An Integrated Approach for Intelligent Path Planning and Control of Mobile Robot in Structured Environment

Duško Katić, Aleksandar Ćosić, Marko Šušić and Stevica Graovac

Abstract This chapter consists of different approaches for advanced integrated path planning, navigation, guidance and tracking control of mobile service robots in structured environments. As first topic, an algorithm for robot path planning in known environments with aim to generate collision free path from starting to destination point, is considered. It is assumed that all obstacles are static. The path planning problem is solved using particle swarm optimization algorithm. The second topic deals with control problems, providing a method which enabled the mobile robot guidance along a given path. This problem can be divided into two problems: generating trajectory from given path and trajectory tracking. Solution of the first problem is provided using radial basis neural networks, while solution of the second problem is provided using mobile robot guidance controller based on algorithms of rocket homing. Proposed algorithms are pure pursuit and deviated pursuit navigation algorithms. Experimental and simulation results show the efficiency of the proposed approach.

Keywords Path planning · Navigation · Tracking control · Guidance · Mobile robot

D. Katić (✉) · A. Ćosić · M. Šušić
Robotics Laboratory, Mihailo Pupin Institute, University of Belgrade, Volgina 15, 11060 Belgrade, Serbia
e-mail: dusko.katic@pupin.rs

A. Ćosić
e-mail: aleksandar.cosic@pupin.rs

M. Šušić
e-mail: marko.susic@pupin.rs

S. Graovac
Faculty of Electrical Engineering, University of Belgrade, Bulevar Kralja Aleksandra 73, 11000 Belgrade, Serbia
e-mail: graovac@etf.bg.ac.rs

D. Pisla et al. (eds.), *New Trends in Medical and Service Robots*, Mechanisms and Machine Science 16, DOI: 10.1007/978-3-319-01592-7_12,

1 Introduction

The contemporary wheeled mobile robots have great potential applications in various potential areas as service robots for elderly persons, industrial robots for transportation, construction and planet exploration robots. However, these robots should have the ability of moving in complex and unknown environments as ambient intelligent service robots. Hence, this ability must be related on complex sensor systems and intelligent control algorithms. These systems are expected to be widely utilized in the future in human everyday life, but still many challenges remain to be overcomed. The problems of design of such advanced intelligent mobile robots are related to the success of solving complex cognitive control tasks (advanced perception and communication, environmental understanding, path planning, adaptation to variable environmental conditions). In that sense, the main tasks in mobile robotics research are development of algorithms for advanced perception, simultaneous localization and mapping (SLAM), path planning, spatial reasoning, autonomous navigation and intelligent tracking control in unknown, unpredicted dynamic environments.

This chapter is organized in the following way. Section 2 gives a brief state of the art regarding methods and techniques of advanced path planning and tracking control of autonomous mobile robots. Kinematic model of the mobile robot system is presented in Sect. 3. Section 4 is dedicated to synthesis of advanced path planning and tracking control algorithms. As solution for path planning problem, special Particle Swarm Optimization (PSO) algorithm is proposed. As solution for motion tracking control problem, self-guidance controller is designed. The architecture of the experimental mobile robotic setup is described in Sect. 5. In Sect. 6, simulation and experimental results show the performance of the proposed path planning and path following control algorithms. Section 7 gives concluding remarks and notes for future research.

2 State of the Art

Planning the path is an essential phase before actually accomplishing the preferred trajectory by controlling the motion. The main aim is generating collision free path in space with obstacles by optimizing a performance criterion such as distance, time or energy, where distance being the most commonly adopted criterion (Laumond 1998; LaValle 2006). Such a complex problem can be divided into several simpler problems. The first is generation of collision free path in space with obstacles. It is assumed that all obstacles are static and that their positions are known. Solution of the problem is given with NP–hard algorithm, given in (Canny 1988; Raja and Pugazhenthi 2012). There are two different approaches for mobile robot motion planning (Masehian and Sedighizadeh 2007), classic and heuristic. The classic approaches suffer from many drawbacks, such as high time complexity

in high dimensions and trapping in local minima, which makes them inefficient in practice. In order to improve the efficiency of classic methods, many heuristic and meta-heuristic algorithms are used (Masehian and Sedighizadeh 2007): Simulated Annealing (SA), Artificial Neural Networks (ANN), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony (ACO), Honey Bee Mating Optimization (HBMO), Simulated Annealing (SA), Stigmergy, Wavelet Theory, Fuzzy Logic (FL) and Tabu Search (TS). Heuristic algorithms do not guarantee to find a solution, but if they do, are likely to do much faster than deterministic methods.

