshah, 2008) (Yang et al., 2007) (Yun et al., 2011) (Shi and Cui, 2010) (Shamsinejad et al., 2010) (Tian and Collins, 2004) (Tuncer and Yildirim, 2012) (Cosío and neda, 2004).

This work proposes a new dynamic planning navigation strategy based on a genetic algorithm, called the Dynamic Planning Navigation Algorithm optimized with GA (DPNA-GA). The objective of this strategy was to be able to identify an optimum collision-free route to a spatially defined objective. The DPNA-GA employs a set of distance sensors and a spatial localization sensor in order to reach the objective, without any a priori knowledge of the environment.

2 Description of the Navigation Strategy

It was assumed that the robot possesses a localization sensor that returns its spatial position, $p^R = \left(x^R, y^R\right)$, and a set of n evenly distributed distance sensors. The navigation strategy based on the DPNA-GA generates a route composed of M local displacements in order to reach the final objective, $p^{fo} = \left(x^{fo}, y^{fo}\right)$. In each m-th displacement event, there is a local objective,$p^{ol}(m) = \left(x^{ol}(m), y^{ol}(m)\right)$, to which the robot moves.

The selection of the local objective, $p^{ol}(m)$, in each m-th event, is performed by a GA that uses the current position of the robot, $p^R(m)$, taking into consideration the distance to the final objective, p^{fo}, and the obstacles detected by the n distance sensors. All the positions $p^R(m)$ already visited by the robot up to the m-th displacement are stored in vector centers, $\mathbf{p}^R$, expressed by

$$\mathbf{p}^R = \begin{bmatrix} p^R(0) \\ p^R(1) \\ \vdots \\ p^R(m-1) \\ p^R(m) \end{bmatrix} = \begin{bmatrix} \left(x^R(0), y^R(0)\right) \\ \left(x^R(1), y^R(1)\right) \\ \vdots \\ \left(x^R(m-1), y^R(m-1)\right) \\ \left(x^R(m), y^R(m)\right) \end{bmatrix} \quad (1)$$

and are also used in the GA optimization function, avoiding searches in areas that have already been investigated.

The algorithm terminates when the current position of the robot is the same as the final objective, with $p^R(m) = \left(p^{fo} \pm \epsilon\right)$, where ϵ is a tolerance factor, or when the number of displacement

events exceeds a maximum value, M_{max}. The steps processed by the DPNA-GA are presented in Algorithm 1, and are described in detail in the following sections.

Algorithm 1 ANPD-AG

1: $m = 0$
2: $\mathbf{p}^R = \left[p^R(0)\right]$
3: **while** $p^R(m) \neq \left(p^{fo} \pm \epsilon\right)$ **E** $m < M_{max}$ **do**
4: $\quad \mathbf{p}^{DP}(m) = Scanning$
5: $\quad \mathbf{p}^O(m) = ObstaclesDetection\left(\mathbf{p}^{DP}(m)\right)$
6: $\quad p^{lo}(m) = SearchLocalObjective\left(\mathbf{p}^O(m)\right.,$ $\left.\mathbf{p}^{DP}(m), \mathbf{p}^R\right)$
7: $\quad p^R(m+1) = Displacement\left(p^{ol}(m)\right)$
8: $\quad \mathbf{p}^R = \left[\mathbf{p}^R, p^R(m+1)\right]^T$
9: $\quad m = m + 1$
10: **end while**

2.1 Scanning

In the scanning step (line 4 of Algorithm 1), the DPNA-GA forces the robot to perform a 360° scan of the environment around its axis. From this scan, each j-th sensor, during the m-th event, returns an analog signal, $s_j(m)$, proportional to the distance limit, d_{max}, in which

$$s_j(m) = \begin{cases} d_j(m) \text{ for } d_j(m) \leq d_{max} \\ \\ d_{max} \text{ for } d_j(m) > d_{max} \end{cases} \quad (2)$$

where d_j is the distance measured by the j-th sensor attached to the robot.

