10

Neural Networks Based Navigation and Control of a Mobile Robot in a Partially Known Environment

Diana D. Tsankova
Technical University – Sofia, Branch Plovdiv
Bulgaria

1. Introduction

Developing mobile robots able to move autonomously and staying operational in unexplored environments is one of the most important aims of intelligent robotics research. Navigation (here in the sense: collision-free goal following behaviour) depends on the amount of a priori available information. It could be assumed that this information comes in the following three categories: (I) complete information about the robot's workspace is available, (II) a priori knowledge about the environment is not available and the robot would have to rely on its sensors at execution time to obtain the information needed to accomplish the task, (III) partial knowledge of the environment is available and sensors are used at the execution time to acquire additional information. The robot may interweave planning and execution monitoring activities, but the more incomplete the prior knowledge, the less important the role of global planning.

Most of the global path-planning methods, which assume a static and completely known to the robot environment, are considered to belong to or to be variations of a few basic approaches: roadmap, cell decomposition, and potential field methods (Latombe, 1991; Choset et al., 2005). However, the dynamic and complex features of robot environments make their accurate modelling and predicting fundamentally impossible. Hence, during path following a local re-planning is needed in order to make the robot avoid collisions with unknown obstacles. By combining global and local path-planning methods mobile robots are able to operate in dynamic environments (Yahja et al., 2000). Each of the path-planning methods has its own advantages and drawbacks and is more or less proper to be used in a particular situation. How to select the most suitable approach and how to use local path-planning in collaboration with the global one in a practical example, related to a partially known or fast-changing environment - these are still open research problems.

Navigation relies on the tracking and regulation capabilities of the feedback control design (the lower control level of the robot). Standard approaches to non-holonomic control design often deal only with the kinematic steering system, but good performance cannot be obtained without taking into account the vehicle dynamics. A stable control algorithm that considers the complete vehicle dynamics has been developed using back stepping

kinematics into dynamics and the results have been reported in the literature (Fierro & Lewis, 1995; Zulli et al., 1995). According to this approach the dynamics of the vehicle has to be completely known. However, in many practical cases, exact knowledge of the mobile robot dynamics is unattainable, e.g., friction is very difficult to model by conventional techniques. A solution to this problem requires implementation of robust adaptive control methods combining conventional approaches with new learning techniques (Gomi & Kawato, 1990; Gomi & Kawato, 1993), and these techniques have to be suitable for the real time applications.
This research work treats problems that belong to the third of the categories, mentioned above. The proposed integrated three-component system for navigation and control is similar to that, reported in Ref. (Topalov et al., 1998b), but all the three parts are implemented as neural networks here and the overall performance of the system is improved. The global path-planning algorithm seeks the shortest collision-free path by minimizing a cost function using neural penalties for obstacle collisions (Meng & Picton, 1992). The local obstacle avoidance controller is implemented as Braitenberg's No.3c vehicle modified by an artificial emotion mechanism. The emotional intervention significantly improves the performance of the vehicle (Mochida, 1995; Tsankova, 2009). Finally the path planner and local obstacle avoidance controller are integrated with a nonlinear trajectory tracking controller, whose control algorithm is based on a dynamical extension that makes possible the integration of a kinematic controller and a radial basis function (RBF) net based torque controller into an adaptive feedback controller using on-line *feedback-error-learning* scheme. The use of the recursive least squares technique based on Givens QR decomposition (Rosipal et al., 1998) makes the RBF net's weights tuning faster than the learning procedures based on genetic algorithms (Topalov et al., 1997; Topalov et al., 1998a) or backpropagation (Topalov et al., 1998b). The effectiveness of the proposed navigation and control strategies, is confirmed in MATLAB simulations by a collision-free goal following task, presented in different types of environment (including static and dynamic obstacles, a priori known and unknown obstacles).

2. The Task, the Robot and the General Design

The basic effort in this work is directed toward developing a system integrating three control devices (algorithms), which implements a collision-free trajectory tracking behaviour in a partially known environment. The devices (a path planner, a local obstacle avoidance controller and a trajectory tracking controller) are briefly described below.

2.1 General Design Algorithm

The overall structure of the design algorithm is shown in Fig.1. The design algorithm uses three different models of the mobile robot - the geometric, kinematic, and dynamic models. The path planner needs only the geometric model of the robot and the information about the a priori known static obstacles, which are a part of the „partially known" environment. It produces the global collision-free path between the given start and goal positions, which furthermore is transformed by an appropriate interface, linking the path-planning and trajectory tracking devices, into time-indexed sequence of points - reference trajectory. Based on the knowledge of the three models, the trajectory tracking controller is able to find the appropriate commands (torques) for the wheel actuators, so that the robot follows the

desired trajectory. If the robot, tracking the trajectory, encounters an obstacle that has been unknown during the path-planning procedure, then the local obstacle avoidance controller is switched to turn the robot aside from the trajectory and thus to overcome the obstacle. Reactive or behaviour-based control architectures are very appropriate for this purpose. It is evident that the obstacle avoidance controller needs to know the geometric model of the robot and needs to collect information about the a priori unknown static and/or dynamic obstacles.

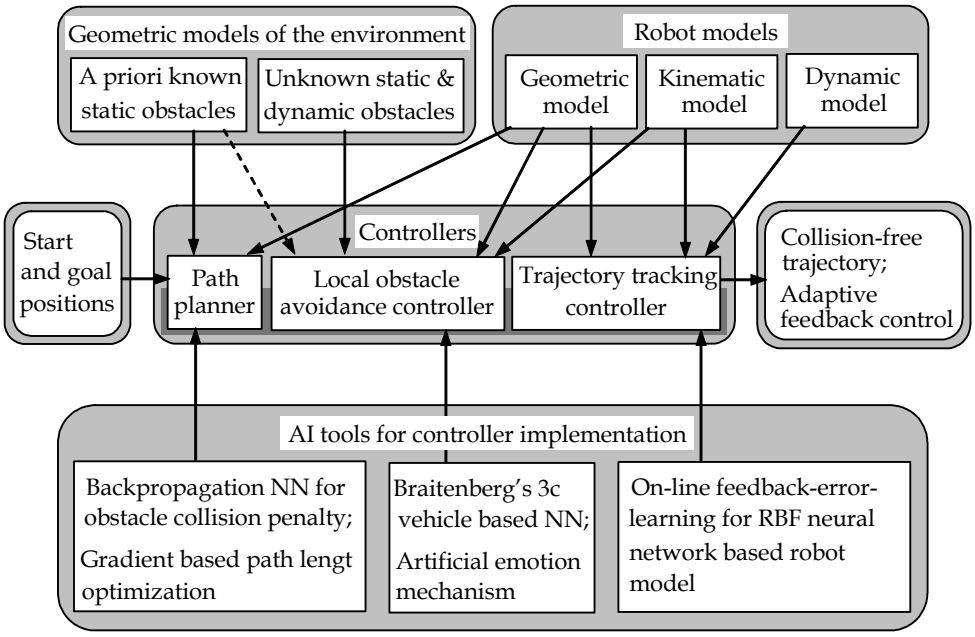


Fig. 1. Structure of the design algorithm

The three types of algorithms, included in the general design, make use of neural networks (NNs) as a powerful tool of artificial intelligence (AI). The path planner generates a path, which is optimal according to a criterion, which guarantees „minimum path length and no collisions with obstacles" (Meng & Picton, 1992). An appropriate gradient based method is used for minimization of the path length together with the penalties for collisions with a priori known obstacles. The penalties are produced using an approximation of obstacle-oriented repulsive potential function, made by a feed-forward neural network, trained by error backpropagation algorithm.

The local obstacle avoidance controller includes the Braitenberg's No.3c architecture (Braitenberg, 1984), implemented as a sum of two NNs: one for an obstacle avoidance behaviour and another for a goal following one. The goal is represented by the corresponding reference trajectory points. To improve the performance, a special type of neural network known as „artificial emotion mechanism" (Mochida et al., 1995) is applied to modulate the weights of the goal-oriented NN.