In this chapter, trajectory generation on off-line path planning of robots is realized in environments where complete information about stationary obstacles and trajectory of moving obstacles are known. Hence, special PSO algorithm is used. PSO represents multi-agent search technique (Kennedy and Eberhart 1995), which is based on simplified social behavior of large animal groups. It has been proven that in many cases PSO algorithm provides fast and efficient solution of optimization problems (Xiao et al. 1997; Nasrollahy and Javadi 2009; Li and Chen 2005; Dutta 2010; Chen and Li 2007).

Trajectory tracking problem can be considered as a part of mobile robot navigation problem, which has been intensively researched (Laumond 1998; LaValle 2006). Typically, different proportional, derivative and integral linear (PID) controllers (Ćosić et al. 2011) are used on the cross-track error, which is the lateral deviation from a desired path.

However, when tasks require tight tracking of complex curved paths, linear feedback on the cross-track error may not provide satisfactory performance. Hence, a possible approach is the use of guidance logic with an anticipatory control element which overcomes the inherent limitation of feedback control in following curved paths. There are several terminal phase guidance laws for short-range tactical missiles that can be used to do trajectory following by using an imaginary point moving along the desired path as a pseudo target (Patrick et al. 1981). The laws are based on the interception of missile guidance techniques which are classified into six major categories: a) (LS)-control based on of the line of sight b) pure pursuit (PP) c) proportional navigation guidance (PNG) d) the optimal operating (OG) and f) deviated pursuit (DP). These methods are very robust, and on the other hand, their important advantage is the possibility of the implementation of using different types of sensors. The mentioned self-guidance techniques have low computational requirements and ease of implementation in real-time and time-optimal performance with possibility to ensure a quick interception and coordination in a dynamic environment with obstacles (Kunwar et al. 2010). The path following task in this chapter, is presented by computationally efficient and robust controllers based on missile homing and guidance laws (pure pursuit, deviated pursuits).

3 Kinematic Model of the Mobile Robot

Schematic model of mobile robot is shown on Fig. 1. Wheels on the same side are turning with the same angular velocity. World coordinate frame is denoted by $\{X, O, Y\}$, while $\{x_l, COM, y_l\}$ denotes local coordinate frame, attached at the robot. Origin of the local coordinate frame is placed at the robot center of mass (COM). State variables are position and orientation of the robot, i.e. COM position (x, y) and angle φ between x axis of the world and local coordinate frame, while ω_L and ω_D denote angular velocities of left and right side wheels of the robot, respectively, and represent control inputs. Let v_L and v_D denote linear velocities of the left and right side wheels centers, respectively, v_C COM linear velocity and $\dot{\varphi}$ robot angular velocity around COM. Relations between linear and angular velocities are given by:

$$\begin{aligned} v_L &= \omega_L r\,,\; v_D = \omega_D r \\ v_c &= \frac{1}{2}(v_L + v_D)\,,\; 2b\dot{\varphi} = v_D - v_L \end{aligned} \tag{1}$$

Combining previous equations leads to:

$$v_c = \frac{r}{2}(\omega_L + \omega_D)\,,\; \dot{\varphi} = \frac{r}{2b}(\omega_D - \omega_L) \tag{2}$$

Previous equations are related to the local coordinate system. In order to describe motion in global coordinate frame it is necessary to describe mobile robot velocity v_C in global coordinate system. If $\dot{x}$ and $\dot{y}$ denote projections of velocity v_C onto coordinate axis of global coordinate system, these projections can be written as:

$$\begin{aligned} \dot{x} &= v_c \cos\varphi \\ \dot{y} &= v_c \sin\varphi \end{aligned} \tag{3}$$

Finally, combining (2) and (3), kinematic model of mobile robot is obtained:

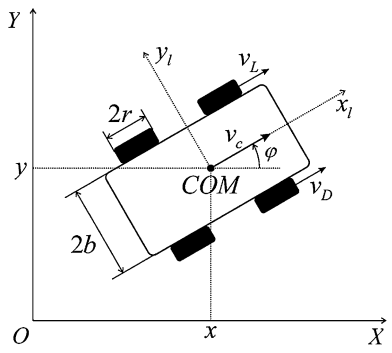

Fig. 1 Schematic model of the mobile robot

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos\varphi & \frac{r}{2}\cos\varphi \\ \frac{r}{2}\sin\varphi & \frac{r}{2}\sin\varphi \\ -\frac{r}{2b} & \frac{r}{2b} \end{bmatrix} \begin{bmatrix} \omega_L \\ \omega_D \end{bmatrix} \tag{4}$$

4 Advanced Algorithms for Motion Tracking Problem

4.1 Path Generation Using PSO Algorithm

The goal is generation of collision free path from starting to destination point, so that the path is as short as possible. This task will be solved using PSO algorithm. Generated path is given as an array of two-dimensional points, so the obtained path is not smooth. Robot has fixed maximum step size, i.e. maximum distance between current and next point V_r^{max}. It is also assumed that all obstacles are circular and there is no overlapping between obstacles, although they can touch each other, but not more than two. Sizes of all obstacles are increased for the dimension of mobile robot. Region of search is illustrated on Fig. 2. Let (x_i, y_i) denotes optimal point generated by PSO in previous iteration, which represents the center of search region in current iteration, while (x_f, y_f) denotes destination point.