The scanning resolution, θ_r, can be expressed by

$$\theta_r = \frac{360°}{n \cdot p}, \quad (3)$$

where $p - 1$ represents the number of angular displacements the robot can make in its axis, considering the reduced resolution due to a small number of sensors. The number of displacements can be easily calculated using

$$p = \frac{360°}{n \cdot \alpha}, \quad (4)$$

where α is the value of the angular displacement. At the end of the scanning process, the DPNA-GA generates a polygon, known here as a delimiting polygon (DP), composed of a set of K points, expressed by the vector

$$\begin{aligned} \mathbf{p}^{DP}(m) &= \begin{bmatrix} p_0^{DP}(m) \\ \vdots \\ p_k^{DP}(m) \\ \vdots \\ p_{K-1}^{DP}(m) \end{bmatrix} \quad (5) \\ &= \begin{bmatrix} \left(x_0^{DP}(m), y_0^{DP}(m)\right) \\ \vdots \\ \left(x_k^{DP}(m), y_k^{DP}(m)\right) \\ \vdots \\ \left(x_{K-1}^{DP}(m), y_{K-1}^{DP}(m)\right) \end{bmatrix} \end{aligned}$$

where $K = p \times n$ and (x_k, y_k) represent the plane coordinates of the k-th point associated with the DP. This polygon is used to delimit the search space of the genetic algorithm. In other words, the individual points generated within the polygon have greater value than points generated outside it. Meanwhile, the ability of each individual point is not only determined by whether or not it is within the polygon. Also taken into account are the distances between the point generated and the obstacles detected by the scan, as well as other factors. Figure 1 illustrates the polygon generated by the DPNA-GA for a case where $n = 4$ and $\alpha = 30°$.

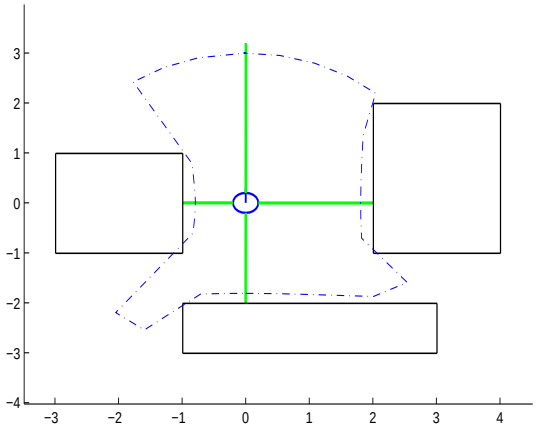

Figure 1: Example of the delimiting polygon (blue line) for a case where $n = 4$ and $\alpha = 30°$.

2.2 Detection of obstacles

Once the scanning stage is complete, the detection of obstacles phase begins (line 5 of Algorithm 1), in which a virtual polygon (VP) is generated in addition to the delimiting polygon. The VP considers a circumference centered on the robot (at position p^R), with a radius r_{PV} that is slightly shorter than the maximum range (d_{max}) of the

sensors. Then,

$$r_{PM} = d_{max}(1-\eta), \tag{6}$$

where the factor η is limited to $0,9 \leq \eta < 1$. The role of the VP is to detect only those points of the DP that are associated with the obstacles, denoted p^O. Hence, after this step, a new set of L points is generated, represented by the vector

$$\begin{aligned} \mathbf{p}^O(m) &= \begin{bmatrix} p_0^O(m) \\ \vdots \\ p_l^O(m) \\ \vdots \\ p_{L-1}^O(m) \end{bmatrix} \\ &= \begin{bmatrix} \left(x_0^O(m), y_0^O(m)\right) \\ \vdots \\ \left(x_l^O(m), y_l^O(m)\right) \\ \vdots \\ \left(x_{L-1}^O(m), y_{L-1}^O(m)\right) \end{bmatrix}, \end{aligned} \tag{7}$$

where

$$p_l^O(m) = p_k^{DP}(m) \text{ se } f_{ed}\left(p^R(m), p_k^{DP}(m)\right) \leq r_{PV}, \tag{8}$$

and $L \leq K$. The function $f_{ed}(\cdot,\cdot)$, which calculates the Euclidean distance between any two points, can be described by

$$f_{ed}(p_i, p_b) = \sqrt{(x_i - x_b)^2 + (y_i - y_b)^2}. \tag{9}$$

Figure 2 illustrates an example of the VP (green circle) and the set of points $\mathbf{p}^O$ (red asterisks).

