A 3-level autonomous mobile robot navigation system designed by using reasoning/search approaches

Article *in* Robotics and Autonomous Systems · December 2006

DOI: 10.1016/j.robot.2006.05.006 · Source: DBLP

3 authors:

Jasmin Velagic
University of Sarajevo
101 PUBLICATIONS 599 CITATIONS
SEE PROFILE

Bakir Lacevic
University of Sarajevo
52 PUBLICATIONS 484 CITATIONS
SEE PROFILE

Branislava Peruničić-Drazenovic
University of Sarajevo
88 PUBLICATIONS 1,032 CITATIONS
SEE PROFILE

Available online at www.sciencedirect.com
ScienceDirect

Robotics and Autonomous Systems 54 (2006) 989–1004

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

A 3-level autonomous mobile robot navigation system designed by using reasoning/search approaches

Jasmin Velagic*, Bakir Lacevic, Branislava Perunicic

University of Sarajevo, Faculty of Electrical Engineering, Zmaja od Bosne bb, 71000 Sarajevo, Bosnia and Herzegovina

Received 3 November 2004; received in revised form 24 April 2006; accepted 15 May 2006
Available online 14 July 2006

Abstract

This paper describes how soft computing methodologies such as fuzzy logic, genetic algorithms and the Dempster–Shafer theory of evidence can be applied in a mobile robot navigation system. The navigation system that is considered has three navigation subsystems. The lower-level subsystem deals with the control of linear and angular volocities using a multivariable PI controller described with a full matrix. The position control of the mobile robot is at a medium level and is nonlinear. The nonlinear control design is implemented by a backstepping algorithm whose parameters are adjusted by a genetic algorithm. We propose a new extension of the controller mentioned, in order to rapidly decrease the control torques needed to achieve the desired position and orientation of the mobile robot. The high-level subsystem uses fuzzy logic and the Dempster–Shafer evidence theory to design a fusion of sensor data, map building, and path planning tasks. The fuzzy/evidence navigation based on the building of a local map, represented as an occupancy grid, with the time update is proven to be suitable for real-time applications. The path planning algorithm is based on a modified potential field method. In this algorithm, the fuzzy rules for selecting the relevant obstacles for robot motion are introduced. Also, suitable steps are taken to pull the robot out of the local minima. Particular attention is paid to detection of the robot's trapped state and its avoidance. One of the main issues in this paper is to reduce the complexity of planning algorithms and minimize the cost of the search. The performance of the proposed system is investigated using a dynamic model of a mobile robot. Simulation results show a good quality of position tracking capabilities and obstacle avoidance behavior of the mobile robot.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Dynamic model of mobile robot; Backstepping algorithm; Hybrid position controller; Map building; Genetic algorithm; Fuzzy logic; Path planning; Pruning of relevant obstacles; Local minima problem

1. Introduction

The basic feature of an autonomous mobile robot is its capability to operate independently in unknown or partially known environments. The autonomy implies that the robot is capable of reacting to static obstacles and unpredictable dynamic events that may impede the successful execution of a task [10]. To achieve this level of robustness, methods need to be developed to provide solutions to localization, map building, planning and control. The development of such techniques for autonomous robot navigation is one of the major trends in current robotics research [23].

The robot has to find a collision-free trajectory between the starting configuration and the goal configuration in a static or dynamic environment containing some obstacles. To this end, the robot needs the capability to build a map of the environment, which is essentially a repetitive process of moving to a new position, sensing the environment, updating the map, and planning subsequent motion.

Most of the difficulties in this process originate in the nature of the real world [17]: unstructured environments and inherent large uncertainties. First, any prior knowledge about the environment is, in general, incomplete, uncertain, and approximate. For example, maps typically omit some details and temporary features; also, spatial relations between objects may have changed since the map was built. Second, perceptually acquired information is usually unreliable. Third, a real-world environment typically has complex and unpredictable dynamics: objects can move, other agents can modify the environment, and apparently stable features may change with time. Finally, the effects of control actions are not

* Corresponding author. Tel.: +387 33 25 07 65; fax: +387 33 25 07 25.
E-mail address: jasmin.velagic@etf.unsa.ba (J. Velagic).

0921-8890/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2006.05.006

completely reliable, e.g. the wheels of a mobile robot may slip, resulting in accumulated odometric errors.

Robot navigation can be defined as the combination of three basic activities [4,16]:

- Map building. This is the process of constructing a map from sensor readings taken at different robot locations. The correct treatment of sensor data and the reliable localization of the robot are fundamental in the map-building process.
- Localization. This is the process of getting the actual robot's location from sensor readings and the most recent map. An accurate map and reliable sensors are crucial to achieving good localization.
- Path planning. This is the process of generating a feasible and safe trajectory from the current robot location to a goal based on the current map. In this case, it is also very important to have an accurate map and a reliable localization.

Path planning is one of the key issues in mobile robot navigation. Path planning is traditionally divided into two categories: global path planning and local path planning. In global path planning, prior knowledge of the workspace is available [21]. Local path planning methods use ultrasonic sensors, laser range finders, and on-board vision systems to perceive the environment to perform on-line planning [1,8]. In our paper, the workspace for the navigation of the mobile robot is assumed to be unknown and it has stationary obstacles only.

In local path planning methods, particular attention is paid to local minima problem. This problem occurs when a robot navigates towards a desired target with no prior knowledge of the environment and gets trapped in a loop [7,14]. This happens usually if the environment consists of concave obstacles, mazes, and such objects. To get out of the loop, the robot must comprehend its repeated traversal through the same environment, which involves memorizing the environment that has already been seen [12].

The main contribution of this paper is the design of a robust autonomous mobile robot control system suitable for on-line applications with real-time requirements by using soft computing methodologies. This system provides the mobile robot that may navigate in an a priori unknown indoor environment using sonar sensor information. To achieve these requirements, the proposed system is hierarchically organized into three distinct separated subsystems with arbitrary responsibility. At each level of this system, one or more soft computing methodologies are adopted to solve its specific problems.

A low-level velocity controller is developed using the standard PI multivariable control law. The medium-level position control law has to be nonlinear in order to ensure stability of the error, that is, its convergence to zero [6,18]. Some of the control parameters are continuous time functions, and usually the backstepping method [6,15,22] was used for their adjustment. In order to achieve the optimal parameter values, we used a genetic algorithm. Both controllers are based on a dynamic model of a differential drive mobile robot having angular velocities as main variables.

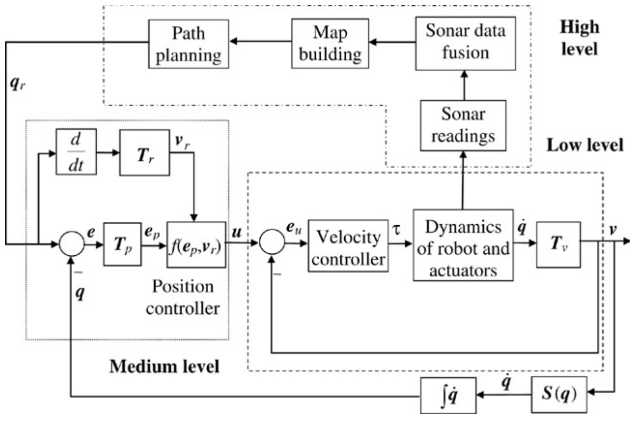

Fig. 1. Mobile robot navigation system for real-time requirements.

A high-level subsystem contains map-building and path-planning algorithms. In this paper, the application of an occupancy-based map [17] using Dempster–Shafer evidence theory based on sonar measurements is demonstrated. The integration of sonar data is obtained by a low-level fusion [9]. The building of occupancy maps is well suited to path planning and obstacle avoidance [10]. In this paper, we propose a new path-planning approach based on the repulsive and attractive forces, which sets the fuzzy rules for determining which obstacles should have an influence on the mobile robot motion. Fuzzy logic offers the possibility to mimic expert human knowledge [2,3,11,26]. This approach provides both obstacle-avoidance and target-following behaviors and uses only the local information for decision making for the next action. Also, we propose a new algorithm for the identification and solution of the local minima situation during the robot's traversal using the set of fuzzy rules.

2. Navigation system

The proposed three-level control navigation system is shown in Fig. 1.

The low-level velocity control system is composed of a multivariable PI controller and a dynamic model of a mobile robot and actuators. At the medium level, the position control system generates a nonlinear control law whose parameters are obtained using a genetic algorithm. The high-level system performs map-building and path-planning tasks. This system is in charge of sensor interpretation, sensor data fusion, map building, and path planning. All the modules are designed using fuzzy logic and the Dempster–Shafer theory of evidence.

In the following sections, the design of the navigation system blocks from Fig. 1 is described.

3. Dynamics of mobile robot

In this section, a dynamic model of a nonholonomic mobile robot with viscous friction will be derived first. A typical representation of a nonholonomic mobile robot is shown in Fig. 2.

The robot has two driving wheels mounted on the same axis and a free front wheel. The two driving wheels are

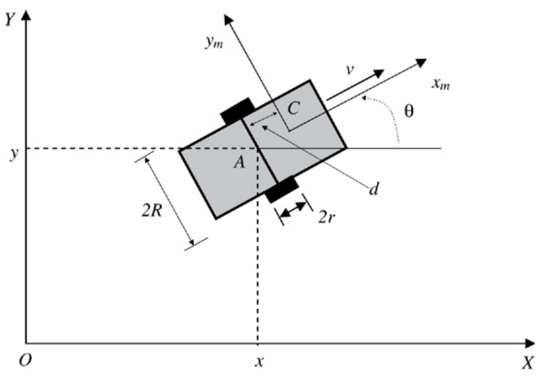

Fig. 2. The representation of a nonholonomic mobile robot.

independently driven by two actuators to achieve both transition and orientation. The position of the mobile robot in the global frame $\{X, O, Y\}$ can be defined by the position of the mass center of the mobile robot system, denoted by C, or alternatively by the position A, which is the center of the mobile robot gear, and the angle between the robot's local frame $\{x_m, C, y_m\}$ and the global frame. The kinetic energy of the whole structure is given by the following equation:

$$T = T_l + T_r + T_{kr}, \tag{1}$$

where T_l is the kinetic energy that is the consequence of pure translation of the entire vehicle, T_r is the kinetic energy of rotation of the vehicle in the XOY plane, and T_{kr} is the kinetic energy of rotation of the wheels and rotors of the DC motors. The values of energy terms introduced can be expressed by Eqs. (2)–(4):

$$T_l = \frac{1}{2}Mv_c^2 = \frac{1}{2}M(\dot{x}_c^2 + \dot{y}_c^2), \tag{2}$$

$$T_r = \frac{1}{2}I_A\dot{\theta}^2, \tag{3}$$

$$T_{kr} = \frac{1}{2}I_0\dot{\theta}_R^2 + \frac{1}{2}I_0\dot{\theta}_L^2, \tag{4}$$

where M is the mass of the entire vehicle, v_c is the linear velocity of the vehicle's center of mass C, I_A is the moment of inertia of the entire vehicle considering point A, θ is the angle that represents the orientation of the vehicle (Fig. 2), I_0 is the moment of inertia of the rotor/wheel complex, and $\mathrm{d}\theta_R/\mathrm{d}t$ and $\mathrm{d}\theta_L/\mathrm{d}t$ are the angular velocities of the right- and left-hand wheels, respectively.