Region of search is circular sector with central angle of 270°, symmetric relative to line joining points (x_i, y_i) and (x_f, y_f). On this way, algorithm always progresses in sense that every new point is closer to destination than previous one.

More complex situation arises when the region of search collides with obstacles. These situations can be avoided on two ways. The first way is to mark these points as inadequate by giving them large positive value of fitness function. This solution is simple, but in this way population loses some particles, i.e. artificially reduces the size of the population. The other way is to move points that lie inside the obstacles to the obstacle edge (see Fig. 2). On this way algorithm will move particles into the allowable part of search region, and there is no loss of population

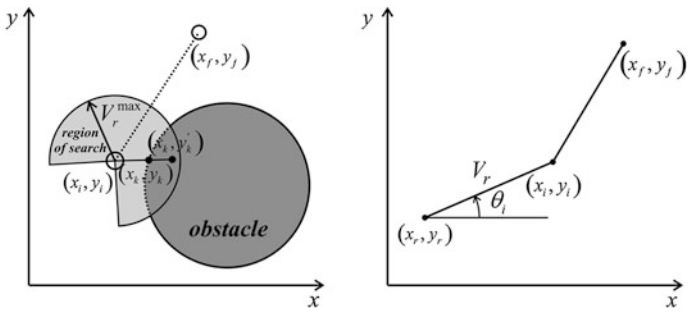

Fig. 2 Search region of the PSO algorithm

particles. Particle (x_k, y_k) which lies inside the obstacle is moved in the point (x'_k, y'_k), which is located at the intersection between obstacle edge and line joining particle (x_k, y_k) and centre of the search region (x_i, y_i), and for this new particle fitness function is evaluated. Particles in the PSO algorithm represent two-dimensional points in polar coordinates (radius with respect to the centre of the search area V_r, and angle θ between x axis and line joining particle and the centre of the search area). Next position should be obtained such that total length of the path is minimal and collision with obstacles is minimal. Path length F, from (x_r, y_r) to the destination (x_f, y_f) over (x_i, y_i) is:

$$F = V_r + \sqrt{(x_r + V_r \cos\theta_i - x_f)^2 + (y_r + V_r \sin\theta_i - y_f)^2} \qquad x_i = x_r + V_r \cos\theta_i \,,\; y_i = y_r + V_r \sin\theta_i \tag{5}$$

Fitness function is weighted sum of path length F, given in (5) and additional term $P(i)$, which represents penalties if path leads over the obstacles. Finally, fitness function fit_i is given by:

$$fit_i = w_1 F_i + w_2 P(i) \,,\; P(i) = w_2 P_1(i) + w_3 P_2(i) \qquad F_i = V_r + \sqrt{(x_r + V_r \cos\theta_i - x_f)^2 + (y_r + V_r \sin\theta_i - y_f)^2} \tag{6}$$

Penalization factor $P(i)$ consists of two factors $P_1(i)$, which represents penalization due to the existence of intersection between obstacle and path generated from current particle to the destination point, and $P_2(i)$, which represents penalization due to the existence of intersection between obstacle and path generated from particle given in the previous algorithm iteration to the current particle. Weights w_1, w_2, w_3 weight path length and path intersection with obstacles, respectively.

Let us assume that there is obstacle between current and destination point, as shown on Fig. 3. Robot can circumvent obstacle from either side, but it is rational to select the shorter path. This implies that penalization factor should be shorter of these two paths. So, penalization factor can be defined as:

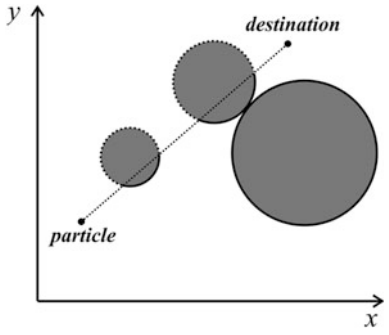

Fig. 3 Penalization factor evaluation

$$P_k(i) = \min\left\{ arc_j + \sum_{s=1}^{S} c(s) \right\}, \quad j \in \{1,2\},\ k \in \{1,2\} \tag{7}$$

where arc_j denotes arc of the obstacle intersected by path, and $c(s)$ denotes circumference of adjacent obstacle. Algorithm must proceed in both directions, and choose shorter path.