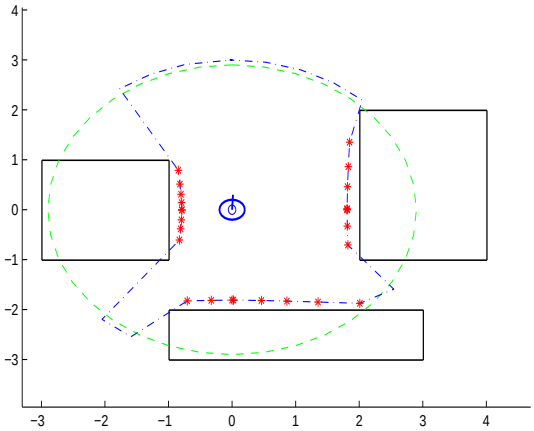

Figure 2: Example for a case where $\eta = .01$, illustrating the VP (green line) and the set of detected points ($\mathbf{p}^O$, red asterisks) that surround the obstacles.

2.3 Search for the local objective

In this step, (line 6 of Algorithm 1), the proposed navigation strategy uses a GA to find a possible local objective, p^{ol}, to which the robot can move. For each m-th displacement event, the GA, with a new population, is executed for H generations. The individuals are characterized by the vector

$$\begin{aligned} \mathbf{p}^{GA}(h,m) &= \begin{bmatrix} p_0^{GA}(h,m) \\ \vdots \\ p_j^{GA}(h,m) \\ \vdots \\ p_{J-1}^{GA}(h,m) \end{bmatrix} \\ &= \begin{bmatrix} \left(x_0^{GA}(h,m), y_0^{GA}(h,m)\right) \\ \vdots \\ \left(x_j^{GA}(h,m), y_j^{GA}(h,m)\right) \\ \vdots \\ \left(x_{J-1}^{GA}(h,m), y_{J-1}^{GA}(h,m)\right) \end{bmatrix}, \end{aligned} \tag{10}$$

where $p_j^{GA}(h)$ represents the j-th individual of the population with size J, associated with the h-th generation of the m-th displacement of the robot. In each generation, h, all the individual points are generated according to the non-linear restriction given by

$$\begin{aligned} r_d \geq & \left(x_j^{GA}(h,m) - x^R(m)\right)^2 \\ & + \left(y_j^{GA}(h,m) - y^R(m)\right)^2. \end{aligned} \tag{11}$$

This restriction limits the individual points of the population to a circumference with radius r_d, centered on the position of the robot in the m-th instant, $p^R(m)$.

The evaluation function associated with the j-th individual point of the h-th generation of the m-th displacement is expressed by

$$\begin{aligned} g_j(h,m) = & \; d_j^{fo}(h,m) + \beta(m)\frac{1}{d_j^o(h,m)} \\ & + \beta(m) A_j(h,m) C_j(h,m) \end{aligned} \tag{12}$$

where $d_j^{fo}(h,m)$ is the Euclidean distance between the j-th point of the h-th generation and the final objective, p^{fo}, characterized as

$$d_j^{fo}(h,m) = f_{ed}\left(p_j^{GA}(h,m), p^{fo}\right), \tag{13}$$

where $d_j^o(h,m)$ is the shortest Euclidean distance between the j-th point of the h-th generation and all the L obstacles found, expressed as

$$\begin{aligned} d_j^o(h,m) = & \min f_{ed}\left(p_j^{GA}(h,m), p_l^O(m)\right) \\ & \text{for } l = 0, \ldots, L-1. \end{aligned} \tag{14}$$

The variables $\beta(m)$, $C_j(h,m)$ and $A_j(h,m)$ can be characterized as penalty factors added to each j-th individual of the GA. When no obstacle is found in the m-th displacement event ($L = 0$), it

is assumed that the optimum evaluation function is simply $d_j^{fo}(h,m)$, so that

$$\beta(m) = \begin{cases} 1 \text{ for } L \neq 0 \\ \\ 0 \text{ for } L = 0 \end{cases} \tag{15}$$

Starting from the principle that the circumferences with radius r_d, centered in the center vector $\mathbf{p}^R$, are areas that have already been visited, the penalty factor, $C_j(h,m)$, can be characterized as

$$C_j(h,m) = \begin{cases} 1 & \text{if} \\ & \nexists i \in \{0,\cdots,m-1\}: \\ & f_{ed}\left(p_j^{GA}(h,m), p^R(i)\right) < r_d \\ \\ Z & \text{if} \\ & \exists i \in \{0,\cdots,m-1\}: \\ & f_{ed}\left(p_j^{GA}(h,m), p^R(i)\right) < r_d, \end{cases} \tag{16}$$

where Z is a relatively large number. Hence, an individual point $p_j^{GA}(h,m)$, located within one of the m circumferences with radius r_d, centered in the center vector, $\mathbf{p}^R$, would be positively penalized, reducing its chances of being selected. Finally, the penalty, $A_j(h,m)$, refers to individual points, $p_j^{GA}(h,m)$, located outside the DP, where

$$A_j(h,m) = \begin{cases} 1 & \text{if } \in PD \\ \\ \infty & \text{if } \notin PD. \end{cases} \tag{17}$$