Further, the components of the velocity of the point A can be expressed in terms of $\mathrm{d}\theta_R/\mathrm{d}t$ and $\mathrm{d}\theta_L/\mathrm{d}t$:

$$\dot{x}_A = \frac{r}{2}(\dot{\theta}_R + \dot{\theta}_L)\cos\theta, \tag{5}$$

$$\dot{y}_A = \frac{r}{2}(\dot{\theta}_R + \dot{\theta}_L)\sin\theta, \tag{6}$$

$$\dot{\theta} = \frac{r(\dot{\theta}_R - \dot{\theta}_L)}{2R}. \tag{7}$$

Since $\dot{x}_C = \dot{x}_A - d\dot{\theta}\sin\theta$ and $\dot{y}_C = \dot{y}_A + d\dot{\theta}\cos\theta$, where d is the distance between points A and C, it is obvious that the equations below follow:

$$\dot{x}_C = \frac{r}{2}(\dot{\theta}_R + \dot{\theta}_L)\cos\theta - d\dot{\theta}\sin\theta, \tag{8}$$

$$\dot{y}_C = \frac{r}{2}(\dot{\theta}_R + \dot{\theta}_L)\sin\theta + d\dot{\theta}\cos\theta. \tag{9}$$

By substituting terms in Eq. (1) with expressions in Eqs. (2)–(9), the total kinetic energy of the vehicle can be calculated in terms of $\mathrm{d}\theta_R/\mathrm{d}t$ and $\mathrm{d}\theta_L/\mathrm{d}t$:

$$\begin{aligned} T(\dot{\theta}_R, \dot{\theta}_R) = {} & \left(\frac{Mr^2}{8} + \frac{(I_A + Md^2)r^2}{8R^2} + \frac{I_0}{2}\right)\dot{\theta}_R^2 \\ & + \left(\frac{Mr^2}{8} + \frac{(I_A + Md^2)r^2}{8R^2} + \frac{I_0}{2}\right)\dot{\theta}_L^2 \\ & + \left(\frac{Mr^2}{4} - \frac{(I_A + Md^2)r^2}{4R^2}\right)\dot{\theta}_R\dot{\theta}_L. \end{aligned} \tag{10}$$

Now, the Lagrange equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial\dot{\theta}_R}\right) - \frac{\partial L}{\partial\theta_R} = \tau_R - K\dot{\theta}_R, \tag{11}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial\dot{\theta}_L}\right) - \frac{\partial L}{\partial\theta_L} = \tau_L - K\dot{\theta}_L, \tag{12}$$

are applied.

Here, τ_R and τ_L are right and left actuation torques and $K\mathrm{d}\theta_R/\mathrm{d}t$ and $K\mathrm{d}\theta_L/\mathrm{d}t$ are the viscous friction torques of right and left wheel-motor systems, respectively.

Finally, the dynamic equations of motion can be expressed as:

$$A\ddot{\theta}_R + B\ddot{\theta}_L = \tau_R - K\dot{\theta}_R, \tag{13}$$

$$B\ddot{\theta}_R + A\ddot{\theta}_L = \tau_L - K\dot{\theta}_L, \tag{14}$$

where

$$\begin{aligned} A &= \left(\frac{Mr^2}{4} + \frac{(I_A + Md^2)r^2}{4R^2} + I_0\right) \\ B &= \left(\frac{Mr^2}{4} - \frac{(I_A + Md^2)r^2}{4R^2}\right). \end{aligned} \tag{15}$$

In this paper, we used a mobile robot with the following parameters: $M = 10$ kg, $I_A = 1$ kg m^2, $r = 0.035$ m, $R = 0.175$ m, $d = 0.05$ m, $m_0 = 0.2$ kg and $I_0 = 0.001$ kg m^2.

In the following section, a design for both velocity and position controls will be established.

4. Position and velocity control designs

The function of this controller is to implement a mapping between the known information (e.g. reference position, velocity and sensor information) and the actuator commands designed to achieve the robot task. For a mobile robot, the controller design problem can be described as follows: given the reference position $\boldsymbol{q}_r(t)$ and velocity $\dot{\boldsymbol{q}}_r(t)$, design a control law for the actuator torques so that the mobile robot velocity may track a given reference control path with a given smooth

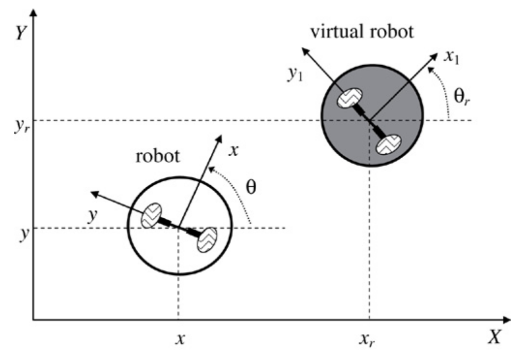

Fig. 3. The concept of tracking of a virtual reference robot.

velocity. Let the velocity and position of the reference robot (Fig. 3) be given as:

$$\boldsymbol{q}_r = [x_r\, y_r \theta_r]^{\mathrm{T}}, \tag{16}$$

where $\dot{x}_r = v_r \cos\theta_r$, $\dot{y}_r = v_r \sin\theta_r$, $\dot{\theta}_r = \omega_r$, and $v_r > 0$ is the reference linear velocity and ω_r is the reference angular velocity.

4.1. Position control

The trajectory tracking problem for a mobile robot is based on a virtual reference robot [5] that has to be tracked (Fig. 3). The tracking position error between the reference robot and the actual robot can be expressed in the robot frame as:

$$\boldsymbol{e}_p = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \boldsymbol{T}_p \boldsymbol{e}_q = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}, \tag{17}$$

where $\boldsymbol{e}_q = \begin{bmatrix} e_x & e_y & e_\theta \end{bmatrix}^{\mathrm{T}}$.

The dynamics of the position error derived in (17) is postulated as:

$$\begin{aligned} \dot{e}_1 &= \omega e_2 + u_1, \\ \dot{e}_2 &= -\omega e_1 + v_r \sin e_3, \\ \dot{e}_3 &= u_2, \end{aligned} \tag{18}$$

where inputs u_1 and u_2 are new control inputs.

In this paper, we propose the following control inputs in the velocity control loop:

$$\begin{aligned} u_1 &= v_r \cos e_3 + \frac{k_1 e_1}{\sqrt{k_4 + e_1^2 + e_2^2}}, \\ u_2 &= \omega_r + \frac{k_2 v_r e_2}{\sqrt{k_5 + e_1^2 + e_2^2}} + \frac{k_3 v_r \sin e_3}{\sqrt{k_6 + e_3^2}}, \end{aligned} \tag{19}$$

where k_1, k_2, k_3, k_4, k_5 and k_6 are positive parameters.

Eq. (19) is a modified backstepping control law given first in [6]. The modification consists of the introduction of denominators. In [6], Lyapunov's stability theory was used to prove that the control law that is considered provides a uniformly bounded norm of error $\|\boldsymbol{e}_p(t)\|$. The issue of rigorous proof of stability for the control law introduced (19) remains open. However, usage of this law is justified by simulation results.

The key problem in such a control design is to obtain control coefficients k_1 to k_6. To solve this problem, a genetic algorithm is used to find the optimal values of those coefficients. To apply this method, a low-level velocity controller has to be designed first.

4.2. Velocity control

The dynamics of the velocity controller is given by the following equations in the Laplace domain:

$$\boldsymbol{\tau}(s) = \begin{bmatrix} \tau_R(s) \\ \tau_L(s) \end{bmatrix} = \frac{1}{r} \begin{bmatrix} g_1(s) & g_2(s) \\ g_1(s) & -g_2(s) \end{bmatrix} \begin{bmatrix} e_v(s) \\ e_\omega(s) \end{bmatrix}, \tag{20}$$

where $e_v(s)$ is the linear velocity error and $e_\omega(s)$ is the angular velocity error. This structure differs from previously used diagonal structures. Transfer functions $g_j(s)$ are chosen to represent PI controllers:

$$\begin{aligned} g_1(s) &= K_1 \left(1 + \frac{1}{T_{i1} s}\right) \cdot R, \\ g_2(s) &= K_2 \left(1 + \frac{1}{T_{i2} s}\right) \cdot R. \end{aligned} \tag{21}$$

The particular choice of the adopted multivariable PI controller described by Eqs. (20) and (21) is justified with the following theorem.

Theorem. *Torque control (20) ensures tracking servo inputs u_1 and u_2 with zero steady-state errors.*

Proof. When we substitute $\dot{\theta}_R$ with ω_R, $\dot{\theta}_L$ with ω_L, and consider (20), we can write another form of (13) and (14):

$$\begin{bmatrix} As + K & Bs \\ Bs & As + K \end{bmatrix} \begin{bmatrix} \omega_R(s) \\ \omega_L(s) \end{bmatrix} = \frac{1}{r} \begin{bmatrix} g_1(s) & g_2(s) \\ g_1(s) & -g_2(s) \end{bmatrix} \begin{bmatrix} u_1(s) - v(s) \\ u_2(s) - \omega(s) \end{bmatrix}, \tag{22}$$

where ω_R and ω_L can be expressed in terms of ω and v as:

$$\omega_R = \frac{v + R\omega}{r}, \quad \omega_L = \frac{v - R\omega}{r}$$

and then (22) can be transformed to:

$$\begin{bmatrix} As + K & Bs \\ Bs & As + K \end{bmatrix} \begin{bmatrix} v(s) + R\omega(s) \\ v(s) - R\omega(s) \end{bmatrix} = \begin{bmatrix} g_1(s) & g_2(s) \\ g_1(s) & -g_2(s) \end{bmatrix} \begin{bmatrix} u_1(s) - v(s) \\ u_2(s) - \omega(s) \end{bmatrix}$$

and further to:

$$\begin{bmatrix} \alpha_1(s) & \alpha_2(s) \\ \alpha_1(s) & -\alpha_2(s) \end{bmatrix} \begin{bmatrix} v(s) \\ \omega(s) \end{bmatrix} = \begin{bmatrix} g_1(s) & g_2(s) \\ g_1(s) & -g_2(s) \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}, \tag{23}$$

where

$$\begin{aligned} \alpha_1(s) &= \frac{(A + B)s^2 + (K + K_1 T_{i1})s + K_1}{s} \\ \alpha_2(s) &= \frac{R(A - B)s^2 + (RK + K_2 T_{i2})s + K_2}{s}. \end{aligned} \tag{24}$$

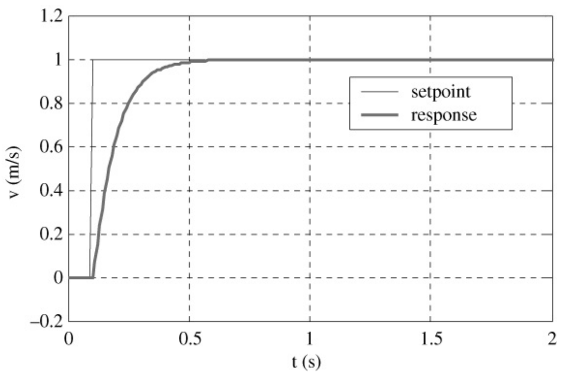

Fig. 4. Linear velocity step response.