4.2 Path Interpolation Using RBFNN

Path smoothing is achieved by interpolation using Radial basis function neural networks (RBFNN). Neural networks are used for function approximation. This means that in general case they cannot be used here directly, because obtained path may not be a function. So, the idea is to parameterize x and y coordinate of the path, i.e. to associate time instant to every point of the path. Path interpolation is performed using RBFNN with one input (time) and two outputs (x and y coordinate of path point). The first step is generation of training set. This means that time instant must be associated to every point of the path. The simplest way to generate time vector is to adopt uniform velocity of motion v along the path. Now, time vector can be evaluated recursively using (8), where P_i and P_{i-1} denote two successive points of the path.

$$P_k(i) = \min\left\{ arc_j + \sum_{s=1}^{S} c(s) \right\}, \quad j \in \{1,2\},\ k \in \{1,2\} \tag{8}$$

Region centers w_i and output layer weights l_{ij} for neural network are determined in training process, but region width σ and hidden layer size must be adjusted experimentally. Choice of these two parameters is critical. It is important to note that after training it could happen that path generated by RBFNN is not collision free.

4.3 Trajectory Generation Based on Interpolated Path

Trajectory generation based on given path means introduction of time, i.e. trajectory represents time-parameterized path. Hence, the idea is to approximate curvature numerically. Let (x_i, y_i) denote current robot position and let the part of the path from point (x_i, y_i) to point (x_k, y_k) is circle arc of length L, as shown on Fig. 4. If the length L is fixed, measure of path curvature could be the central angle α of the arc. This angle could be obtained as a sum of individual angles $\alpha_1, \alpha_2, \ldots, \alpha_{k-i}$.. Angle α_1 stands for the angle between vectors $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$, while angle α_{k-i} stands for the angle between vectors $\mathbf{p}_{k-1}$ and $\mathbf{p}_k$. If the sampling of the

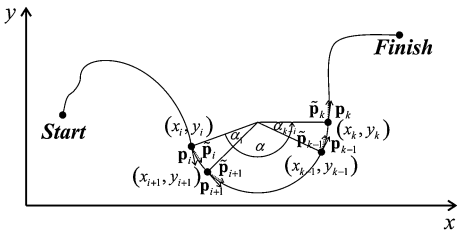

Fig. 4 Measure of curvature determination

path is frequent enough, the path can be approximated adequately with piecewise straight line, and tangent line becomes a line which joins two successive points, i.e. vector $\tilde{\mathbf{p}}_i$ becomes very close to real tangent vector $\mathbf{p}_i$. So, the individual angles can be evaluated using the following formula:

$$\begin{aligned} \alpha^{(i)} &= \sum_{j=1}^{k-i} \alpha_j \,,\ \alpha_j = \arccos \frac{\tilde{\mathbf{p}}_j \cdot \tilde{\mathbf{p}}_{j+1}}{\left\|\tilde{\mathbf{p}}_j\right\| \cdot \left\|\tilde{\mathbf{p}}_{j+1}\right\|} \,, \\ \tilde{\mathbf{p}}_j &= \left(x_{j+1} - x_j, y_{j+1} - y_j\right),\ \tilde{\mathbf{p}}_{j+1} = \left(x_{j+2} - x_{j+1}, y_{j+2} - y_{j+1}\right) \end{aligned} \tag{10}$$

When the measure of curvature α is determined in every point of the path, velocity vector v can be evaluated, which is inversely proportional to α. So, velocity in point (x_i, y_i), $v^{(i)}$, becomes:

$$v^{(i)} = \frac{k}{\alpha^{(i)}} \tag{10}$$

Parameter k is proportionality constant and it must be chosen such that robot passes the sharpest curve of the path safely. Time vector is evaluated using obtained velocity profile. Let the initial time instant is zero, and let (x_0, y_0) denotes the starting position of the robot. Based on evaluated velocity profile, time vector can be obtained:

$$\begin{aligned} v^{(i)} &= \frac{\Delta s}{\Delta t} = \frac{s_i - s_{i-1}}{t_i - t_{i-1}} = \frac{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{t_i - t_{i-1}} \\ t_i &= t_{i-1} + \frac{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{v^{(i)}} \end{aligned} \tag{11}$$

4.4 Design of Self-Guidance Controllers

The assumption is that guided object (real robot) has all the relevant information, such as their own position and the position of the target (virtual robots), so that the controller is able to generate control that leads to the goal, in order to make