Individual points that are subject to this penalty will have minimal possibility of surviving until the next generation. Figure 3 illustrates the calculation of the evaluation function for a j-th point $p_j^{GA}(h,m)$.

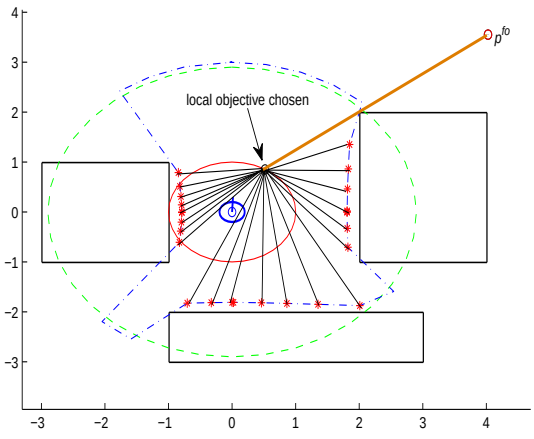

Figure 3: Example illustrating calculation of the evaluation function for a j-th individual point, $p_j^{GA}(h,m)$, in relation to the final objective, p^{fo}, and the obstacles, $\mathbf{p}^O$.

The evaluation function presented in Equation 12 follows the same principle as the potential fields technique (Siegwart et al., 2011), in which $d_j^{fo}(n,m)$ (the Euclidean distance between the j-th point and the final objective, p^{fo}) represents a force of attraction to the final point. The term $\frac{1}{d_j^o(n,m)}$ (the shortest Euclidean distance between the j-th individual point and all the points associated with obstacles) represents the greatest repulsive force between the j-th individual point and all the obstacles found. After N generations, the smallest evaluation point is chosen as the local objective, $p^{lo}(m)$, associated with the m-th displacement event.

2.4 Displacement

The displacement step (line 7 of Algorithm 1) is characterized by movement of the robot to the local objective found in the previous step. After this movement, a new center point$p^R(m+1)$ is generated, described by

$$p^R(m+1) \neq \left(p^{lo}(m) \pm \epsilon\right) \tag{18}$$

where ϵ is an admissible tolerance related to the local objective. This tolerance can be most useful in the case of robots that have restricted movement, such as non-holonomic robots (Siegwart et al., 2011). Figure 4 presents a sequence of $M = 6$ displacements up to the final point, p^{fo}.

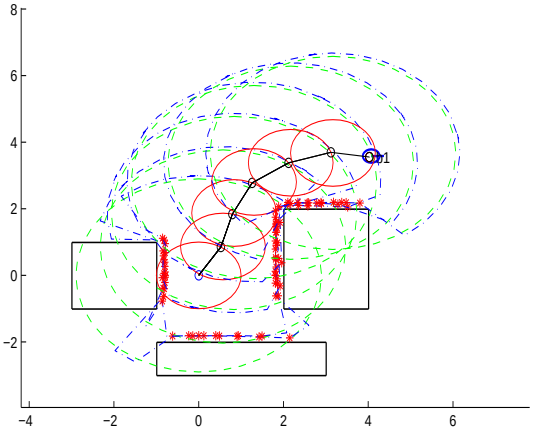

Figure 4: Example illustrating the displacements ($M = 6$) made by the robot up to the final point, p^{fo}.

3 Results and Simulations

Simulations with different environments were used to validate use of the DPNA-GA and compare its performance with other procedures. The simulations were implemented in Matlab employing an updated version of the iRobot Create toolbox

(Esposito et al., 2011), simulating a circular nonholonomic robot with differential action, possessing four distance sensors with 90° spacing. Table 1 presents the parameters utilized in the simulations. The results are presented in Figures 5 – 9, which show the routes followed by the robot (continuous black lines), the m displacement events (circles on the route lines), and the DP associated with each displacement (dashed blue lines).