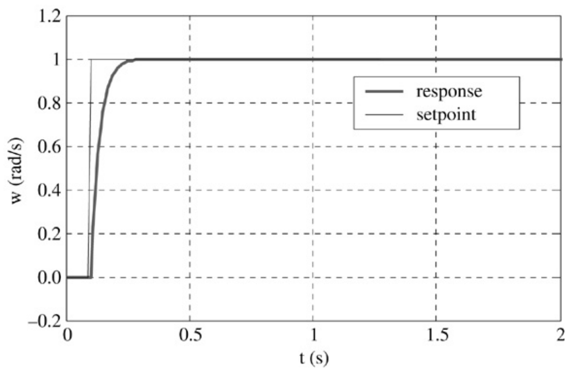

Fig. 5. Angular velocity step response.

The following equations could be easily derived from (23):

$$
\begin{aligned}
v(s) &= \frac{g_1}{\alpha_1} u_1(s) = G_1 u_1(s) \\
&= \frac{K_1 T_{i1} s + K_1}{(A+B)s^2 + (K + K_1 T_{i1})s + K_1} u_1(s) \\
\omega(s) &= \frac{g_2}{\alpha_2} u_2(s) = G_2 u_2(s) \\
&= \frac{K_2 T_{i2} s + K_2}{R(A-B)s^2 + (RK + K_2 T_{i2})s + K_2} u_2(s).
\end{aligned}
\tag{25}
$$

It is obvious that transfer functions G_1 and G_2 are static with gains equal to “1”, which completes the proof.

The velocity control loop structure is shown in Fig. 1, as an inner loop. From the simulation results obtained (Figs. 4 and 5), it can be seen that the proposed PI controller successfully tracks the given linear and angular velocity profiles. The controller parameters used for this simulation are $K_1 = 129.7749$, $K_2 = 41.0233$, $T_{i1} = 11.4018$, and $T_{i2} = 24.1873$, which are tuned using standard GA.

4.3. Evolution of the coefficients

In this paper, a simple genetic algorithm is used for parameter evolution of coefficients k_1 to k_6, which encodes into a binary chromosome (Fig. 6).

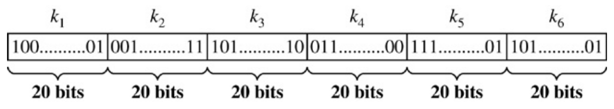

Fig. 6. Chromosome structure.

If $e_x(t)$, $e_y(t)$ and $e_\theta(t)$ $(0 < t < t_s)$ are error functions, the objective function is calculated as in (26) [13]. Parameters a_x, a_y and a_θ are some positive real numbers. N is the number of error samples. The mapping between the objective and the fitness function has not been performed, so it is obvious that the better individual has the smaller value of the objective function (fitness). In order to evaluate the fitness of the individual, it is necessary to run the simulation:

$$
\begin{aligned}
F = {} & a_x \sum_{i=0}^{N-1} \ln\left(1 + \left| e_x\left(i \frac{t_s}{N}\right)\right|\right) \\
& + a_y \sum_{i=0}^{N-1} \ln\left(1 + \left| e_y\left(i \frac{t_s}{N}\right)\right|\right) \\
& + a_\theta \sum_{i=0}^{N-1} \ln\left(1 + \left| e_\theta\left(i \frac{t_s}{N}\right)\right|\right).
\end{aligned}
\tag{26}
$$

Values of the objective function parameters are: $a_x = a_\theta = 1$, $a_y = 2$, $N = 1000$ and $t_s = 10$ s.

In (26), we have chosen the following criterion:

$$J_1 = \sum_{i=1}^{N} \ln(1 + |e_i|), \tag{27}$$

rather than the criteria that are normally used, given by (28) and (29):

$$J_2 = \sum_{i=1}^{N} |e_i| \tag{28}$$

$$J_3 = \sum_{i=1}^{N} e_i^2. \tag{29}$$

The criterion J_1 penalizes more smaller errors (expected in the stationary state) (Fig. 7) and consequently it ensures better position tracking.

Error $e_\theta(t)$ is calculated as follows:

$$
e_\theta = \begin{cases}
\theta_r - \theta, \\
\quad \text{if} \quad |\theta_r - \theta| < \min(|\theta_r - \theta - 2\pi|, |\theta_r - \theta + 2\pi|) \\
\theta_r - \theta - 2\pi, \\
\quad \text{if} \quad |\theta_r - \theta - 2\pi| < \min(|\theta_r - \theta|, |\theta_r - \theta + 2\pi|) \\
\theta_r - \theta + 2\pi, \\
\quad \text{if} \quad |\theta_r - \theta + 2\pi| < \min(|\theta_r - \theta|, |\theta_r - \theta - 2\pi|).
\end{cases}
\tag{30}
$$

This type of error is appropriate, since it prevents unnecessary full-circle rotation. The coefficient evolution is

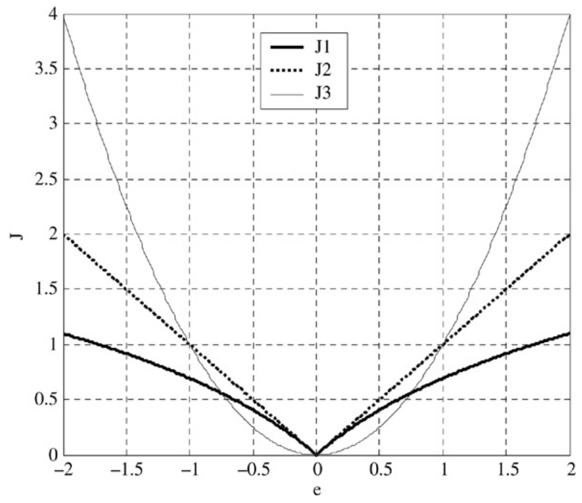

Fig. 7. Graphical illustration of different criteria.

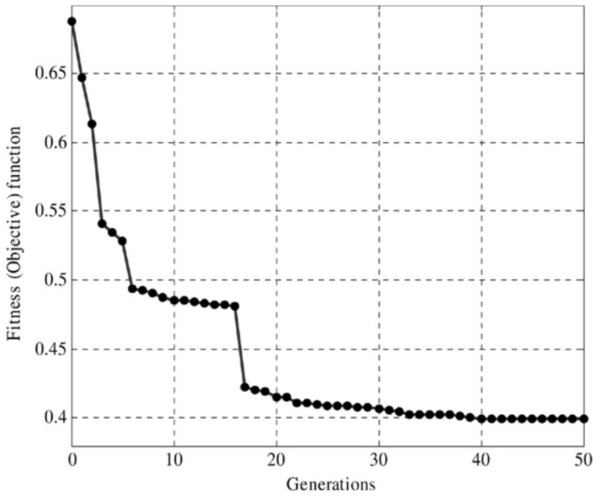

Fig. 8. Change of the best fitness in the population through generations.

performed using a lemniscate as a suitable complex trajectory:

$$x_r(t) = \frac{a\sin(\alpha t)}{1+\sin^2(\alpha t)}$$
$$y_r(t) = \frac{a\sin(\alpha t)\cos(\alpha t)}{1+\sin^2(\alpha t)} \tag{31}$$
$$\theta_r(t) = \arctan\left(\frac{\dot{y}_r(\alpha t)}{\dot{x}_r(\alpha t)}\right),$$

where $a = \alpha = 1$. The evolution process (population size is 40) is depicted in Fig. 8.

This evolution yielded the following coefficient values:

$K_1 = 6.2457$, $K_2 = 221.2306$, $K_3 = 2.3433$, $K_4 = 13.5117$, $K_5 = 3.1933$, $K_6 = 8.3361$.

4.4. Design of hybrid position controller

It has been noticed that, during the preliminary simulations, at the beginning of tracking the control torques increase rapidly if the initial position of the reference robot does not belong

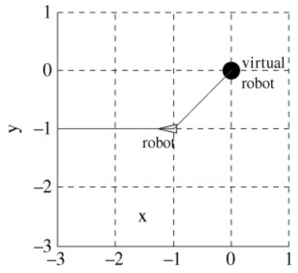

Fig. 9. Tracking robot does not “see” virtual robot.

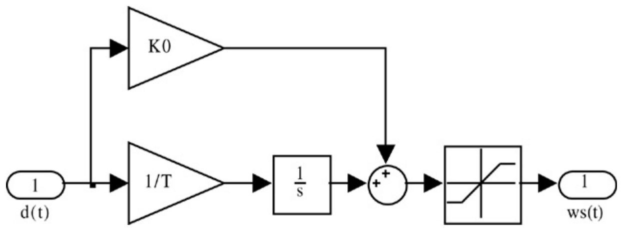

Fig. 10. Producing $\omega_s(t)$.

to the straight line, determined with the robot and its initial orientation (Fig. 9).

For that purpose, the following control law, which provides velocity servo inputs, is proposed:

$$\begin{aligned}\bar{u}_1(t) &= \alpha(t)u_1(t)\\ \bar{u}_2(t) &= \alpha(t)u_2(t) + (1-\alpha(t))\omega_s(t).\end{aligned} \tag{32}$$

Function $\omega_s(t)$ is generated as the output of the following system (Fig. 10).

The input $d(t)$ is given by:

$$\mathrm{d}(t) = \operatorname{sgn}(\mathit{atan2}(e_y(t), e_x(t)) - \theta(t)). \tag{33}$$

The function $\alpha(t)$ satisfy the following differential equation:

$$b_0\frac{\mathrm{d}^2\alpha(t)}{\mathrm{d}t^2} + b_1\frac{\mathrm{d}\alpha(t)}{\mathrm{d}t} + \alpha(t) = z(t), \tag{34}$$

where $z(t)$ is close to a step function and is given by:

$$z(t) = \begin{cases}1, & \text{if } \exists t_1 \in [0,t] : \theta(t_1) = \mathit{atan2}(e_y(t_1), e_x(t_1))\\ 0, & \textit{otherwise}.\end{cases} \tag{35}$$

This way, the robot does not start tracking the virtual robot instantly; it first rotates around its own axis with an increasing angular velocity $\omega_s(t)$, until it “sees” the virtual robot (Fig. 11).

4.5. Simulation results

The validation of the proposed hybrid controller was tested with a lamniscate trajectory using an ordinary backstepping algorithm [13] and a hybrid backstepping algorithm.

The results of the trajectory tracking in both cases are shown in Figs. 12 and 13.

From these figures, it can be concluded that satisfactory tracking results are obtained using both control strategies. However, better tracking of the reference trajectory is achieved

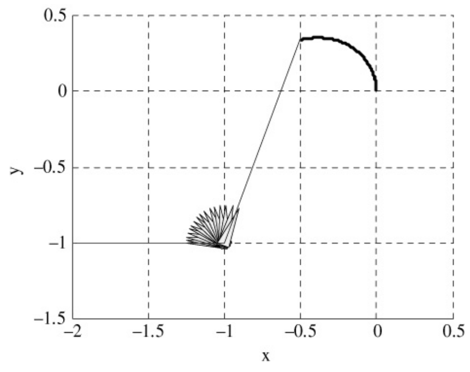

Fig. 11. "Seeking" a virtual robot.