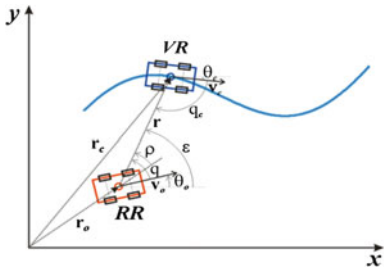

Fig. 5 Self guidance scenario

interception. Thus, the goal of control algorithm is to realize interception with virtual robot. Self-Guidance scenario is indicated in Fig. 5 with defined variables: ***VR***—virtual robot—goal; ***RR***—real robot—guided object; $\mathbf{v}_o, \mathbf{v}_c$—linear velocity of the real and virtual robot; $\mathbf{r}_o, \mathbf{r}_c$—position vector of the real and virtual robot; $\mathbf{r}$—vector of the relative position of the virtual robot in relation to the real robot; θ_o, θ_c—orientation of the velocity vectors of the real and virtual robot; q, q_c—passing angles, and orientation of the difference in relative position and velocity (for the real and virtual robot); ε—orientation of the virtual robot position in relation to the real robot; ρ—sighting angle.

4.4.1 Pure Pursuit Guidance

Pure Pursuit Guidance starts from the idea that the controller should generate such a control action that the real robot is always directed towards current position of the virtual, that velocity vector of the real robot should be directed to the current position of the center of mass of the virtual robot. In other words, the vectors $\mathbf{r}$ and $\mathbf{v}_o$ must be collinear or overtaking angle q should be 0. This means that the control objective can be expressed as:

$$\mathrm{v}_o = \begin{bmatrix} v_{ox} \\ v_{oy} \end{bmatrix} = k\mathrm{r} = k(\mathrm{r}_c - \mathrm{r}_o) = k \begin{bmatrix} x^* - x \\ y^* - y \end{bmatrix} \tag{12}$$

where k is a gain, which affects the performance monitoring, while (x, y, φ_o) is the marked position and orientation of the center of mass of the real robot, and $(x^*, y^*, \varphi_c)$is the position of the center of mass and orientation of a virtual robot. However, since the control inputs are the angular velocity of left and right wheel, it is necessary to use kinematic equations of the robot, but in the reverse direction (i.e. from the output to the input):

$$\begin{aligned} \|\mathrm{v}_o\| &= \sqrt{v_{ox}^2 + v_{oy}^2},\ \varphi_o = arctg\frac{v_{oy}}{v_{ox}} \\ \omega_L &= \frac{1}{r}\left[\|\mathrm{v}_o\| - b\dot{\varphi}_o\right],\ \omega_D = \frac{1}{r}\left[\|\mathrm{v}\|_o + b\dot{\varphi}_o\right] \end{aligned} \tag{13}$$

4.4.2 Deviated Pursuit Guidance

Deviated Pursuit Guidance is based on pursuit with keeping a constant overtaking angle. It starts from the idea that the real robot is not directed towards the current position of the virtual robot, but in front of him, or in the direction of his passing. In other words, the controller should generate control that would ensure that the velocity vector is pointing in the direction of the vector **r**, but rotated by a constant angle q. Control objective can be expressed by the equation:

$$v_o = kTr = kT(r_c - r_o)$$
$$T(q) = \begin{bmatrix} \cos q & -\sin q \\ \sin q & \cos q \end{bmatrix}, \; T = \begin{cases} T(-q), \varphi_c \in (-\pi/2, \pi/2) \\ T(0), \varphi_c \approx \pm\pi/2 \\ T(q), \text{inače} \end{cases} \tag{14}$$

where T is the transform matrix indicated that performs vector $k\mathbf{r}$ rotation for overtaking fixed angle q.

5 Experimental Setup

The considered experimental 4 wheel-based mobile robotic platform is shown in Fig. 6. It has a modular structure that consists of the following subsystems (modules): (1) four-wheel drive rover, (2) five degrees of freedom (dof) robot arm for small objects manipulation, (3) IP stereovision camera, (4) advanced sensor system including different localization and navigation sensors (visual and non-visual), (5) PC-based on-board controller and (6) Wi-Fi communication module, (7) immobile obstacles. A Hagisonic StarGazer infrared sensor unit (presented in Fig. 7) is used for global localization of the robotic system in the workspace.

6 Simulation and Experimental Case Studies

6.1 Simulation Results for PSO Based Robot Path Generation

Proposed algorithm for motion planning of mobile robot is implemented in MATLAB. The scenario with seven obstacles is adopted. In order to include robot dimensions, obstacles are enlarged with the dimension of mobile robot.

It is assumed that robot width is $2b = 30$ cm and wheel radius is $r = 6$ cm. Maximum angular velocities of the wheels are $\omega_L^{max} = \omega_D^{max} = 15$ rad/s. It is assumed that the robot position and orientation measurements are corrupted with white Gaussian noise, which standard deviations are 1 cm and 1°, respectively.