Table 1: Parameters employed in the simulations.

Number of distance sensors (n)	4
Sensor range (d_{max})	3 m
Angular displacement (α)	30°
Number of generations (H)	30
Population size (J)	30
Radius (r_d)	1 m
Z (equation 16)	1000

Figure 5 shows the displacement of the robot in a moderately complex environment, where the robot required $M = 24$ displacement events in order to arrive at its destination. It can be seen that in the first instance ($0 \leq m \leq 10$), the DPNA-GA gave a sub-optimal route in which the robot was strongly attracted to the target point. However, for $m > 11$ the DPNA-GA provided an optimum route to the destination.

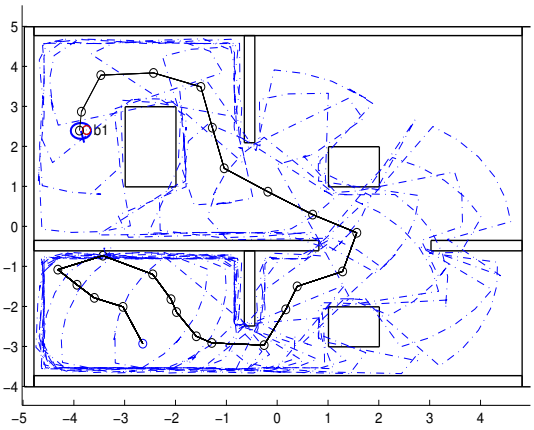

Figure 5: Displacement of the robot in a moderately complex environment. In a route that was close to the optimum, the robot required $M = 24$ displacement events.

For the scenario shown in Figure 6, the robot was required to enter a type of U-shaped garage. In this case, the DPNA-GA required approximately $M = 18$ displacement events and provided the optimum path. In the case of the scenario shown in Figure 7, the robot was faced with a situation opposite to that of the previous example, in which it was required to leave a garage. Here, the robot required $M = 14$ displacement events, and once again followed the optimum route.

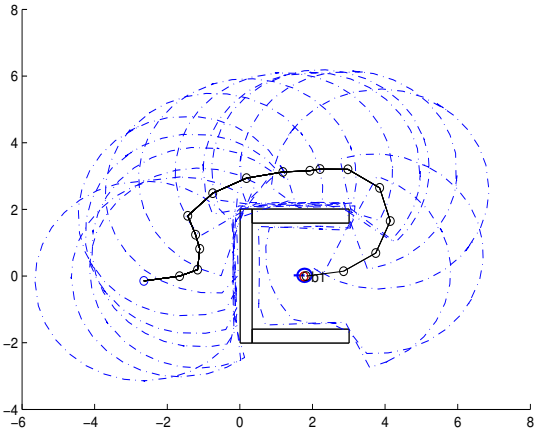

Figure 6: Displacements required for the robot to enter a U-shaped environment using an optimum route ($M = 18$).

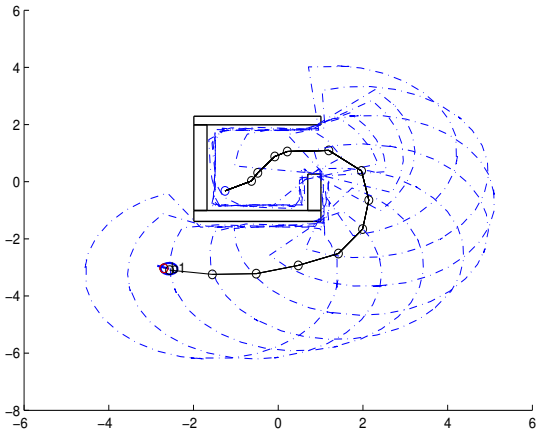

Figure 7: Displacements required for the robot to leave a U-shaped environment using an optimum route ($M = 14$).

The scenarios illustrated in Figures 8 and 9 show the performance of the navigation algorithm in the case of open environments with randomly scattered obstacles. Here, the robot followed routes that were very close to optimum, requiring $M = 17$ and $M = 18$ displacements, respectively.