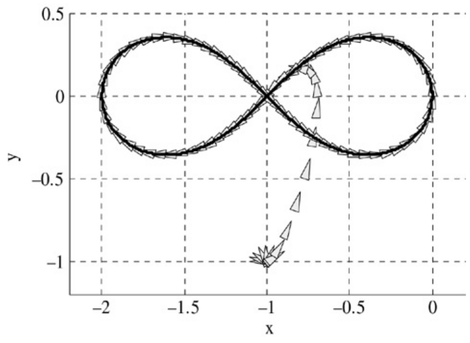

Fig. 12. Tracking a lemniscate trajectory (hybrid).

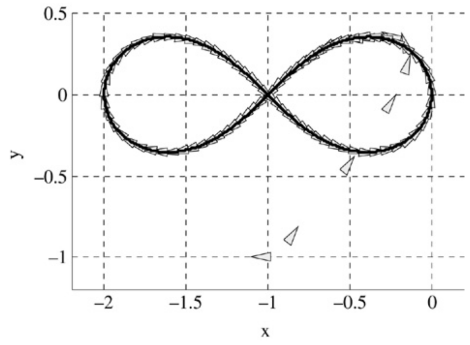

Fig. 13. Tracking a lemniscate trajectory (ordinary).

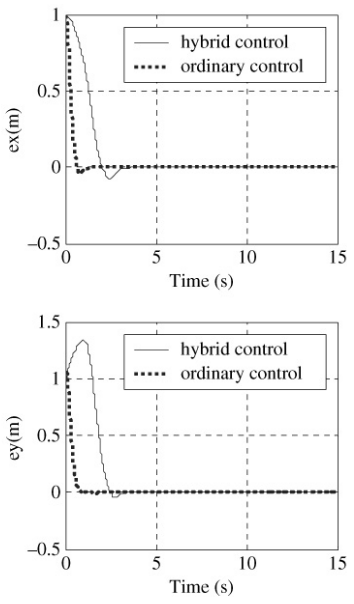

Fig. 14. X and Y coordinate error.

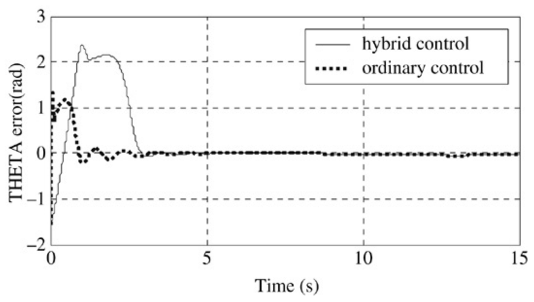

Fig. 15. Orientation error.

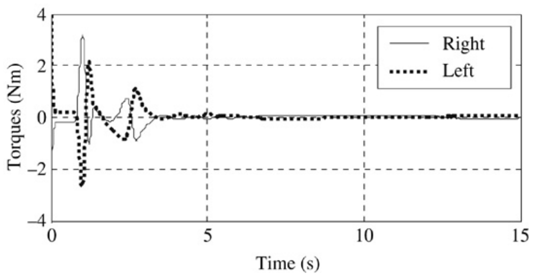

Fig. 16. Control torques (hybrid controller).

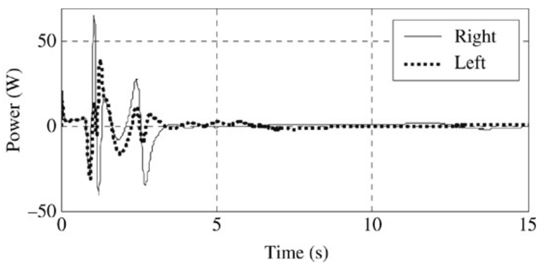

Fig. 17. Required power of DC motors (hybrid controller).

in the case of an ordinary controller, especially at the beginning of transient process. This is illustrated in Figs. 14 and 15. However, the hybrid controller ensures that much smaller values of the control input torques are needed to obtain the reference position and orientation trajectories (Figs. 16 and 18). Consequently, the required power of the DC motors is also much smaller in the case of a hybrid controller (Figs. 17 and 19).

Finally, the comparison of Cartesian error norm profiles is presented for various friction coefficients in Fig. 20. This experiment has also been performed for a lemniscate path. It is obvious that the increase in friction causes degradation of the control system performance.

The next section describes a map-building process which uses Dempster–Shafer evidence theory.

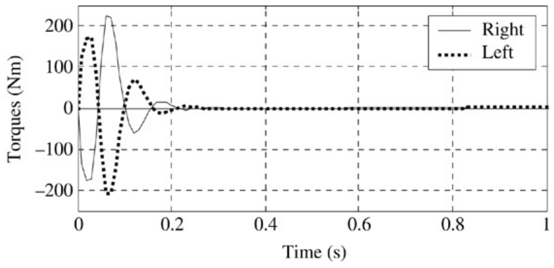

Fig. 18. Control torques (ordinary controller — the first second of tracking).

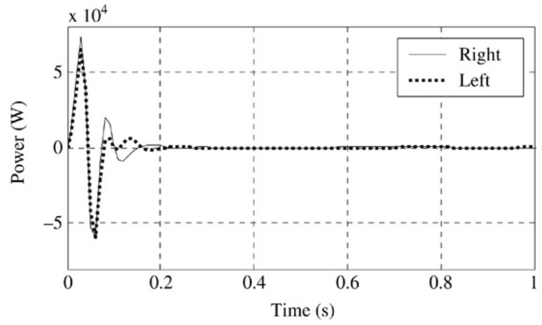

Fig. 19. Required power of DC motors (ordinary controller — the first second of tracking).

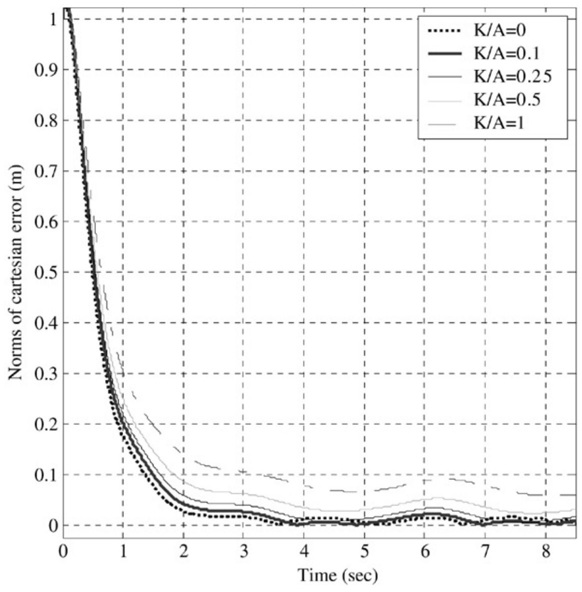

Fig. 20. Norms of Cartesian error for various friction coefficients.

5. Map building

An occupancy grid is essentially a data structure that indicates the certainty that a specific part of space is occupied by an obstacle. It represents an environment as a two-dimensional array. Each element of the array corresponds to a specific square on the surface of the actual world, and its value shows the certainty that there is some obstacle there. When new information about the world is received, the array is adjusted on the basis of the nature of the information.

Here, the proposed map-building process utilizes Dempster–Shafer evidence theory. The sonar sensor readings are interpreted by this theory and used to modify the map using Dempster's inference rule. Whenever the robot moves, it catches new information about the environment and updates the old map.

More specifically, to build an occupancy map of the environment, we first construct a grid representing the whole space. Every discrete region of the map (each cell) may be in two states: *Empty* and *Full*. A series of range readings $\{r_1, \ldots, r_n\}$ collected at known sensor locations is available. In principle, the task of the Map Building System is to process the readings in order to assess, as accurately as possible, which cells are (even partially) occupied by obstacles and which cells are (completely) empty and thus suitable for robot navigation.

5.1. Review of Dempster–Shafer's theory of evidence

The theory of evidence appears in the definition of a frame of discernment, Θ. This is a set of labels representing mutually exhaustive events. As described in [19,20], the relevant labels for map-building applications are $\Theta = \{E, O\}$. This means that a grid cell can be empty (E) or occupied (O). A basic probability assignment is a function $m : \Lambda \to [0, 1]$, where Λ is the set of all subsets of Θ; in our case $\Lambda = \{\phi, E, O, \{E, O\}\}$. The state of each cell is described by assigning basic probability values to each label in Λ. However, we know that, for each cell i, j in the grid,

$$m_{i,j}(\phi) = 0, \tag{36}$$

$$m_{i,j}(A) = \sum_{A \subset \Psi} m_{i,j}(\phi) + m_{i,j}(E) + m_{i,j}(O) + m_{i,j}(\{E, O\}) = 1. \tag{37}$$

Thus, the label A is assigned a basic probability value $m(A)$ describing the degree of belief that is committed to exactly A.

Every cell in the map is first initialized, as follows:

$$m_{i,j}(E) = m_{i,j}(O) = 0, \quad m_{i,j}(\{E, O\}) = 1. \tag{38}$$

A basic probability assignment usually defines the current four states of each cell. However, the number of states can be reduced to two $(m_{i,j}(E), m_{i,j}(O))$, assuming that $m_{i,j}(\phi) = 0$ and applying (37). The state $(0, 0)$ means total ignorance, and so $m_{i,j}(\{E, O\}) = 1$. When the robot is sure about the cell occupancy, then $m_{i,j}(O) = 1$, and that makes the other labels equal to zero. Conversely, $m_{i,j}(E) = 1$ when the robot is sure that a cell is empty.

5.2. Sensor modeling and measurements interpretation

The Polaroid Ultrasonic Rangefinder (Polaroid, 1987) is used for map building. This is a very common device that can detect distances in the range of 0.47–10 m with 1% accuracy. The multi-lobed power diagram of the transmitter is obtained from the radiation directivity function of a plane circular piston:

$$D(\vartheta) = 2\frac{J_1(\omega\eta)}{\omega\eta \sin\vartheta} \tag{39}$$

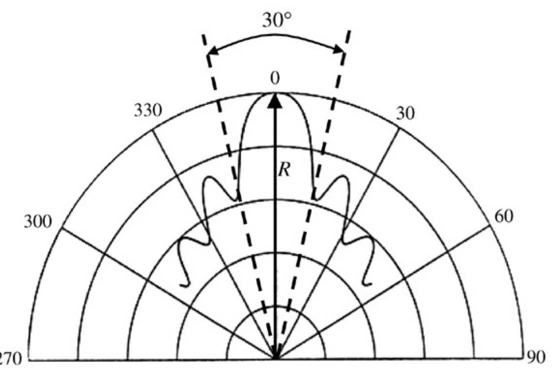

Fig. 21. Typical beam pattern of Polaroid ultrasonic sensor.