The trajectory tracking controller represents an adaptive feedback controller using the on-line *feedback-error-learning* scheme (Gomi & Kawato, 1990; Gomi & Kawato, 1993) based on an RBF net based model. The tracking behaviour has to be adaptable and robust to changes in the robot dynamics and/or in the external conditions. The contribution to this controller (in its „model" part) is the use of the recursive least squares technique based on Givens QR decomposition (Rosipal et al., 1998), that makes the RBF net's weights tuning relatively fast and easy.

2.2 Geometric and Kinematic models of the robot

Consider a mobile robot with two driving wheels, mounted on the same axis, and a front free wheel (Fig.2a). The mobile robot is equipped with two kinds of detectors: obstacle detectors and a goal detector. The obstacle detectors are installed in five directions, as shown in Fig.2b. They can detect the existence of obstacles in their directions (sectors S_i, $i = 1,2,...,5$), and the detecting range of sensors is assumed to be equal to the diameter of the robot. The obstacles are physical existing structures, but the goal is a virtual one – it corresponds to the current reference trajectory point. The goal detector is a software detector, which has to calculate the distance and orientation to this „goal" with respect to the mobile base, using information for the reference point and the robot position and heading. After that the simulated goal detector can recognize the direction of the goal at any position of the obstacle detectors.

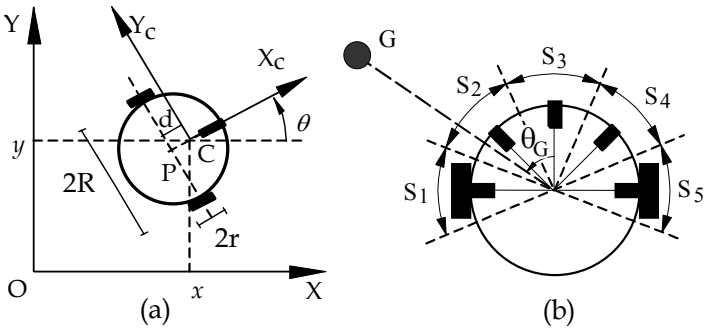


Fig. 2. Geometric model of a non-holonomic mobile robot

The motion and orientation of the robot are achieved by independent actuators (DC motors). The position of the robot in an inertial Cartesian frame $\{O, X, Y\}$ (Fig.2a) is completely specified by the *posture* $\mathbf{q} = (x, y, \theta)^T$ where (x, y) and θ are the coordinates of the reference point C, and the orientation of the mobile basis $\{C, X_C, Y_C\}$ with respect to the inertial basis, respectively. The motion of the mobile robot is controlled by its *linear velocity* υ and *angular velocity* ω, which are also functions of time. The kinematics of the vehicle is defined by Jacobian matrix $\mathbf{J}$, which transforms velocities $\mathbf{v} = (\upsilon, \omega)^T$ expressed in mobile basis into velocities $\dot{\mathbf{q}}$ expressed in a Cartesian one (Fierro & Lewis, 1995):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \dot{\mathbf{q}} = \mathbf{J}\mathbf{v} = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \upsilon \\ \omega \end{bmatrix}, \tag{1}$$

where $|\upsilon| \le \upsilon_{\max}$ and $|\omega| \le \omega_{\max}$. $\upsilon_{\max}$ and $\omega_{\max}$ are the maximum linear and angular velocities of the mobile robot. System (1) is called the steering system of the vehicle.

2.3 Dynamic model of the robot

The dynamical equations of the mobile robot system having an n-dimensional configuration space with generalized coordinates $\mathbf{q} = (q_1, q_2, ..., q_n)^{\mathrm{T}}$ and m constraints can be described by (Fierro and Lewis, 1995):

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}_{\mathrm{m}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \boldsymbol{\tau}_{\mathrm{d}} = \mathbf{B}(\mathbf{q})\boldsymbol{\tau} - \mathbf{A}^{\mathrm{T}}(\mathbf{q})\lambda, \tag{2}$$

where $\mathbf{M}(\mathbf{q}) \in R^{n \times n}$ is a symmetric positive definite inertia matrix, $\mathbf{V}_{\mathrm{m}}(\mathbf{q}, \dot{\mathbf{q}}) \in R^{n \times n}$ is the centripetal and coriolis matrix, $\mathbf{F}(\dot{\mathbf{q}}) \in R^n$ denotes the surface friction, $\mathbf{G}(\mathbf{q}) \in R^n$ is the gravitational vector, $\boldsymbol{\tau}_{\mathrm{d}} \in R^n$ denotes bounded unknown disturbances including unstructured unmodelled dynamics, $\mathbf{B}(\mathbf{q}) \in R^{n \times r}$ is the input transformation matrix, $\boldsymbol{\tau} \in R^r$ is the input vector, $\mathbf{A}(\mathbf{q}) \in R^{m \times n}$ is the matrix associated with the constraints, and $\lambda \in R^m$ is the vector of constraint forces. The dynamical equations of the mobile base presented in Fig.2 can be expressed in the matrix form (2) where

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} m & 0 & md\sin\theta \\ 0 & m & -md\cos\theta \\ md\sin\theta & -md\cos\theta & I \end{bmatrix}, \qquad \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} md\dot{\theta}^2\cos\theta \\ md\dot{\theta}^2\sin\theta \\ 0 \end{bmatrix},$$

$$\mathbf{B}(\mathbf{q}) = \frac{1}{r}\begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ R & -R \end{bmatrix}, \qquad \mathbf{G}(\mathbf{q}) = 0, \qquad \boldsymbol{\tau} = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}, \qquad \mathbf{A}^{\mathrm{T}}(\mathbf{q}) = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ -d \end{bmatrix}, \tag{3}$$

$$\lambda = -m(\dot{x}\cos\theta + \dot{y}\sin\theta)\dot{\theta}.$$

3. The Path Planner

The collision-free path-planning can be stated as: Given an object with an initial position, a desire goal position, and a set of obstacles; the problem is to find a continuous path from the initial position to the goal position, which avoids colliding with obstacles along it. The path-planning procedure proposed in this work is based on the theoretical work of Meng and Picton (1992). The path for the object is represented by a set of N via points. The path

finding algorithm is equivalent to optimizing a cost function, defined in terms of the total path length and the collision penalty, by moving the via points in the direction that minimizes the cost function. A two-layer log-sigmoid/log-sigmoid backpropagation neural network is used to produce the collision penalty. The environment area is divided into 2D grid cells and a binary value is assigned to each grid cell to indicate an obstacle presence: "0" means that the cell is fully unoccupied, and "1" - the cell is occupied. To take into account the robot's geometry the obstacles could be modified by growing its size isotropically by the robot's radius plus a small tolerance.
The x , y coordinates of the cells' centres and the corresponding assigned binary values are used as learning patterns for the 2-input/1-output neural network. The output of the network represents the collision penalty for the current position (x, y) of the object. The collision penalty of a path is defined as the sum of individual collision penalties of all the via points. The energy function for collision is defined as:

$$E_C = \sum_{i=1}^{N} C_i \ , \tag{4}$$

where C_i is the collision penalty for the i^{th} via point. The energy function for the path length is defined as the sum of squares of all the segments' lengths connecting the via points:

$$E_L = \sum_{i=1}^{N} L_i^{\,2} = \sum_{i=1}^{N} [(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2] \,, \tag{5}$$

The total energy is

$$E = E_C + E_L \ . \tag{6}$$

The equation (6) is modified by multiplying both energy functions with positive weight coefficients k_C and k_L to express our preference for the collision criterion or for the path length criterion, respectively (Topalov et al., 1997):

$$E = k_C E_C + k_L E_L \ . \tag{7}$$

The dynamical equation for a via point is chosen to make time derivative of the energy be negative along the trajectory, because the low energy implies less collisions and a shorter path. The time derivative of E is

$$\frac{dE}{dt} = \frac{d}{dt}(k_C E_C + k_L E_L) = \sum_{i=1}^{N} (k_L \frac{\partial L_i^2}{\partial x_i} + k_C \frac{\partial C_i}{\partial x_i}) \frac{dx_i}{dt} + \sum_{i=1}^{N} (k_L \frac{\partial L_i^2}{\partial y_i} + k_C \frac{\partial C_i}{\partial y_i}) \frac{dy_i}{dt} \ . \tag{8}$$