Fig. 6 4 Wheel-based mobile robot

Fig. 7 Hagisonic StarGazer system

Odometer range is $R_g = 1.5$ m. Starting point is $(2, 4.5)$ m, while the destination point is $(4.5, 0.5)$m. PSO algorithm searches space with the 30 particles in the swarm. Particles velocities are bounded on interval $[-1, +1]$. Search area radius is $V_r^{max} = 0.25$ m. Values of weights in fitness function are $w_1 = 1, w_2 = w_3 = 5$. Parameters c_1, c_2, w are iteration-variable, chosen according to following law:

$$c_1(k) = 0.4e^{-\frac{k-2}{8}} + 0.8\,,\; c_2(k) = 0.01k + 0.48\,,\; w(k) = -0.05k + 0.89 \qquad (15)$$

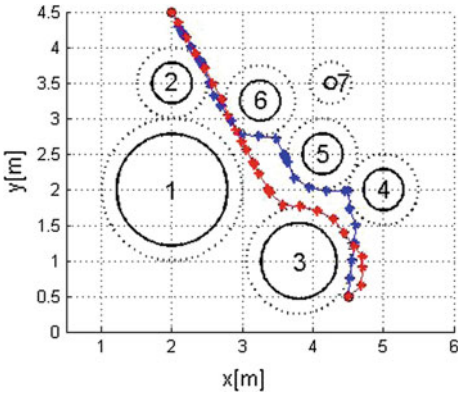

Fig. 8 Paths designed by PSO approach

Because of the stochastic nature of the algorithm, paths generated from starting to destination point (blue line on Fig. 8 and vice versa (red line on Fig. 8) will not be the same. Length of the blue path is 5.20 m, while the length of the red path is 5.13 m. So, in this case, red path is better one. On Fig. 8, points generated by algorithm are represented with stars, while real obstacles are represented with solid black line. Obstacles enlarged by robot dimension are represented with dashed black line.

The next step is path interpolation using RBFNN. Training set consists of input–output training pairs. Each training pair consists of point on the path and corresponding time instant, which is obtained assuming constant velocity of robot motion. It is assumed that this velocity is $v = 0.2$ m/s. Hidden layer size should be chosen such that approximation of the path is adequate and collision free. Approximation performance heavily depends on hidden layer size and region width σ. These parameters have been determined experimentally. Hidden layer has 24 neurons, while $\sigma = 2.375$. Neural network response is shown on Fig. 9.

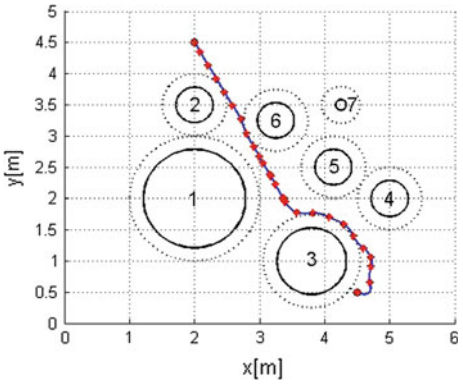

Fig. 9 Interpolated path by RBFNN

6.2 Experimental Results for Self-Guidance Controllers

Experimental results have been carried out on the four wheel differentially driven DF Robot. The proposed algorithm is written in the programming language C, which operates under the Ubuntu operating system on the robot. Hagisonic StarGazer infrared camera has been used as a localization system.

The allowed space of search for PSO algorithm is restricted to the space visible to the StarGazer camera, i.e. $-1.2\text{ m} \leq x \leq 1.2\text{ m}, -0.6\text{ m} \leq y \leq 0.6\text{ m}$. Starting point is $P_{START} = (-0.95, -0.2)\text{m}$, while the destination point is $P_{FINISH} = (0.85, 0.4)\text{m}$. Space of search is circular sector whose radius is $VR_{max} = 2\,\text{cm}$ and angle regarding its line of symmetry is 150°. Size of the swarm is 30 particles, while the algorithm terminates after 100 iterations. Self-confidence and swarm-confidence parameters, c_1 and c_2 changes as algorithm iterates (c_1 decreases, while c_2 increases), as well as value of inertia. These parameters change their values according to (15).