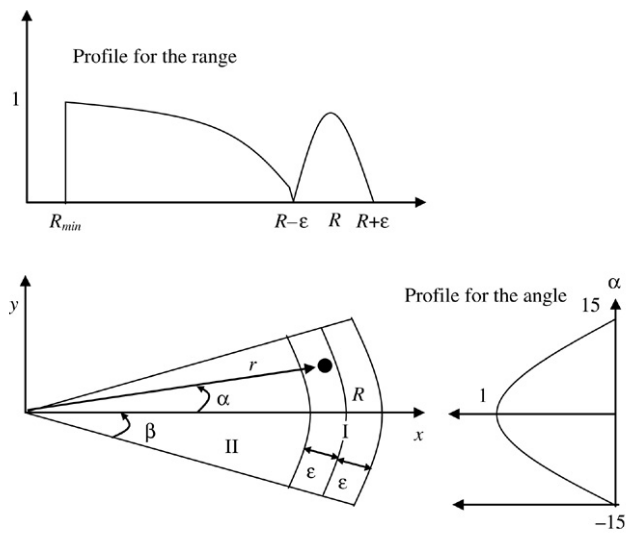

Fig. 22. The profile of the ultrasonic sensor model.

where $J_1(\cdot)$ is the Bessel function of the first order, $\omega = 2\pi/\lambda$ depends on the wavelength λ, η is the piston radius, and ϑ is the azimuthal angle measured with respect to the beam's central axis. For the particular sensor that we used, $\eta = 0.01921$ m and $\lambda = v/\varphi$, where v is the sound speed in air and $\varphi = 49.410$ kHz. In practice, it is sufficient to take into account only the principal lobe of the pattern, and consider the waves to be diffused over a radiation cone of 30° width (Fig. 21).

A single reading provides the information that one or more obstacles are located somewhere along the 30° arc of circumference of the radius R (Fig. 21). Hence, there is evidence that cells located in the proximity may be 'occupied'. On the other hand, cells well inside the circular sector of radius R are likely to be 'empty'. To model this knowledge, we use the Dempster–Shafer theory of evidence.

The evidence is obtained by projecting the raw ultrasonic sensor responses onto the evidence grid through the sonar profile model. This profile sonar model is a function of the angle and the sonar range reading, as shown in Fig. 22.

The model converts the range information into probability values. The model in Fig. 22 is given by Eqs. (40)–(45).

In region I, where $R - \varepsilon < r < R + \varepsilon$:

$$m(O) = \frac{\left(\frac{\beta-\alpha}{\beta}\right)^2 + \left(\frac{\varepsilon-|R-r|}{\varepsilon}\right)^2}{2}, \tag{40}$$

$$m(E) = 0, \tag{41}$$

$$m(\{E, O\}) = 1 - m(O). \tag{42}$$

In region II, where $R_{\min} < r < R - \varepsilon$:

$$m(O) = 0, \tag{43}$$

$$m(E) = \frac{\left(\frac{\beta-\alpha}{\beta}\right)^2 + \left(\frac{R-\varepsilon-r}{R-\varepsilon}\right)^2}{2}, \tag{44}$$

$$m(\{E, O\}) = 1 - m(E). \tag{45}$$

R is the range response from the ultrasonic sensor, and (r, α) is the coordinate of a point inside the sonar cone. ε is the range error, and it distributes the evidence in Region I. β is the half open beam angle of the sonar cone.

5.3. *The fusion of sensor data*

The sonar data interpreted by Dempster–Shafer's theory of evidence are collected and updated into a map using the same theory of evidence. In our approach, the basic probability assignment corresponding to a range reading r is obtained directly using Eqs. (40)–(45).

Finally, each cell in the map is updated using Dempster's rule of combination. This rule allows us to combine independent evidence about a certain event A, $m_1(A)$ and $m_2(A)$. In our case, they will be the mentioned basic probability assignments for the old map belief $m_M^{t-1}(A)$ and the current sensor reading $m_S^t(A)$. For the events E and O, the update rules are:

$$\begin{aligned} m_M^t(E) &= m_M^{t-1} \oplus m_S^t(E) \\ &= \frac{m_M^{t-1}(E)m_S^t(E) + m_M^{t-1}(E)m_S^t(\{E, O\})}{1 - m_M^{t-1}(E)m_S^t(O) - m_M^{t-1}(O)m_S^t(E)} \\ &\quad + \frac{m_M^{t-1}(\{E, O\})m_S^t(E)}{1 - m_M^{t-1}(E)m_S^t(O) - m_M^{t-1}(O)m_S^t(E)} \end{aligned} \tag{46}$$

$$\begin{aligned} m_M^t(O) &= m_M^{t-1} \oplus m_S^t(O) \\ &= \frac{m_M^{t-1}(O)m_S^t(O) + m_M^{t-1}(O)m_S^t(\{E, O\})}{1 - m_M^{t-1}(E)m_S^t(O) - m_M^{t-1}(O)m_S^t(E)} \\ &\quad + \frac{m_M^{t-1}(\{E, O\})m_S^t(O)}{1 - m_M^{t-1}(E)m_S^t(O) - m_M^{t-1}(O)m_S^t(E)}. \end{aligned} \tag{47}$$

The next example illustrates the usage of Dempster–Shafer's theory of evidence for the map-building process.

Example 1. Let the robot be located in cell (14, 10) in the beginning of the navigation process. The basic probability assignments of the map cells have zero values before the start. Then, sonars scan the environment and sonars S_0 and S_1 detect some possible obstacles on distance $r_0 = 7$ and $r_1 = 5$, with angles $\alpha_0 = 0$ and $\alpha_1 = 3$ (Fig. 23). In this example, $R = 8$ and $\varepsilon = 1.5$.

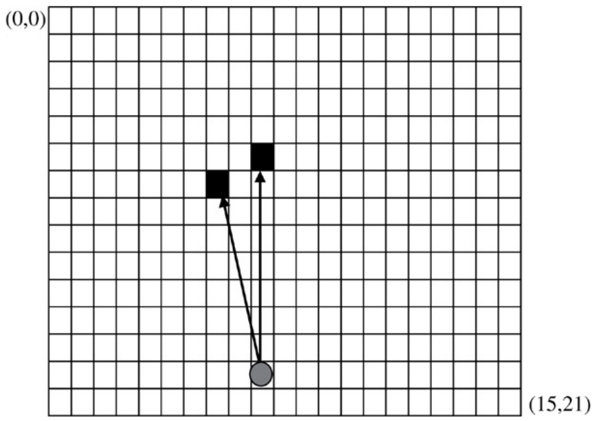

Fig. 23. Initial position.

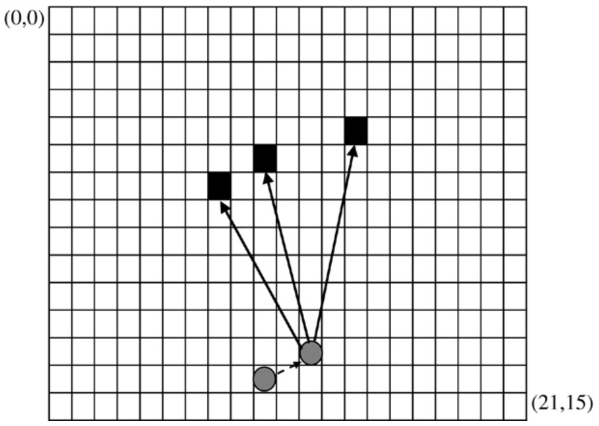

Fig. 24. Occupancy determination based on sonar measurements in $t = 2$ s.

The new basic probability assignments for cells (8, 8) and (6, 10) in the map become (the first lies in region I and the second lies in region II):

$$m_{8,8}^{S_0(7)}(O) = \frac{\left(\frac{15-0}{15}\right)^2 + \left(\frac{1.5-|8-7|}{1.5}\right)^2}{2} = 0.556$$
$$m_{8,8}^{S_0(7)}(E) = 0$$
$$m_{8,8}^{S_0(7)}(\{E, O\}) = 1 - m_{8,8}^{S_0(7)}(Z) = 0.444,$$

and

$$m_{6,10}^{S_1(5)}(O) = 0$$
$$m_{6,10}^{S_1(5)}(E) = \frac{\left(\frac{15-3}{15}\right)^2 + \left(\frac{8-1.5-5}{10-1.5}\right)^2}{2} = 0.336$$
$$m_{6,10}^{S_1(5)}(\{E, O\}) = 1 - m_{6,10}^{S_1(5)} = 0.664.$$

In the next moment, the robot is moving in a direction depicted by the broken line and sonars scan the environment again. This situation is given in Fig. 24.

The possibilities that cells (8, 8), (6, 10) and (5, 14) are occupied is based on sonar measurements (S_0, S_1 and S_{15} ($r_0 = 6.5, r_1 = 7, r_{15} = 7.5, \alpha_0 = 2, \alpha_1 = 5$ and $\alpha_{15} = 4$)):

$$m_{8,8}^{S_0(6.5)}(O) = \frac{\left(\frac{15-2}{15}\right)^2 + \left(\frac{1.5-|8-6.5|}{1.5}\right)^2}{2} = 0.376$$
$$m_{8,8}^{S_0(6.5)}(E) = 0$$
$$m_{8,8}^{S_0(6.5)}(\{E, O\}) = 1 - m_{8,8}^{S_0(6.5)}(Z) = 0.624,$$
$$m_{6,10}^{S_1(7)}(O) = \frac{\left(\frac{15-5}{15}\right)^2 + \left(\frac{1.5-|8-7|}{1.5}\right)^2}{2} = 0.278$$
$$m_{6,10}^{S_1(7)}(E) = 0$$
$$m_{6,10}^{S_1(7)}(\{E, O\}) = 1 - m_{6,10}^{S_1(7)} = 0.722,$$
$$m_{5,14}^{S_{15}(7.5)}(O) = \frac{\left(\frac{15-4}{15}\right)^2 + \left(\frac{1.5-|8-7.5|}{1.5}\right)^2}{2} = 0.491$$
$$m_{5,14}^{S_{15}(7.5)}(E) = 0$$
$$m_{5,14}^{S_{15}(7.5)}(\{E, O\}) = 1 - m_{5,14}^{S_{15}(7.5)} = 0.509.$$

Evidence about occupancies of cells (8, 8) and (6, 10) are updated using Dempster's rules of combination (46) and (47):

$$m_{8,8}^{t}(E) = m_{8,8}^{t-1} \oplus m_{8,8}^{t}(E) = 0$$
$$m_{8,8}^{t}(O) = m_{8,8}^{t-1} \oplus m_{8.8}^{t}(O) = 0.723$$
$$m_{8,8}^{t}(\{E, O\}) = 0.277,$$
$$m_{6,10}^{t}(E) = m_{6,10}^{t-1} \oplus m_{6,10}^{t}(E) = 0$$
$$m_{6,10}^{t}(O) = m_{6,10}^{t-1} \oplus m_{6.10}^{t}(O) = 0.519$$
$$m_{6,10}^{t}(\{E, O\}) = 0.481,$$
$$m_{5,14}^{t}(Z) = 0.491$$
$$m_{5,14}^{t}(P) = 0$$
$$m_{5,14}^{t}(\{P, Z\}) = 0.509.$$

From the new results it can be concluded that the evidence about the occupancy of cells (8, 8) and (6, 10) is increased. This process continues until the target is reached.

In the following section, the new path-planning algorithm is introduced.

6. Path planning

The solution of the obstacle avoidance problem in this paper is a modified principle of acting attractive and repulsive forces.