Let

$$\frac{dx_i}{dt} = -(k_L \frac{\partial L_i^2}{\partial x_i} + k_C \frac{\partial C_i}{\partial x_i}), \qquad \frac{dy_i}{dt} = -(k_L \frac{\partial L_i^2}{\partial y_i} + k_C \frac{\partial C_i}{\partial y_i}), \tag{9}$$

then

$$\frac{dE}{dt} = -\sum_{i=1}^{N}[(\frac{dx_i}{dt})^2 + (\frac{dy_i}{dt})^2] < 0 . \tag{10}$$

$dE/dt = 0$ if and only if $dx_i/dt = 0$ and $dy_i/dt = 0$. Therefore, all via points move along trajectories, decreasing the energy, and finally reach equilibrium positions. In (9)

$$\frac{\partial C_i}{\partial x_i} = f'(I2)\sum_{j=1}^{S} W2_j f'(I1_j) W1^x{}_j , \qquad \frac{\partial C_i}{\partial y_i} = f'(I2)\sum_{j=1}^{S} W2_j f'(I1_j) W1^y{}_j , \tag{11}$$

where C_i is the output of the network, $I2$ is the input of the output layer neuron, $I1_j$ is the input of the j[th] hidden layer neuron, S is the number of hidden layer neurons, $W1^x{}_j$ is the weight coefficient of the input "x" with respect to j[th] hidden layer neuron, $W2_j$ is the weight coefficient of j[th] hidden layer neuron's output with respect to the output layer neuron. From (9) and (11), the dynamical equations for x_i and y_i are derived as:

$$\begin{aligned} \frac{dx_i}{dt} &= -[k_L(2x_i - x_{i-1} - x_{i+1}) + k_C f'(I2)\sum_{j=1}^{S} W2_j f'(I1_j) W1^x{}_j] \\ \frac{dy_i}{dt} &= -[k_L(2y_i - y_{i-1} - y_{i+1}) + k_C f'(I2)\sum_{j=1}^{S} W2_j f'(I1_j) W1^y{}_j]. \end{aligned} \tag{12}$$

4. The Local Obstacle Avoidance Controller

Following the trajectory that is based on the path-planning procedure, described in the previous section, the mobile robot would not have an ability to avoid unexpected local obstacles (obstacles whose location or geometry is different from those in the robot's model of the workspace). A solution to this problem is the introduction of a sensor-based local-obstacle avoidance controller. It operates in cooperation with the trajectory tracking controller as it switches and functions *only when* the sensors detect an obstacle. Based on sensor readings the local obstacle avoidance controller calculates the target velocities of the robot and passes them to trajectory tracking controller. The last produces torques for the robot's motion control. As it has been mentioned in Section 2.1, the sensor-based controller consists of Braitenberg's No.3c architecture as an action selection mechanism and an artificial emotion mechanism as a superstructure over it. The reason for this choice together with the description of both the components and the functions of the controller are presented below.

4.1 Artificial emotion mechanism

Emotion is a key element of adaptive behaviour, increasing the possibility of survival of living organisms. According to J. LeDoux (1996) the *amygdala*, being deep in the brain's center, makes the memory emotional. It is responsible for the emotions, especially for the most fundamental one - the fear. Sensor information obtained by receptors is firstly gathered at *thalamus*, and then forks into cerebral *cortex* and amygdala. In cerebral cortex,

fined-grained information processing is carried out. The signals coming from the thalamus are fast and crude, reaching the amygdala before signals from the cortex, but providing only general information about the incoming stimulus. The signals from the cortex are slow and refined, providing detailed information about the stimulus. The coarse information processing accomplished in amygdala just evaluates whether the current situation is pleasant or not. It requires less computing time compared with the one in the cortex. This coarse but fast computation in the emotional system is indispensable for self preservation since living organisms have to cope with a continually changing world. The pathways that connect the amygdala with the cortex ("the thinking brain") are not symmetrical - the connections from the cortex to the amygdala are considerably weaker than those from the amygdala to the cortex. The amygdala is in a much better position to influence the cortex. Due to the above characteristics, it can be considered that the emotional system regulates activities in the cerebral cortex feedforwardly (Mochida et al., 1995).
In the computational model of the amygdala proposed by Mochida et al. (1995), the emotion of robots is divided into two states: *pleasantness* and *unpleasantness*, represented by a state variable called *frustration*. The neural network representation of this model is shown in Fig.3. Using sensory inputs the level of frustration is formulated as (Mochida et al., 1995):

$$f_{k+1} = \xi_1 \sum_{i=1}^{n} W_i S_i + (1 + \xi_2 \sum_{i=1}^{n} W_i S_i - b) f_k \ , \tag{13}$$

where f_k represents the frustration level of the agent at the moment k, ξ_1 and ξ_2 are coefficients, W_i denotes the weight parameter with respect to the obstacle detector S_i, and b is the threshold, which determines the patience for unpleasantness; n is the number of equipped obstacle detectors; S_i, $i = 1,2,...,5$ is the continuous signal from the obstacle detector, which is presented in Fig. 4, and represents the level of danger derived from the distance between the agent and the detected obstacle; In (13), the first and the second terms on the right hand side denote the frustration levels caused by the direct stimulation of the agent and the recently experienced relationship between the agent and the situation, respectively. The regulation output $\gamma = \gamma(f)$ of the emotional mechanism is determined here as:

$$\gamma = -\frac{1 - e^{-2(f + b_\gamma)}}{1 + e^{-2(f + b_\gamma)}} \ , \tag{14}$$

where $f \leftarrow f_k$ is taken from (13), b_γ is a bias, and $\gamma \in [-1; 1]$ (Fig.3).

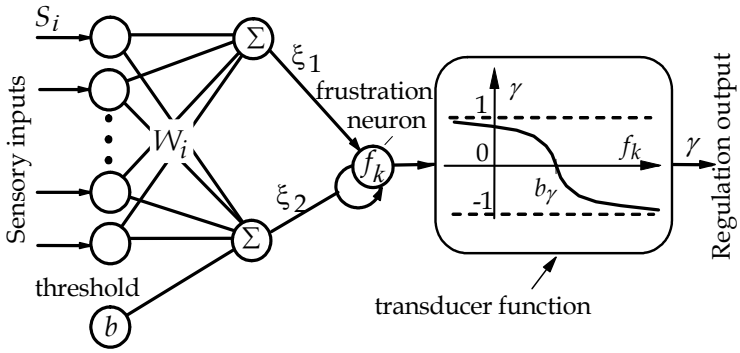
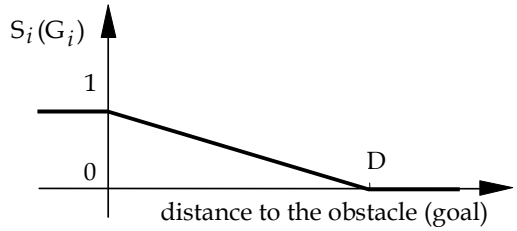


Fig. 3. Model of amygdala (Mochida et al., 1995)



D is**:** the diameter of the robot (obstacle detector) or
the diagonal of the rectangular working field (goal detector)

Fig. 4. Normalized sensor readings of the obstacle/goal detectors

4.2 Emotionally influenced Braitenberg's vehicle No.3c

Before incorporating the artificial emotion mechanism into the local sensor-based controller, the innate action selection mechanism has to be determined. It is realized in the form of Braitenberg's architecture No.3c (Braitenberg, 1984) containing two neural networks, NN1 and NN2, for obstacle avoidance and goal following behaviours, respectively. The goal, as it has been mentioned in Section 2.1, is a virtual target, which corresponds to the current reference trajectory point. These networks directly connect sensors with the motor neurons of each wheel as shown in Fig. 5. Each of the connections has a positive (excitatory) or negative (inhibitory) weight. Depending on the weights of the connections, these structures can display repulsive (Fig. 5a) or attractive (Fig. 5b) behaviours. The robot motor control is determined by simple summation of the output signals of the corresponding motor neurons. The two neural networks, NN1 and NN2, simultaneously take part in a control action, that represents their consensus (Fig. 6a). In Fig. 6a S and G are sensor readings of the obstacle and goal detectors, respectively (as it has been stated in Fig. 4 and in Section 2.2); $\mathbf{v}_S = (v_{S,right}, v_{S,left})^{\mathrm{T}}$ and $\mathbf{v}_G = (v_{G,right}, v_{G,left})^{\mathrm{T}}$ are vectors with target linear velocities towards the motors of the right and left wheels in the obstacle avoidance and goal following networks, respectively. Each wheel is assumed to rotate according to the total activation level of the corresponding motor neuron.