Values of parameters in the criteria are $w_1 = 1, w_2 = 10$. For the purpose of path interpolation RBFNN with 18 hidden neurons is chosen. Value of the region width is $\sigma = 0.9$. Proportionality constant used for generating the velocity vector is $k = 0.125$, while the length of the path on which the path curvature is evaluated is 0.5 m. Minimum and maximum velocity of the virtual robot along the path is

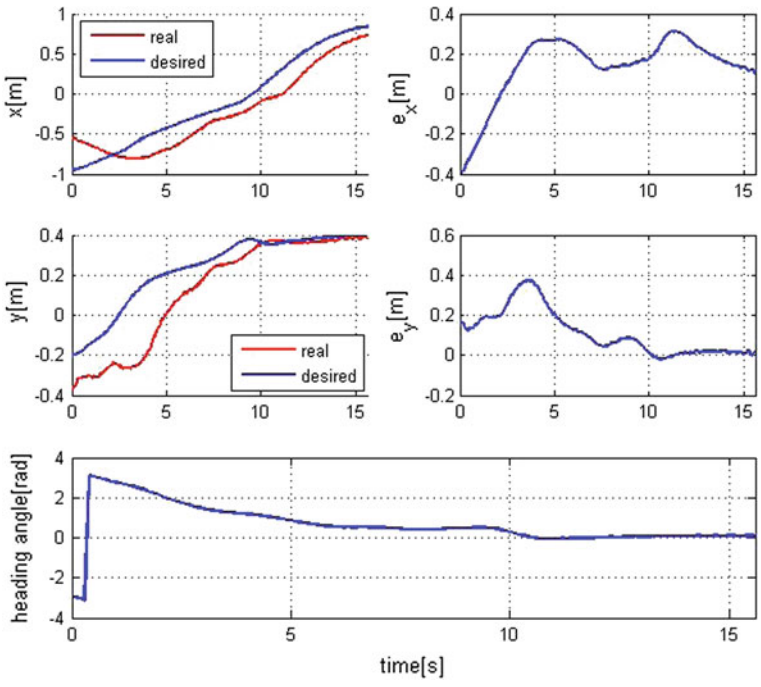

Fig. 10 State variables and error components—PP controller

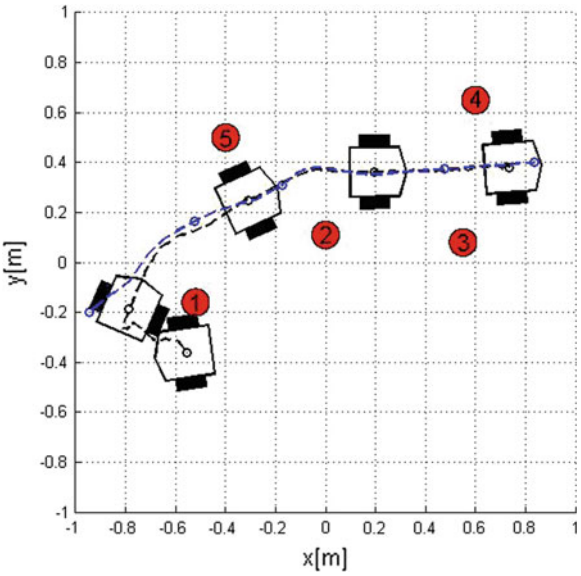

Fig. 11 2 D view—PP controller

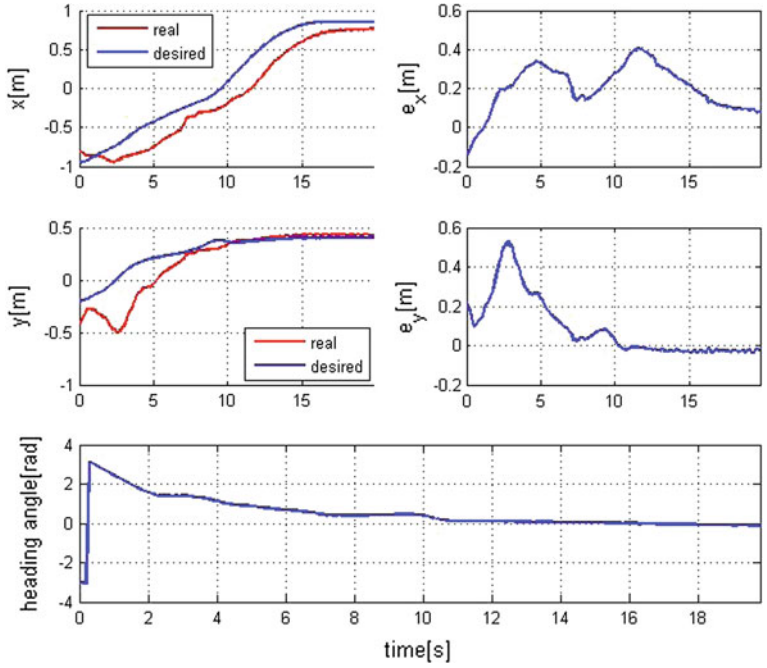

Fig. 12 State variables and error components—DP controller

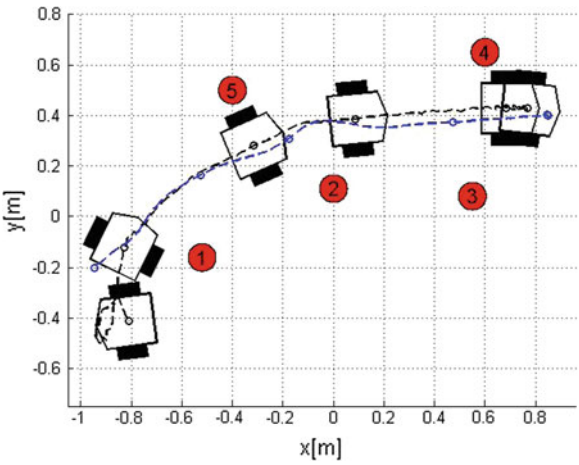

Fig. 13 2 D view—DP controller

$V_{min} = 0.05$ m/s and $V_{max} = 0.2$ m/s, respectively. Controllers based on pure pursuit (PP) and deviated pursuit (DP) have been proposed for the trajectory tracking. Parameters of the PP controller are: $K = 2, a_v = 8, a_\varphi = 2$, while the parameters of the DP controller are: $K = 1.4, q = 0.13, a_v = 10, a_\varphi = 2$.

Trajectory tracking performance, when the PP controller is used, is shown in the Fig. 10. It shows plots of the x and y coordinates, as well as orientation (heading angle of the robot) and x and y components of the error. The tracking performance in the two dimensional world is shown in the Fig. 11. Blue dotted line designates desired trajectory (virtual vehicle), while the black solid line designates real robot. Red filled circles represent the obstacles.

Trajectory tracking performance, when the DP controller is used, is shown in the Fig. 12, while the path following in the two dimensional plane is shown in the Fig. 13. Comparing the results shown on Figs. 11 and 12, it can be seen that two controllers provide similar tracking performance. If the mean square error is adopted as criteria for comparing these two controllers, the following results are obtained: $MSE(PP) = 0.25$ m and $MSE(DP) = 0.26$ m, i.e. the obtained results are very similar. From the other side, path following performance is better in the PP case, in the sense that real robot better follows the desired path than in the DP case.

7 Conclusions

The solution for motion planning in structured environments is presented in this chapter. This problem can be divided into generation of collision free path using PSO and RBFNN and trajectory tracking using advanced self-guidance laws. Although it is assumed that obstacles are circular, proposed method, with slight modifications, can be applied on obstacles of arbitrary shape. This approach can be