6.1. Calculation of attractive and repulsive forces

In this approach, environment obstacles exert a repulsive force on the robot and the target position exerts an attractive force. The amplitude of the repulsive force is defined as

$$F = \frac{1}{R_r^2}, \tag{48}$$

where R_r is the distance between the current robot location and the obstacle location. In this situation, the planning algorithm

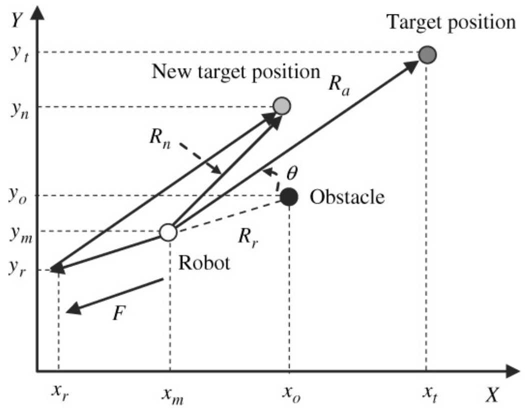

Fig. 25. The concept of path planning based on repulsive and attractive forces.

calculates the following values (Fig. 25):

$$\begin{aligned} \Delta x_o &= x_o - x_m \\ \Delta y_o &= y_o - y_m \\ R_r &= \sqrt{\Delta x_o^2 + \Delta y_o^2}, \end{aligned} \tag{49}$$

where Δx_o and Δy_o are coordinate distances between the mobile robot position and the obstacle position. The projections of force F on x and y axis are:

$$F_x = |F \cdot \cos(\theta)|, \tag{50}$$

$$F_y = |F \cdot \sin(\theta)|, \tag{51}$$

where θ denotes the angle between the robot velocity direction and the repulsive force direction:

$$\theta = \arctan \frac{\Delta y_o}{\Delta x_o}. \tag{52}$$

The coordinates of the vector of the repulsive force are:

$$x_r = \begin{cases} x_m + F_x, & \text{for } \Delta x_o < 0 \\ x_m - F_x, & \text{for } \Delta x_o > 0, \end{cases} \tag{53}$$

$$y_r = \begin{cases} y_t + F_y, & \text{for } \Delta y_p < 0 \\ y_t - F_y, & \text{for } \Delta y_p > 0. \end{cases} \tag{54}$$

The attractive force is defined using R_a as

$$\begin{aligned} \Delta x_t &= x_t - x_m \\ \Delta y_t &= y_t - y_m \\ R_a &= \sqrt{\Delta x_t^2 + \Delta y_t^2}. \end{aligned} \tag{55}$$

In this paper, we proposed the following law for attractive forces:

$$F_a = \begin{cases} \kappa |R_a|, & \text{for } |R_a| \geq 1 \text{ m} \\ |R_a|, & \text{for } |R_a| < 1 \text{ m}, \end{cases} \tag{56}$$

where $\kappa = 5$. This parameter is involved to reach the target when obstacles (relevant obstacles which satisfy pruning rules in the next subsection) are located in the circle with radius 1 m and then they produce strong repulsive forces. The discontinuity at a point of 1 m causes the orientation of the robot to be slightly modified, since the new intermediate target point is moved to another location. This parameter is related to the radius R of the sonar model. In the following consideration, the R is 4 m. In the case of a larger value of R, another value of κ will be selected.

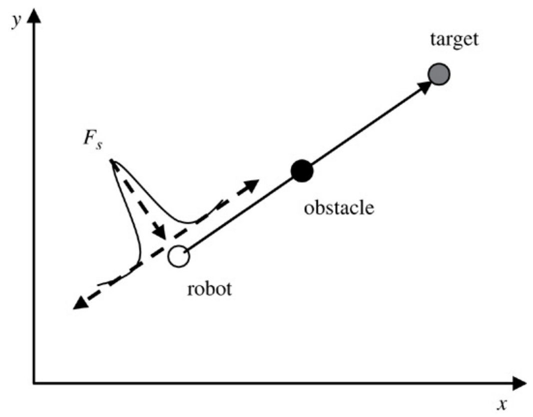

Fig. 26. Stochastic force acting.

The resulting vector, which determines the new target position of the robot, is obtained as a sum of the vectors of repulsive and attractive forces (Fig. 25). The new target coordinates of the robot are:

$$\begin{aligned} x_n &= x_r + \Delta x_t \\ y_n &= y_r + \Delta y_t. \end{aligned} \tag{57}$$

To avoid a possible standstill which may arise if the robot, the obstacle and the target aligned, we introduce a stochastic force orthogonal to the planned trajectory when the sum of forces is zero, and the robot is not yet at the target. This situation is shown in Fig. 26 and explained by fuzzy rules (58).

The fuzzy rules for action of the stochastic force are:

$$\begin{aligned} &\text{If } |F_a| = |F| \text{ and } \textit{the angle between } F_a \text{ and } F \\ &\quad \textit{is less then } 0.01 \text{ rad} \\ &\text{then } F_s \textit{ is big and acts orthogonally on the robot}. \\ &\textit{In other cases}, \ F_s \textit{ is small}. \end{aligned} \tag{58}$$

In the following subsection, the proposed concept of the pruning of relevant obstacles is demonstrated.

6.2. *Pruning of relevant obstacles*

The number of obstacles in the environment may be large. If all the obstacles exert a repulsive force, the calculation of the field is extensive, and yet it may produce erratic trajectories. Therefore, we introduce a pruning of obstacles with an aim to preserve only those relevant for the trajectory planning. This pruning is performed in a set of steps. In these steps, we use the concept of the *visibility field* of the mobile robot [24,25].

The practical visibility field of a mobile robot here is a circle with the radius $r_v = 4$ m. However, in our approach we use only a square inscribed in this circle. The reason is that the local map has $m \times m$ points. This local map is now represented as an $m \times m$ matrix, and each cell in the map is one matrix element.

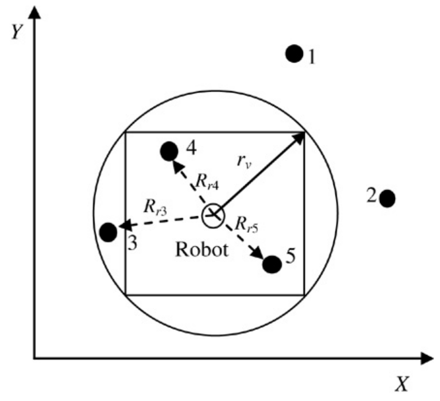

Fig. 27. The visibility region of a mobile robot using sonar sensors.

Due to this geometry, some obstacles will be in visibility in the field given by

$$R_r < r_v. \tag{59}$$

but they will not be taken into account. For example, in Fig. 27, the obstacles 3, 4 and 5 satisfy (59). However, only obstacles 4 and 5 remain as relevant.

Next, all the obstacles in the square are not equally relevant to the future robot motion towards the target. To pinpoint the most relevant, we first define a circular area around the target, named *target circle* with a radius R_a [24]. The value R_a is slightly larger than the distance between the robot and the target, as represented in Fig. 28 ($R_a + 0.1$). The extension of 0.1 ensures a safe region around the robot. We consider that only the obstacles in the section where the target circle and the visibility square overlap are really relevant to the motion planning. Then, among these obstacles, we take the one nearest to the robot. The distance between the target and the obstacle is R_p:

$$R_p = \sqrt{(x_t - x_o)^2 + (y_t - y_o)^2}. \tag{60}$$

In our approach, the rule for the visibility of an obstacle inside the square is given as:

If $((R_a + 0.1) > R_p)$ and $(R_r < r_v)$ then
the obstacle is visible and it should influence the motion. (61)

In Fig. 28, only obstacle 3 influences the new target point determination.

The further reduction of obstacles is still possible. An obstacle located in the above-defined area (61) but behind the current robot position, or behind the target, and not on the direction of motion should not be taken into account. The force produced by such an obstacle can be strong, but obviously it should be neglected. This situation is illustrated in Fig. 29.

To solve this problem, we introduce a new circle around the robot with a radius equal to the distance from the robot to the target (Fig. 30). In this way, the obstacles located behind the target and outside this circle are not considered and the robot will not be pushed aside to go back.

So, we introduce a new restriction in the final visibility rule:

If $((R_a + 0.1) > R_p)$ and $(R_r < r_v)$ and $(R_r < R_a)$ then
the obstacle is visible and it should influence the motion. (62)

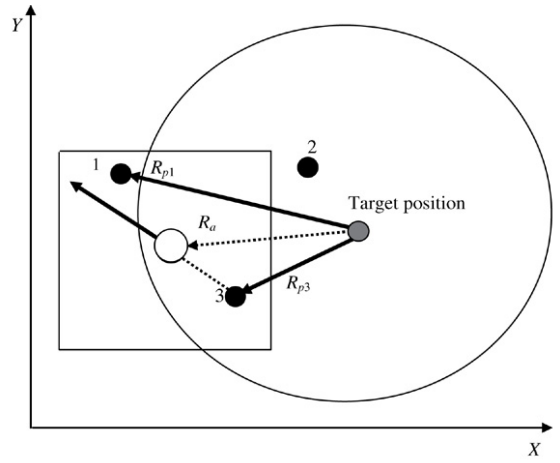

Fig. 28. The forces acting on the robot motion.

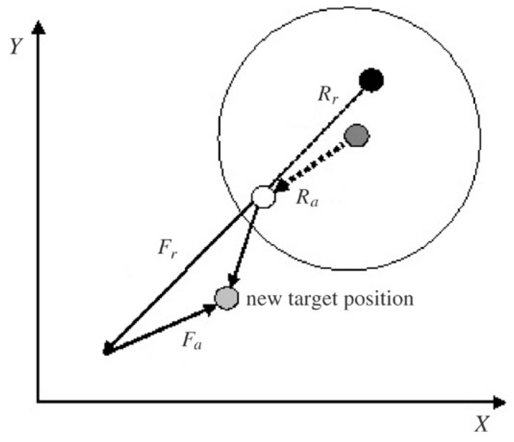

Fig. 29. The strong forces produced by an obstacle that is behind and near the target. This force pulls the robot out from the target.

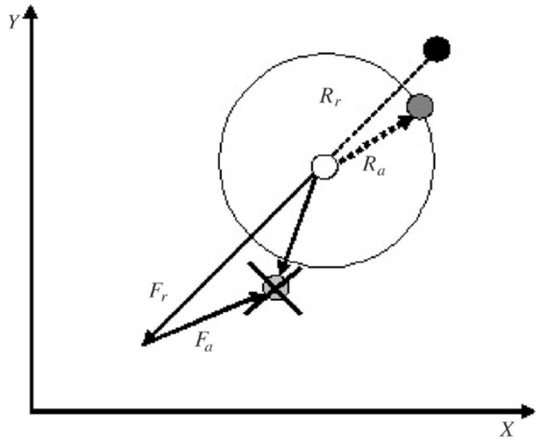

Fig. 30. Introduction of a new circle around the target to avoid the influence of the strong repulsive force.

Finally, in our example (Fig. 31) we have only one relevant obstacle 3.

In the case that no obstacles are located on the robot motion towards the target, its velocity is governed by the position controller. When the distance between the robot and relevant

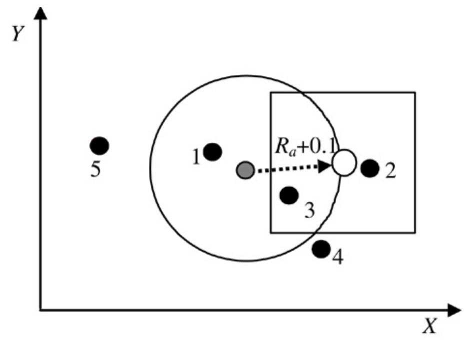

Fig. 31. General concept of determination of new target position.