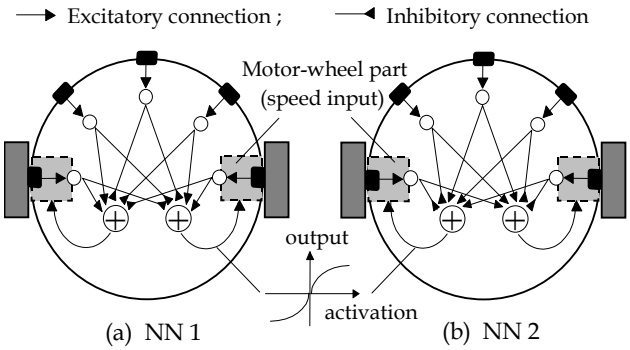


Fig. 5. Neural network architecture for: (a) obstacle avoidance behaviour - and (b) goal following behaviour (Tsankova & Topalov, 1999)

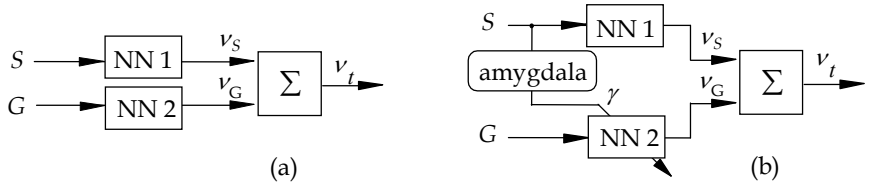


Fig. 6. Braitenberg's architecture No.3c: (a) independently acting, and (b) emotionally influenced

To incorporate the artificial amygdala into Braitenberg's architecture No.3c the outputs of the motor neurons in the goal following network have to be generated by multiplying together the weight of the network connections and the obtained regulation output γ (Fig.6b). The artificial emotion (fear) mechanism influences on NN2 feed-forwardly by modulating the weights of connections with γ, taking positive and negative values. Thus, if obstacles are available, especially if they are critically disposed at the agent's movement to the goal, the goal following behaviour is manipulated: from decreasing the influence of the goal, through forgeting it, to the appearance of a mirror (false) goal, instead of the real one. In case of absence of obstacles the artificial amygdala does not influence on the action selection mechanism, because the weights of NN2 are multiplied by $\gamma = 1$.

5. The Adaptive Trajectory Tracking Controller

5.1 The trajectory tracking controller – task, scheme, and control laws

The trajectory tracking problem for non-holonomic vehicles is posed as follows (Fierro and Lewis, 1995): Given a reference cart

$$\dot{x}_r = \upsilon_r \cos\theta_r, \quad \dot{y}_r = \upsilon_r \sin\theta_r, \quad \dot{\theta}_r = \omega_r, \quad q_r = [x_r \quad y_r \quad \theta_r]^T, \quad \nu_r = [\upsilon_r \quad \omega_r]^T, \qquad (15)$$

with $\upsilon_r > 0$ for all t, find a smooth velocity control υ_t such that $\lim_{t\to\infty}(q_r - q) = 0$, and compute the torque input $\tau(t)$ for (2) by some means such that $\nu \to \nu_t$ as $t \to \infty$.

The proposed structure for the mobile robot's tracking control system is presented in Fig. 7. It is assumed that the solution to steering system tracking problem as found in Kanayama et al.(1990) is available. This is denoted as ν_t. The tracking error posture $\mathbf{e}(t) = (e_1(t), e_2(t), e_3(t))^{\mathrm{T}}$ is expressed in the basis of a frame linked to the mobile platform (Fierro and Lewis, 1995)

$$\mathbf{e} = \mathbf{T}_e(\mathbf{q}_r - \mathbf{q}),$$

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}. \tag{16}$$

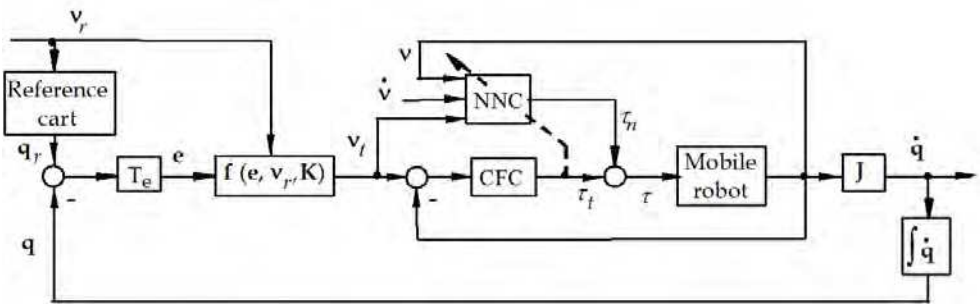


Fig. 7. Block diagram of neural network based tracking control of mobile robot.

The auxiliary velocity control input (target velocity vector $\mathbf{v}_t(t)$) that achieves tracking for (1) is given by (Kanayama et al., 1990):

$$\mathbf{v}_t = \mathbf{f}(\mathbf{e}, \mathbf{v}_r, \mathbf{K}) = \begin{bmatrix} \upsilon_r \cos e_3 + k_1 e_1 \\ \omega_r + k_2 \upsilon_r e_2 + k_3 \upsilon_r \sin e_3 \end{bmatrix}, \tag{17}$$

where k_1, k_2 and k_3 are positive constants, and $\upsilon_r > 0$.

Given the desired velocity $\mathbf{v}_t(t) \in R^{n-m}$, the auxiliary velocity tracking error can be defined as

$$\mathbf{e}_t = \mathbf{v}_t - \mathbf{v}. \tag{18}$$

In the proposed velocity control scheme, no preliminary knowledge of the cart dynamics is assumed. The velocity control input vector $\mathbf{v}_t$ is transformed into desired left and right wheel velocities and compared to the current wheel velocities of the mobile robot. Each wheel velocity is under independent control. In accordance with the *feedback-error-learning*

approach (Gomi & Kawato, 1990; Gomi & Kawato, 1993), a conventional PD feedback controller (CFC) is used both as an ordinary feedback controller to guarantee global asymptotic stability during the learning period, and as a reference model for the responses of the controlled object. To obtain the desired response during tracking-error convergence movement by compensating for the nonlinear object dynamics, the RBF neural network is trained to become an adaptive nonlinear controller (NNC). The output of the CFC is fed to the NNC as the error signal used for learning.

There are two NNCs with identical structure for each robot's wheel velocity. The structure of an NNC is presented in Fig. 8. It has three inputs ($\upsilon^{left/right}$, $\dot{\upsilon}^{left/right}$, $\upsilon_t^{left/right}$) corresponding to the actual wheel velocity and acceleration, and the desired wheel velocity of the robot's left or right wheel, respectively. The output ($\tau_n^{left/right}$) of the network is computed using on-line RBF training procedure, described in Section 5.2. As output learning patterns the values of the signal $\tau^{left/right}$, generated with contribution of the conventional PD controller ($\tau_t^{left/right}$) (Fig. 7), are used.

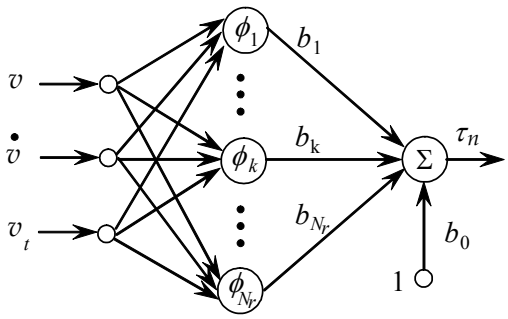


Fig. 8. RBF neural network structure. There are two networks with identical structure for each wheel – left and right. The names of the variables and parameters have to be considered with the label "left" or "right" (e.g., υ^{left} / υ^{right}), respectively.

5.2 RBF network

In this Section, the approach of Rosipal et al. (1998) to on-line constructing a resource-allocating radial basis function network is presented. It exploits weights adaptation using recursive least squares technique based on Givens QR decomposition.