applied in dynamic environments in which exist moving obstacles, due to action of obstacle avoidance fuzzy controller. Some future work should include implementation of online trajectory planning algorithm as well as simulations utilizing more realistic dynamic model of the wheeled mobile robot.

Acknowledgments This project is supported by Ministry of Science and Education of Republic Serbia under the grants TR-35003 and III-44008.

References

Canny J (1988) The complexity of robot motion planning. MIT Press, Cambridge

Chen X, Li Y (2007) Neural network predictive control for mobile robot using PSO with controllable random exploration velocity. Int J Intell Control Syst 12(3):217–229

Ćosić A, Šušić M, Ribić A, Katić D (2011) An Approach for Mobile Robot Trajectory Generation and Tracking. In: Proceedings of the 55. ETRAN conference, Banja Vrućíca, Bosnia and Herzegovina

Dutta S (2010) Obstacle avoidance of mobile robot using PSO based neuro fuzzy technique. Int J Comput Sci Eng 2(2):301–304

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, pp 1942–1948

Kunwar F, Sheridan PK, Benhabib B (2010) Predictive guidance-based navigation for mobile robots: a novel strategy for target interception. J Intell Robot Syst 59:367–398

Laumond J (1998) Robot motion planning and control. Springer, London

LaValle S (2006) Planning algorithms. Cambridge University, Cambridge

Li Y, Chen X (2005) Mobile robot navigation using particle swarm optimization and adaptive neural networks. In: Proceedings of ICNC, pp 628–631

Masehian E, Sedighizadeh D (2007) Classic and heuristic approaches in robot motion planning—a chronological review. World Acad Sci Eng Technol 29:101–106

Nasrollahy AZ, Javadi HHS (2009) Using particle swarm optimization for robot path planning in dynamic environments with moving obstacles and target. Third European Symposium on computer modeling and simulation, Athens, Greece, 60–65, 2009

Patrick HL, Seltzer SM, Warren ME (1981) Guidance laws for short-range tactical missiles. J Guidance Control Dyn 4(2):98–108

Raja P, Pugazhenthi S (2012) Optimal path planning of mobile robots: a review. Int J Phys Sci 7(9):1314–1320

Xiao J, Michalewicz Z, Zhang L, Trojanowski K (1997) Adaptive evolutionary planner/navigator for mobile robots". IEEE Trans Evol Comput 1:18–28