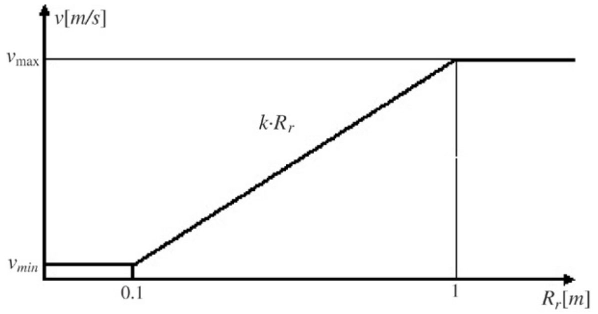

Fig. 32. Dependence of robot velocity on distance between the robot and the relevant obstacle.

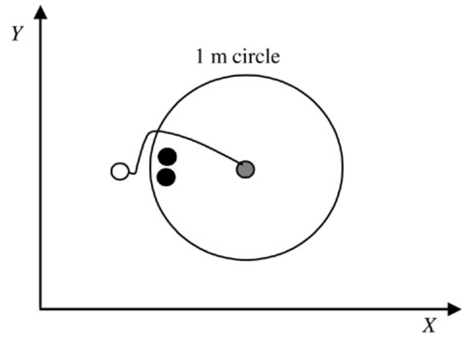

Fig. 33. Movement when two relevant obstacles are in a 1 m circle.

obstacles, which satisfy above conditions, is less then 1 m, we introduced the parameter k, which multiplies the velocities of the left and the right wheels. Using this parameter, possible collisions between the robot and obstacle and the robot turning over are avoided. The graph of robot velocity versus the distance between the robot and the relevant obstacle is shown in Fig. 32.

In the situation represented in Fig. 33, two obstacles are placed in a 1 m circle. The robot cannot pass between them and it continues towards the target during a maneuver around the obstacle so that it gets further away from the robot location (smaller repulsive force). Hence the robot does not enter into an oscillatory movement.

In the rest of the section, the steps for a local minima problem solution are introduced.

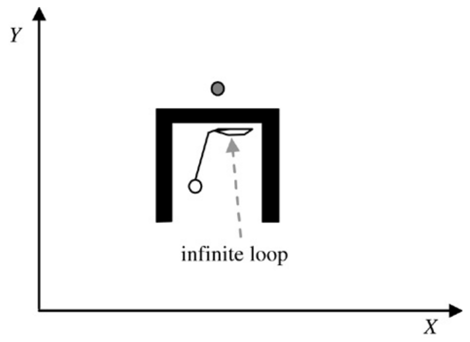

Fig. 34. Illustration of a local minima problem.

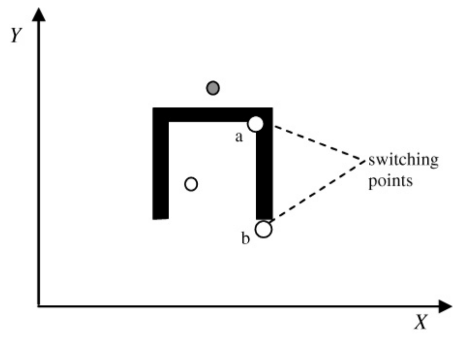

Fig. 35. Identification of point a, in which the robot is trapped.

6.3. Local minima problem solving

For local minima problem avoidance (Fig. 34), we introduce an algorithm that does not require memorizing the already seen environment. This algorithm is capable of identifying the local minima situation during the robot's motion, by recognizing its trapped state (infinite loop). In this state, the robot oscilates between two points.

This algorithm is described as follows:

Algorithm for a local minima avoidance problem.

```
Step 1: Calculate attractive and reflexive forces and their sum
        (resultant force)
Step 2: If resultant force tends to push robot to go back then
        do not take into consideration the attractive force and
        keep the reflexive force constant and equal to the
        closest obstacle distance
        else apply the principle of attractive and reflexive
        forces
end
Step 3: if robot has not reached target point then
        go to Step 1
end
```

The above steps are executed until the robot reaches the target. The algorithm turns the robot right at **switching point a** (point at which the robot tends to go back) and the robot continues to see the wall on its left until it arrives at **switching point b** after it goes towards a target point using the principle of attractive and reflexive forces (Fig. 35).

Simulation results of local minima avoidance using the proposed algorithm are given in Fig. 36. In this case, the

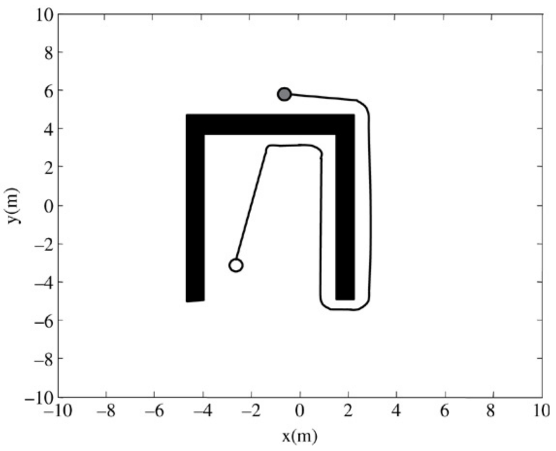

Fig. 36. Simulation results of local minima problem solution using the proposed algorithm.

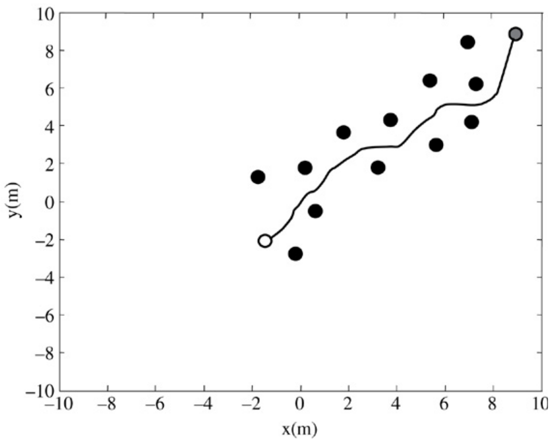

Fig. 37. Obstacle avoidance moving to the target.

proposed algorithm executes a sequence of steps that pulls it out of the trap.

In addition, more simulations are performed in the next section to present the effectiveness of the proposed approach in cluttered environment and environments with obstacle loops and mazes.

7. Simulation results

To show the usefulness of the proposed approach, a series of simulations has been conducted using an arbitrarily constructed environment including obstacles. The robot has been modeled as a small circle, and imaginary sensors (eight in number) are placed in the form of an arc along the circumference of the robot. The minimum distance obtained within the cone of each sensor is considered as the distance of the obstacle from that sensor, and it is available as an input to the algorithm. All of the simulations are performed in the MATLAB/SIMULINK environment.

In these simulations, the positions of all the obstacles in the workspace are unknown to the robot. The robot is aware of its start and finish positions only. Simulation results obtained in seven different working scenarios (with different obstacle constellations) are presented in Figs. 37–44.

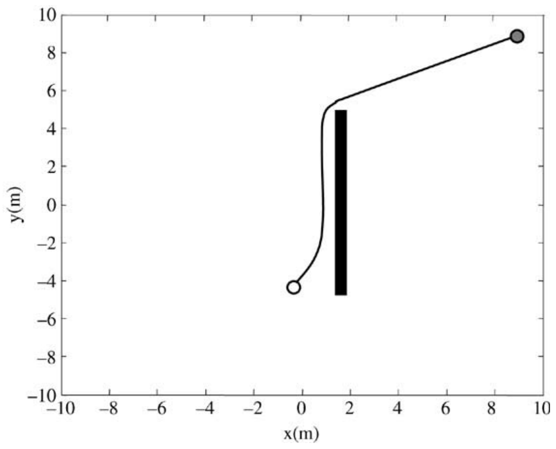

Fig. 38. Wall following.

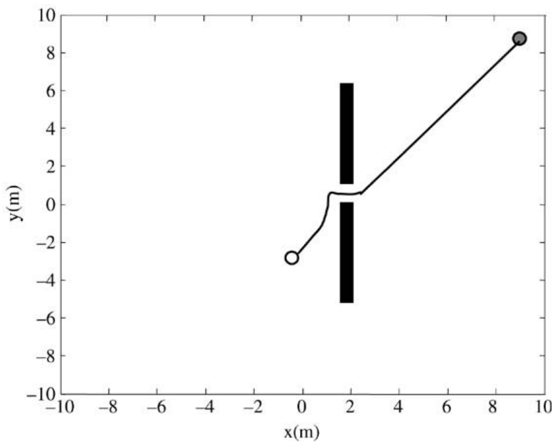

Fig. 39. Passing through a narrow door.

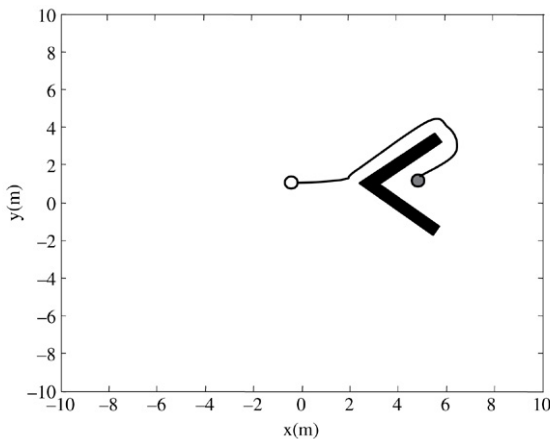

Fig. 40. Finding a target partially surrounded by obstacles.

The proposed algorithm performs well in a cluttered environment, as shown in Fig. 37.

The processes for wall following and passing through a narrow door using our algorithm are shown in Figs. 38 and 39, respectively.

From the results obtained, it can be concluded that the robot successfully tracks the wall and passes through the narrow door.

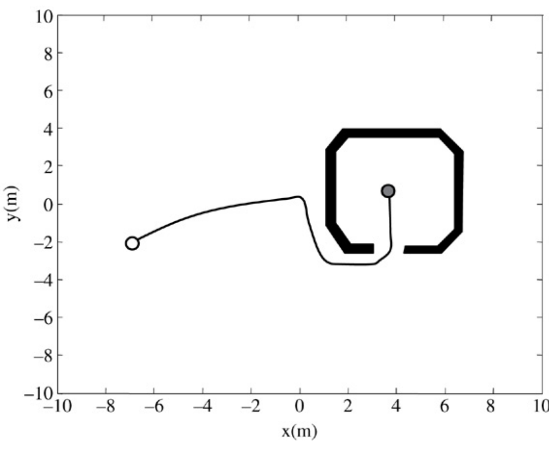

Fig. 41. Finding a target surrounded by obstacles.