Consider a two-layered RBF network, which implements a mapping $\hat{y}: R^n \to R$ according to

$$\hat{y} = b_0 + \sum_{i=1}^{N_r} b_i \phi_i \left(\left\| \mathbf{x} - \mathbf{c}_i \right\| / \sigma_i \right), \tag{19}$$

where $\mathbf{x} \in R^n$ is an input vector, $\phi_i(.)$ is the transfer function (Gaussian function), σ_i is the i-th center width, $\|.\|$ denotes the Euclidean norm, $b_i \in R$ are the weights, $\mathbf{c}_i \in R^n$ represent the positions of RBF centers, and N_r is the number of centers.

Growing phase. The network starts to learn with no centers. The condition for allocating a new center at discrete time j exploits two criteria, proposed by Platt (1991). The first criterion is based on the prediction error $|e_{RBF}(j)| = |y(j) - \hat{y}(j)|$ ($y(j)$ is the desired output), which is compared with the critical value ε. The second criterion is satisfied if the Euclidean distance of input $\mathbf{x}(j)$ from the nearest center $\mathbf{c}_{nearest}$ is greater than the critical scale resolution $\delta(j)$. The learning starts with the largest scale of resolution, i.e. $\delta(0) = \delta_{\max}$ and δ is shrunk during the time steps by a decay constant T_d until it reaches the smallest value $\delta_{\min}$. If both criteria are satisfied, a new center is set as

$$\mathbf{c}_{new}(j) = \mathbf{x}(j), \tag{20}$$

with width

$$\sigma_{new} = k_{RBF} \left\| \mathbf{x}(j) - \mathbf{c}_{nearest}(j) \right\|^2, \tag{21}$$

where k_{RBF} is a constant, and corresponding output weight

$$b_{new}(j) = e_{RBF}(j). \tag{22}$$

LMS gradient descent is used to update the positions of the centers

$$\Delta \mathbf{c}_i(j) = 2 \frac{\alpha}{\sigma_i^2} (\mathbf{x}(j) - \mathbf{c}_i(j)) \phi_i(\mathbf{x}(j)) e_{RBF}(j) b_i(j), \tag{23}$$

where $\alpha > 0$ is the learning factor.
The steps, describing the network learning strategy, are given in Table 1 (Rosipal et al., 1998; 1997).

I. Initialization: $\delta(0)=\delta_{\max}$, $b_0(0)=y(0)$, $N_r=0$, set ε

II. At each iteration j, do:

1. Present a new input-output pair $(\mathbf{x}(j), y(j))$;
2. Evaluate output of network $\hat{y}(j)=b_0(j)+\sum_{i=1}^{N_r} b_i(j)\phi_i(\mathbf{x}(j))$;
3. Compute error $e_{RBF}(j)=y(j)-\hat{y}(j)$;
4. Find distance to nearest center $d=\min_{1\le i\le N_r}\|\mathbf{c}_i(j)-\mathbf{x}(j)\|$;
5. If $|e_{RBF}(j)|>\varepsilon$ and $d>\delta(j)$,
   then allocate new center: $N_r=N_r+1$, $\mathbf{c}_{N_r}(j)=\mathbf{x}(j)$, $b_{N_r}(j)=e_{RBF}(j)$,
   $$\sigma_{N_r}=k_{RBF}d^2;$$
   Otherwise,
   update $(b_0(j), \{b_i(j), \mathbf{c}_i(j)\}_{i=1}^{N_r})$.
6. If $\delta(j)>\delta_{\min}$,
   then update $\delta(j+1)=\delta(j)\exp(-1/T_d)$;
   Otherwise,
   $\delta(j+1)=\delta(j)$.

Table 1. The RBF network learning strategy

After adding a new center, another center addition is forbidden and during the next T_W time steps only adaptation of the network parameters is allowed.

The on-line adaptation of output-layer weights $\mathbf{b}=[b_0, b_1, ..., b_{N_r}]^{\mathrm{T}}$ can be formulated as a problem of finding a weights vector $\mathbf{b}$, which minimizes a performance criterion, e.g.

$$V_t(\mathbf{b})=\sum_{j=0}^{t} w_t(j)\, e_{RBF}^2(j,\mathbf{b}(j)), \tag{24}$$

where $e_{RBF}(j,\mathbf{b}(j))$ is the error signal at time j defined as $e_{RBF}(j,\mathbf{b}(j))=y(j)-\hat{y}(j,\mathbf{b}(j))$, and $w_t(j)$ is a window function reflecting time-varying significance of past and recent error signal:

$$w_t(j)=\lambda^{t-j}, \tag{25}$$

where $0<\lambda<1$ is a forgetting factor.

From the weights adaptation point of view, the RBF network can be considered as a special case of linear regression model

$$\mathbf{y}(t)=\mathbf{\Phi}(t)\mathbf{b}(t)+\mathbf{e}(t), \tag{26}$$

where $\mathbf{y}(t)=[y(1), y(2), ..., y(t)]^{\mathrm{T}}$, $\mathbf{e}_{RBF}(t)=[e_{RBF}(1), e_{RBF}(2), ..., e_{RBF}(t)]^{\mathrm{T}}$, $\mathbf{\Phi}(t)$ is $t\times N_{r+1}$

matrix whose transpose $\mathbf{\Phi}^{\mathrm{T}}(t) = [\phi(1), \phi(2), ..., \phi(t)]$, in which $\phi(j) = [1, \phi_1(j), ..., \phi_{N_r}(j)]^{\mathrm{T}}$ is a $(N_{r+1}+1)\times 1$ vector of the hidden-layer outputs (and bias term) at time j.

In accordance with the criterion (24), weights $\mathbf{b}(t)$ can be determined by a recursive least squares technique based on Givens QR decomposition. The estimation of the vector $\mathbf{b}(t)$ is based on the solution of the following system:

$$\mathbf{\Theta}(t)\hat{\mathbf{b}}(t) = \mathbf{q}(t), \tag{27}$$

where $\mathbf{\Theta}(t)$ is an upper triangular square matrix and $\mathbf{q}(t)$ is a vector. The solution of (27) consists of the steps presented in Table 2 (Rosipal et al., 1998). In Table 2 when a new center is allocated at time j, the forgetting factor $\lambda(j)$ is re-initialized to $\lambda(0)$.

Pruning phase. The above algorithm does not eliminate centers whose contribution to performance accuracy of the network is becoming negligible. Therefore, the updated network has to be checked to prune the redundant hidden units that have minimal contribution to the network output for M consecutive observations in the sliding data window. In this work the pruning strategy of Salahshoor and Jafari (2007) is used. For this purpose, the contributions of all the available hidden units ($i = 1,..., N_r$) are first calculated on the network output as follows:

$$o_i(\mathbf{x}) = b_i\phi_i(\mathbf{x}), \qquad i = 1,..., N_r. \tag{28}$$

The highest absolute value of these contribution values is:

$$o_{\max}(\mathbf{x}) = \max_i |\, o_i(\mathbf{x})\,|. \tag{29}$$

The contribution values on the network output are normalized to the highest absolute value:

$$r_i = |\, o_i / o_{\max}\,|, \qquad i = 1,..., N_r. \tag{30}$$

If the normalized contribution value of the i-th hidden unit on the network output (r_i) falls below a threshold ($r_{\text{threshold}}$) for M consecutive observations, the i-th hidden unit is regarded as a redundant unit and is pruned (Salahshoor & Jafari, 2007).