4 Conclusions

A new strategy based on a genetic algorithm was developed in order to implement a dynamic planning navigation scheme for use with mobile terrestrial robots that have no a priori knowledge of the

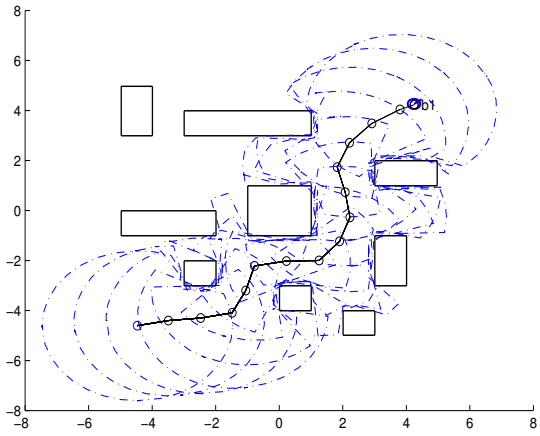

Figure 8: Displacement of the robot in an open environment with randomly scattered obstacles. In a route that was close to optimum, the robot required $M = 17$ displacement events.

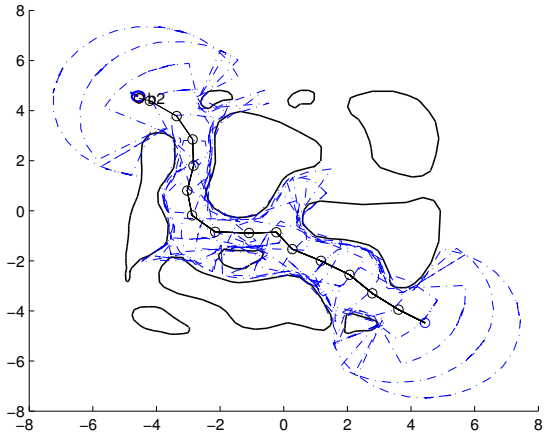

Figure 9: Displacement of the robot in an open environment with randomly scattered round obstacles. In a route that was close to optimum, the robot required $M = 18$ displacement events.

environment. The performance and robustness of the technique, known as the Dynamic Planning Navigation Algorithm optimized with GA (DPNA-GA) was evaluated using simulations representing different environments. The results indicated that the DPNA-GA should be suitable for use in practical tests with terrestrial robots.

References

Cosío, F. A. and neda, M. P. C. (2004). Autonomous robot navigation using adaptive potential fields, *Mathematical and Computer Modelling* **40**(9 - 10): 1141 – 1156.

Esposito, J. M., Barton, O. and Kohler, J. (2011). Matlab toolbox for the irobot create, www.usna.edu/ Users/ weapsys/ esposito/ roomba.matlab/.

Ismail AL-Taharwa, A. S. and Al-Weshah, M. (2008). A mobile robot path planning using genetic algorithm in static environment, *Journal of Computer Science* **4**(4): 341–344.

Juang, C.-F. and Chang, Y.-C. (2011). Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments, *Fuzzy Systems, IEEE Transactions on* **19**(2): 379–392.

Martinez-Soto, R., Castillo, O., Aguilar, L. and Baruch, I. (2012). Bio-inspired optimization of fuzzy logic controllers for autonomous mobile robots, *Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American*, pp. 1–6.

Shamsinejad, P., Saraee, M. and Sheikholeslam, F. (2010). A new path planner for autonomous mobile robots based on genetic algorithm, *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, Vol. 8, pp. 115–120.

Shi, P. and Cui, Y. (2010). Dynamic path planning for mobile robot based on genetic algorithm in unknown environment, *Control and Decision Conference (CCDC), 2010 Chinese*, pp. 4325–4329.

Siegwart, R., Nourbakhsh, I. and Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots*, Intelligent Robotics and Autonomous Agents, Mit Press.

Tian, L. and Collins, C. (2004). An effective robot trajectory planning method using a genetic algorithm, *Mechatronics* **14**(5): 55 – 470.

Tuncer, A. and Yildirim, M. (2012). Dynamic path planning of mobile robots with improved genetic algorithm, *Computers & Electrical Engineering* **38**(6): 1564 – 1572.

Yang, S., Fu, W., Li, D. and Wang, W. (2007). Research on application of genetic algorithm for intelligent mobile robot navigation based on dynamic approach, *Automation and Logistics, 2007 IEEE International Conference on*, pp. 898–902.

Yun, S. C., Parasuraman, S. and Ganapathy, V. (2011). Dynamic path planning algorithm in mobile robot navigation, *Industrial Electronics and Applications (ISIEA), 2011 IEEE Symposium on*, pp. 364–369.