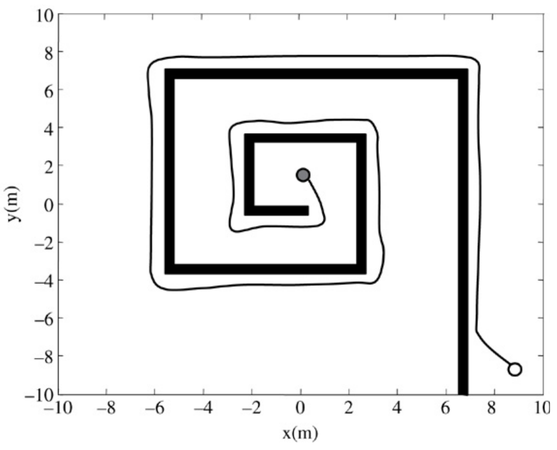

Fig. 42. Finding a target under a long loop obstacle.

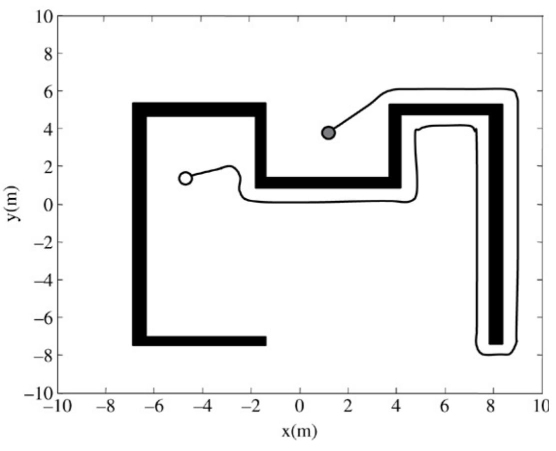

Fig. 43. Finding a target in a maze.

In all of the cases in which the target is surrounded by cluttered obstacles, which form the convex loop, the proposed algorithm, described by Eqs. (48)–(62), helps the robot to find the target (Figs. 40 and 41). The algorithm works very well in the case when a cluttered environment consists of walls in the form of long loops, which is always convex in order to current robot location (Fig. 42).

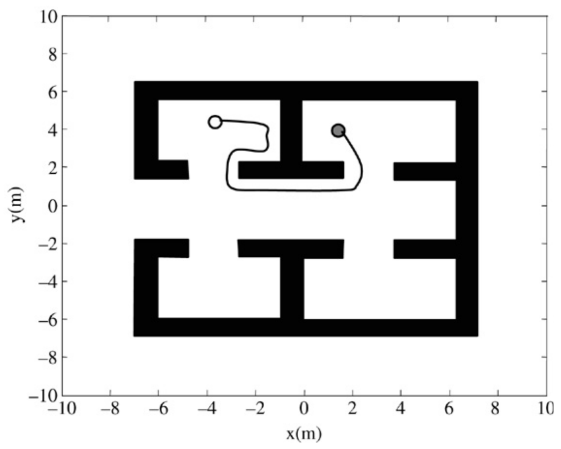

Fig. 44. Finding a target in a more cluttered maze.

A new problem arrises if the environment consists of concave obstacles and the robot gets into an infinite loop or into a local minimum. An even more complex problem arises when the obstacles are long and have many bends and kinks; the target attracting and goal repulsing behavior conflict and the robot also gets itself into an infinite loop (Fig. 43). The effectiveness of the proposed algorithm for solving this problem is illustrated in the same figure. In this way, the robot does not get trapped. Finally, the behavior of the mobile robot in a more cluttered environment is illustrated in Fig. 44.

These results clearly show that this system is able to navigate in complex environments such as mazes.

8. Conclusions

In this paper, we present a new mobile robot navigation strategy based on the soft computing methodologies in an a priori unknown environment. The aim was to obtain robust robot control suitable for on-line applications with real-time requirements. For this purpose, soft computing methodologies, such as fuzzy logic and other reasoning techniques, were successfully used.

A dynamic model of a mobile robot with nonholonomic constraints is derived first. The special feature of this model is that the main variables are the angular velocities of the wheels. Due to this approach, the impossibility of lateral motion is embedded into the model. In addition, such a model is easily simulated. The proposed navigation system consists of three control subsystems.

The low-level control of robot velocities is designed using PI controllers. The efficiency of this controller is demonstrated by various trajectory trackings.

The coefficients of the medium-level nonlinear backstepping position controller are adjusted by a genetic algorithm. This controller reaches good position-tracking performance, but control torque values are too big. To solve this problem, a new control law is introduced which provides velocity servo inputs, and in this way it significantly decreases the control input torques.

The high-level control system consists of sonar data interpretation and fusion, map building, and path planning.

The sonar data interpretation and fusion were realized using the Dempster–Shafer theory of evidence. The local map is represented by an occupancy grid with the updates obtained by Dempster rules of combination.

An efficient local path planning algorithm is required to achieve real-time operation. The proposed local planning is based on repulsive and attractive forces and the pruning of relevant obstacles by fuzzy rules. It shows robust and stable performance and simulation results. For the local minima avoidance problem, an algorithm that does not need to memorize previous movement is proposed.

A possible extension of this work is the navigation of a mobile robot in a dynamic environment.

References

[1] J. Borenstein, Y. Koren, The vector field histogram — fast obstacle avoidance for mobile robots, IEEE Transactions on Robotics and Automation 7 (1991).

[2] W. Chiang, J. Lee, Fuzzy Logic for the Applications to Complex Systems, World Scientific, Singapore, 1996.

[3] D. Driankov, H. Hellendoorn, M. Reinfrank, An Introduction to Fuzzy Control, Springer-Verlag, Berlin, 1996.

[4] G. Dudek, M. Jenkin, Computational Principles of Mobile Robotics, Cambridge University Press, Cambridge, 2000.

[5] M. Egerstedt, X. Hu, A. Stotsky, Control of mobile platforms using a virtual vehicle approach, IEEE Transactions on Automatic Control 46 (11) (2001) 1777–1882.

[6] T. Hu, S.X. Yang, Real-time motion control of a nonholonomic mobile robot with unknown dynamics, in: Proceedings of the Computational Kinematics Conference, Seoul, Korea, 2001.

[7] H.P. Huang, P.C. Lee, A real-time algorithm for obstacle avoidance of autonomous mobile robots, Robotica 10 (1992) 217–227.

[8] S. Ishikawa, A method of indoor mobile robot navigation by fuzzy control, in: Proceedings of the International Conference on Intelligent Robot and Systems, Osaka, Japan, 1991, pp. 1013–1018.

[9] M. Kam, X. Zhu, P. Kalata, Sensor fusion for mobile robot navigation, Proceedings of the IEEE 85 (1) (1997) 108–119.

[10] Y.C. Kim, S.B. Cho, S.R. Oh, Map-building of a real mobile robot with GA-fuzzy controller, International Journal of Fuzzy Systems 4 (2) (2002) 696–703.

[11] B. Kosko, Neural Networks and Fuzzy Systems, Prentice Hall, NJ, 1992.

[12] K.M. Krishna, P.K. Karla, Solving the local minima problem for a mobile robot by classification of spatio-temporal sensory sequences, Journal of Robotic Systems 17 (10) (2000) 549–564.

[13] B. Lacevic, J. Velagic, B. Perunicic, Evolution of parameters of nonlinear position control for dynamic model of mobile robot with friction, in: Proceedings of the 16th IFAC World Congress, Prague, Czech Republic, 3–8 June 2005, Paper no. 2682.

[14] V.J. Lumelsky, A comparative study on the path performance of maze-searching and robot motion planning algorithms, IEEE Transactions on Robotics and Automation 7 (1991) 57–66.

[15] P. Morin, C. Samson, Application of backstepping techniques to time-varying exponential stabilisation of chained systems, European Journal of Control 3 (1) (1997) 15–36.

[16] U. Nehmzow, Mobile Robotics: A Practical Introduction, in: Series on Applied Computing, Springer-Verlag, London, UK, 2000.

[17] G. Oriolo, G. Ulivi, M. Vendittelli, Real-time map building and navigation for autonomous robots in unknown environments, IEEE Transactions on System, Man and Cybernetics part B 28 (3) (1998) 316–333.

[18] G. Oriolo, A. De Luca, M. Vendittelli, WMR control via dynamic feedback linearization: design, implementation and experimental validation, IEEE Transactions on Control System Technology 10 (6) (2002) 835–852.

[19] D. Pagac, E. Nebot, H. Durrant-Whyte, An evidential approach to probabilistic map-building, IEEE Transactions on Robotics and Automation 14 (4) (1998) 623–629.

[20] M. Ribo, A. Pinz, A comparison of three uncertainty calculi for building sonar-based occupancy grids, Robotics and Autonomous Systems 35 (2001) 201–209.

[21] O. Takahashi, R.J. Schilling, Motion planning in a plane using generalized Voronoi diagrams, IEEE Transactions on Robotics and Automation 5 (1989) 143–150.

[22] H.G. Tanner, K.J. Kyriakopoulos, Backstepping for nonsmooth systems, Automatica 39 (5) (2003) 1259–1265.

[23] S.G. Tzafestas, Advances in Intelligent Autonomous Systems, Kluwer, Dordrecht, Boston, 1999.

[24] J. Velagic, B. Perunicic, M. Hebibovic, N. Osmic, Autonomous mobile robot navigation system based on soft computing methodologies, in: Proceedings of the IEEE International Conference on Mechatronics and Robotics, MechRob2004, 13–15 September 2004, Aachen, Germany, 2004, pp. 695–701.

[25] J. Velagic, B. Lacevic, B. Perunicic, New concept of the fast reactive mobile robot navigation using a pruning of relevant obstacles, in: Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE 2005, 20–23 June 2005, Dubrovnik, Croatia, 2005, pp. 161–166.

[26] A. Zilouchian, M. Jamshidi, Intelligent Control Systems Using Soft Computing Methodologies, CRC Press, Boca Raton, 2001.

Jasmin Velagic Ph.D. received a M.Sc.E.E. degree from the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia in 1999 and a Ph.D. degree from the Faculty of Electrical Engineering, University of Sarajevo, in 2005. He was a teaching assistant with the Faculty of Electrical Engineering, University of Sarajevo, from 1997 until 2001. During 2001–2006, he was a senior assistant at the same faculty. Since 2006, he has been an assistant professor with the Department of Automatic Control and Electronics, University of Sarajevo. His research interests include intelligent control, mobile robotics, marine systems, industrial automation, and adaptive and robust control. Dr. Velagic is a member of IEEE Control System and IEEE Robotics and Automation Societies.

Bakir Lacevic received a Diploma in electrical engineering from the Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina, in 2003. He is a teaching and research assistant at the departments for Control Systems and Computer Science at the same faculty, and he is currently working towards an M.S. degree. His research interests include evolutionary computation, mobile robotics, and computational intelligence in control.

Professor Branislava Perunicic received her Candidate of Sciences degree (an equivalent to Ph.D.) from the Institute of Control Problems of the Academy of Sciences of the USSR in 1968. Her dissertation treated conditions of the existence of sliding modes and invariance conditions in sliding mode motion which occurs in Variable Structure Systems. She is currently professor of Electrical Engineering at the Department of Electrical Engineering of the University of Sarajevo. Her research interests are variable structure control, identification, and soft computing applications in control. She has served as a reviewer of scientific journals. Professor Perunicic is an elected member of the Academy of Sciences of Bosnia and Herzegovina. She is also the IEEE Bosnia and Herzegovina section chair.