1. Initialize matrix $\mathbf{\Theta}(.)$ and vector $\mathbf{q}(.)$
   - At time $k = 0$ set $\mathbf{\Theta}(0) = \eta$ and $\mathbf{q}(0) = 0$;
   - If a new center N_r is allocated at time $k \neq 0$, increase dimensions of $\mathbf{\Theta}(k)$ and $\mathbf{q}(k)$ to $(N_r + 1) \times (N_r + 1)$ and $(N_r + 1)$, respectively, and set diagonal elements $\mathbf{\Theta}_{N_r+1,N_r+1}(k) = \eta / e_{RBF}(k)$ and $\mathbf{q}_{N_r+1}(k) = \eta$ (where η is a small positive number);

2. At each time $j \neq k$, update $\lambda(j)$ via $\lambda(j) = \lambda_0 \lambda(j-1) + 1 - \lambda_0$, the appropriate values for λ_0 and $\lambda(0)$ are just less than one;

3. At time $j \neq k$, form the following $(N_r + 2) \times (N_r + 2)$ matrix;

$$\Omega(j) = \left[\begin{array}{c|c} \lambda^{1/2}\mathbf{\Theta}(j-1) & \lambda^{1/2}\mathbf{q}(j-1) \\ \hline \phi^{\mathrm{T}}(j) & y(j) \end{array}\right]$$

4. At time $j \neq k$, perform a sequence of elementary Givens rotations (GR) to transform the matrix $\Omega(j)$ to an upper triangular matrix;

$$\Omega(j) = \left[\begin{array}{c|c} \lambda^{1/2}\mathbf{\Theta}(j-1) & \lambda^{1/2}\mathbf{q}(j-1) \\ \hline \phi^{\mathrm{T}}(j) & y(j) \end{array}\right] \overset{\mathrm{GR}}{\Rightarrow} \left[\begin{array}{c|c} \mathbf{\Theta}(j) & \mathbf{q}(j) \\ \hline 0 & * \end{array}\right],$$

where 0 is $1 \times (N_r + 1)$ zeros vector and $*$ is a *don't care* variable;

5. At time $j \neq k$, compute $\hat{\mathbf{b}}(j) = \mathbf{\Theta}^{-1}(j)\mathbf{q}(j)$.

Table 2. The recursive technique for determining $\mathbf{b}(t)$

5.3 The trajectory tracking controller in collaboration with the local obstacle avoidance one

The overall integrated system for navigation and control of the mobile robot is shown in Fig.9. The path planner generates a path as a sequence of points (piece-wise linear path). Under assumption for a rectilinear movement with a constant reference velocity υ_r between each two points, discrete time step T_0 and the sequence of path points, the box $\mathrm{T_q}$ calculates the reference posture $\mathbf{q}_r$. When sensors detect an obstacle, the local obstacle avoidance controller switches on. The NN1 and the amygdala obtain inputs from obstacle detectors $\mathbf{S}$, and the NN2 – from a goal detector $\mathbf{G}$. Here, the goal detector is a virtual detector that uses the current reference point (x_r, y_r) seen with respect to the mobile base as a virtual goal. The box $\mathrm{T_G}$ derives the first two components of the vector $\mathbf{e}$ ($\mathrm{T_G}$: $\mathbf{G} \subset \mathbf{e}$) and transforms them into a shape, appropriate for the NN2 inputs. In the presence of obstacles the local obstacle avoidance controller replaces the kinematic controller, proposed by Kanayama et al. (1995) and generates the target velocities $\mathbf{v}_t$.

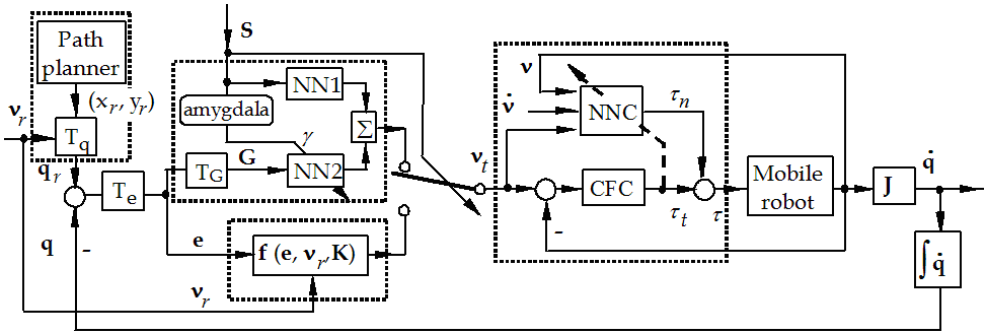


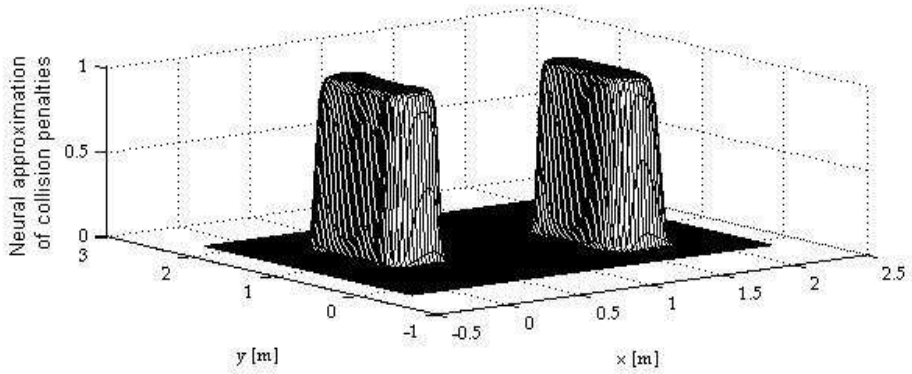
Fig. 9. Block diagram of the overall integrated system for navigation and control of mobile robot.

6. Simulation Results and Discussions

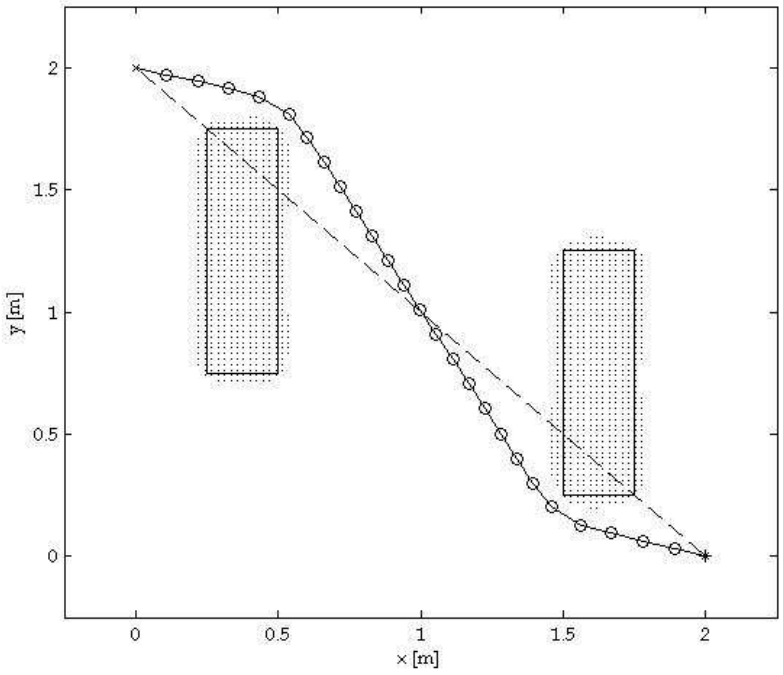
To illustrate the performance of the proposed system integrating the three control devices, they (and the overall system) were simulated in MATLAB environment and tested on several examples. In accordance with Fig. 2 the following parameters of the geometric model were chosen: $R = 0.036$ m, $d = 0.02$ m. The mass of the mobile robot was set to be $m = 0.5$ kg. All the rest parameters used in simulations are presented in the corresponding sections below, where they are at first hand employed.

6.1 Path-planning in a previously known environment

The workspace consisted of two a priori known obstacles. The neural network of the path planner was trained with the error backpropagation algorithm with momentum and adaptive learning rate. The number of neurons in the hidden layer was 25. The workspace was divided into 40×40 grid cells and the neural network used 1600 training samples. After training the neural network was used to produce collision penalties needed to solve the equations (12) and thus to find a path described by a set of $N = 25$ via points (without initial and target points). The virtual sampling time was $T_0 = 0.02$ s. The initial position was $(0, 2)$, and the goal position was $(2, 0)$. The initial path, needed for solution of the equations (12) was chosen as a straight line from the initial position to the goal position (it is shown in Fig.10b as a dashed line). The weight coefficients in (12) were chosen to be $k_L = 1$ and $k_C = 1$. Fig. 10 shows the approximation performance of the neural network ((a) the surface; (b) the dotted area). In the same figure, case (b), the piece-wise liner path obtained by the described above algorithm is presented by the bold line with 'o'-marks.



(a)



(b)

Fig. 10. Results from the path-planning procedure: (a) Surface of collision penalties generated by a neural network, trained to recognize a priori known obstacles; (b) Collision-free path obtained by the path-planning algorithm.

6.2 Performance of the sensor-based local obstacle avoidance controller

In accordance with the neural network structure depicted in Fig.5, a matrix of weights for obstacle avoidance behaviour is shown in Table 3 (Tsankova & Topalov, 1999). To give the robot a basic motivation to go forward when there are no obstacles, a bias was added to the tan-sigmoid threshold function of the motor speed neurons. The matrix of weights for goal following behaviour is shown in Table 4. The biases of the two output neurons were chosen with different signs one compared to another and much smaller than those in Table 3. These biases make the robot turn when the goal is not detected by the sensor in sectors $S_i, i = 1,2,...,5$, i.e., it is behind the robot.

Sensor:	S1	S2	S3	S4	S5	Bias
Motor Left:	0.2	0.3	-0.4	-0.72	-0.33	0.3
Motor Right:	-0.38	-0.77	-0.4	0.3	0.2	0.3

Table 3. Matrix of weights for neural network NN 1

Sensor:	S1	S2	S3	S4	S5	Bias
Motor Left:	-0.2	-0.3	0.35	0.72	0.33	0.1
Motor Right:	0.38	0.77	0.35	-0.32	-0.2	-0.1

Table 4. Matrix of weights for neural network NN 2

The Braitenberg's vehicle No.3c, equipped with the neural networks described above, was simulated in MATLAB environment and tested on a few examples. The results are shown in Fig.11, where the symbols $*$, $\times$ and $\circ$ denote the goal position, the initial and final position of the agent, respectively. The robot orientation with respect to the inertial basis is signed as θ. For the sake of simplicity (of simulations) the obstacles were chosen to be with circular shape. The obstacle avoidance behaviour derived by NN1 (Fig.5a) working alone is illustrated in Fig.11a, and the goal following behaviour obtained by NN2 (Fig.5b) operating unassisted – in Fig.11b. When the two NNs co-operated by simple summation of their outputs (Fig.6a) the resultant behaviour was a collision-free goal following behaviour (Fig.11c). It was evident that the performance of the Braitenberg's vehicle No.3c was not very good, as in some cases the robot did not reach the goal.

In accordance with the choice made in Section 4, the performance of this Braitenberg architecture was improved by emotional intervention on it. The parameters of the amygdala's model, selected as appropriate in Ref. (Tsankova, 2009), were used in this work, i.e.: $\xi_1 = 0.003$, $\xi_2 = 0.3$, $b = 0.12$, $b_\gamma = 5$ and $\mathbf{W} = (0.5 \quad 0.75 \quad 2.5 \quad 0.75 \quad 0.5)^T$. The frustration signal was limited in accordance with its definition domain $f \in [f_{\min}; f_{\max}]$, where $f_{\min} = 0$ and $f_{\max} = 18$. The simulation results with the emotionally affected Braitenberg's vehicle No.3c are shown in Fig.11d. It can be seen that the artificial amygdala assists the innate controller to negotiate some dead-end situations successfully.

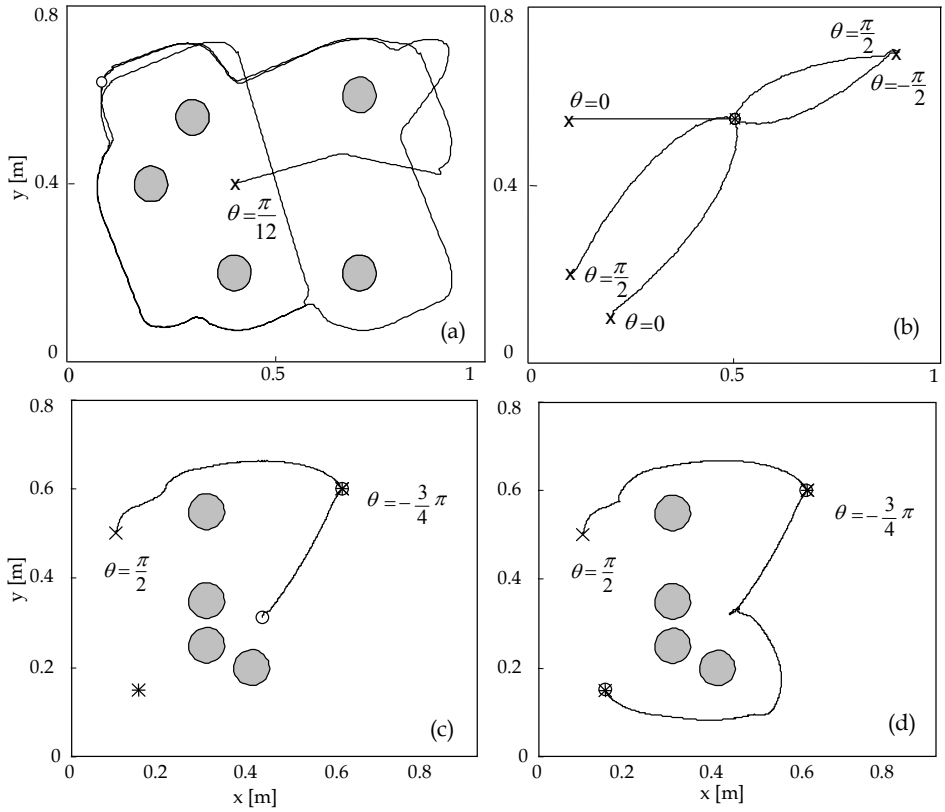


Fig. 11. Simulations with Braitenberg's vehicle No.3c emotionally affected and unaffected: (a) NN1 alone - obstacle avoidance behaviour; (b) NN2 alone – goal following behaviour; (c) NN1+NN2 together – collision-free goal following behaviour; and (d) emotionally affected vehicle.

6.3 The tracking performance and adaptability of the on-line feedback-error-learning controller

The trajectory tracking controller takes into account the operational issues of the previous two devices – the path planner and local obstacle avoidance controller. In order to handle non-smooth reference paths without causing the robot's slippage, the target velocities (υ_t, ω_t) were limited by constants $(\hat{\upsilon}_t, \hat{\omega}_t)$. The values of these constants were adopted to be $\hat{\upsilon}_t = \upsilon_{\max} = 0.4$ m/s and $\hat{\omega}_t = \omega_{\max} = 1.57$ rad/s. The posture controller gains were chosen to be as in Ref. (Kanayama et al., 1995): $k_1 = 10/\text{s}$, $k_2 = 64/\text{m}^2$ and $k_3 = 16/\text{m}$. The robot's sampling time was set to $T_0 = 0.005$ s. The following parameters for the PD controller were used: $k_\text{P} = 1$ and $k_\text{D} = 0.5$. The reference linear velocity was set to $\upsilon_r = 0.25$ m/s, and reference angular velocity – to $\omega_r = 0$ rad/s. In accordance with Section 5.2 the following

RBF network parameters were used: $\delta_{\max} = 0.1$, $\delta_{\min} = 0.005$, $\lambda(0) = 0.9$, $\lambda_0 = 0.99$, $\varepsilon = 0.005$, $\eta = 5.10^{-5}$, $\alpha = 0.001$, $k_{RBF} = 7$, $T_d = 2$, $T_W = 10$, $M = 10$, $r_{\text{threshold}} = 0.1$.

Fig. 12a shows simulation results for $\Delta\theta$ discontinuous jumps ($\Delta\theta = \pi/4$, $\pi/2$, and $\Delta\theta = 3\pi/4$) carried out without the velocity limiter. The actual trajectories are marked with solid lines, while the reference trajectories - with dashed lines. Fig. 12b shows simulation results for the same directional discontinuities with the velocity limitation. The mobile robot tracked the reference trajectory, but its responses under limitation of velocities were slower than those without a limitation.

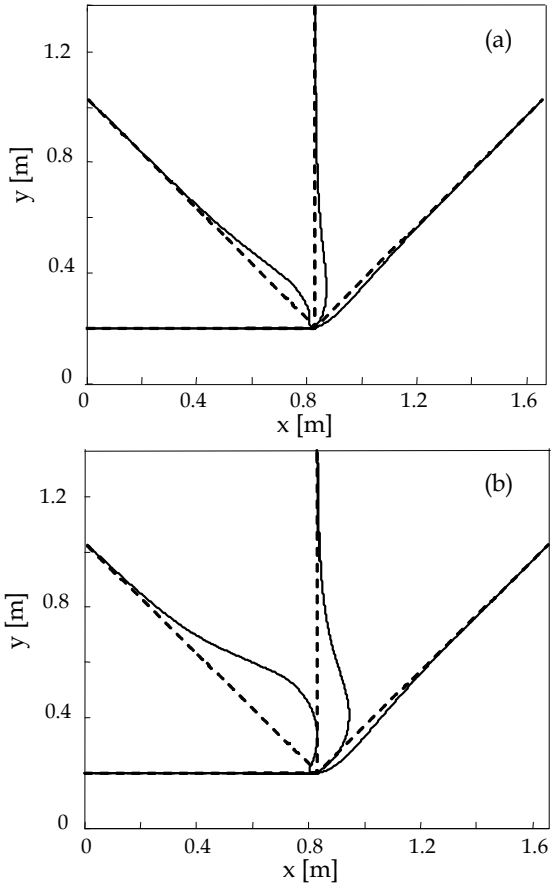


Fig. 12. Tracking under directional discontinuities: (a) without the limiter; (b) with the limiter

In order to demonstrate the adaptation capability of the control scheme to execute desired motions in the face of unforeseen parameter variations, an example when the robot picked up a load during the trajectory tracking task was carried out. During tracking a reference trajectory the mobile robot has to pick up an unknown load, which is considered as an

increase in the robot's mass from 0.5 kg to 0.75 kg. The performance of the considered control scheme under these conditions is depicted in Fig.13, where the actual trajectory is marked with a solid line while the reference trajectory - with a dashed line. The position where the robot undergoes change of its mass is marked with a black point. The tracking performance of the mobile robot before and after the change of mass is almost the same. This is due to the adaptive capabilities of the tracking controller using RBF network with on-line feedback-error-learning.

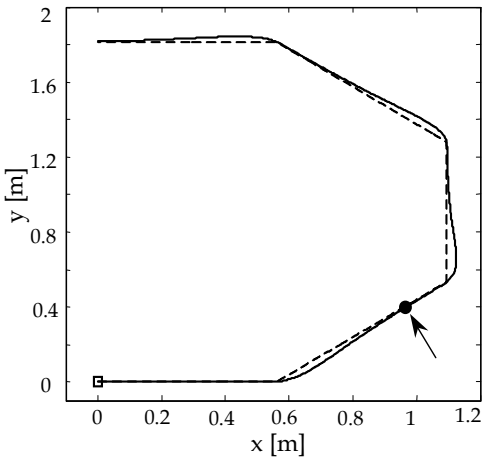


Fig. 13. Trajectory tracking with change of the robot's mass.

6.4 The robot, equipped with the overall integrated system: simulations and discussions

This Section treats the performance of the robot supplied with the overall integrated system tested by simulations in a partially known environment. Tracking the reference trajectory, the robot encountered accidentally appearing obstacles. Two types of obstacles were presented in the experiments: static (non-movable) and dynamic (movable) obstacles. The static obstacles were of both small and large size. Here, it is assumed that the small size is a size, commensurable with the robot size. In the simulations the small obstacles were with circular shape and a radius $R_{\text{obst}} = 0.04$ m . The large obstacles were compositions of a certain number of identical small obstacles. In Fig. 14a it is shown how the robot escapes a small obstacle, accidentally encountered during the movement along the reference trajectory. When the sensors no longer detect the obstacle, the robot returns to the trajectory tracking behaviour. However, going round the obstacle slows down the robot and the next reference point (which is a time-indexed one) turns out to be quite remote from it. In Fig.14 a few pairs of points with equal time indexes on the current and reference trajectory are marked with signs "+" and "★", respectively, to illustrate this delay. Depending on the maximum linear velocity and the location of a priori known obstacles with respect to the actual and the reference positions of the robot after avoiding the local obstacle, the trajectory tracking task may finish successfully or not. The probability for tracking without success increases when the robot encounters obstacles with bigger sizes

(Fig.14c). This drawback could be sold by an on-line modification of the time indexes of the reference trajectory points after the detour of a local obstacle by applying the following rule:

Rule 1: If *(*there is not a detected obstacle at time k) **&** (there was an obstacle detected at time k-1) ***then*** find the reference point $(x_r^{\min}, y_r^{\min})$ nearest to the current robot position and set its time index to be k. Recalculate the time indexes of the rest of the points (to the end) of the reference trajectory and start trajectory tracking behaviour with $(x_r^{\min}, y_r^{\min})$ at time index k.

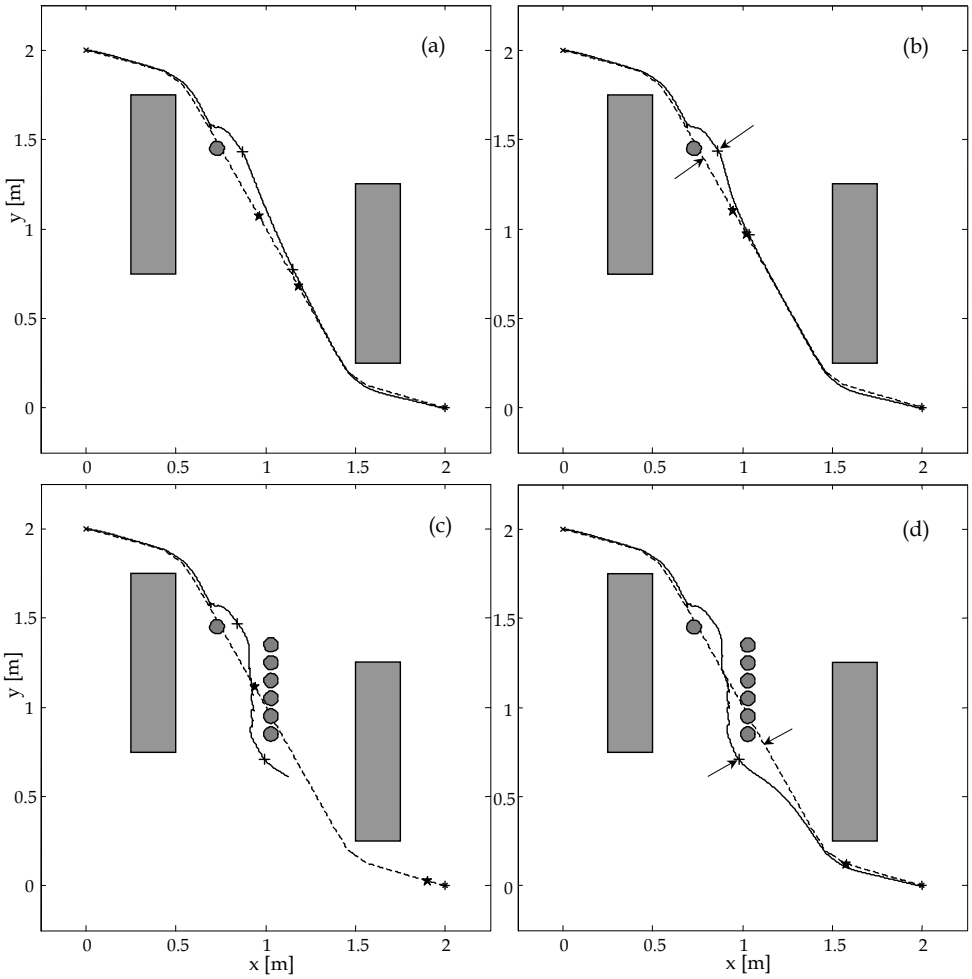


Fig. 14. Trajectory tracking and local obstacle avoidance behaviour: (a) and (c) – ordinary algorithm; (b) and (d) – modified by *Rule 1* algorithm

The contribution of this rule is demonstrated in Fig.14b and Fig.14d, where the two arrows show: (I) the current trajectory point where Rule 1 is switched on, and (II) the recalculated starting reference point $(x_r^{\min}, y_r^{\min})$ at time index k.

The proposed three-component system for a robot navigation and control copes with dynamic obstacles as well as with static ones. In Fig.15 tracking the reference trajectory, the robot successfully passes each other with two movable obstacles. They move rectilinearly with constant velocity $\upsilon_{\text{obst}} = 0.15$ m/s .

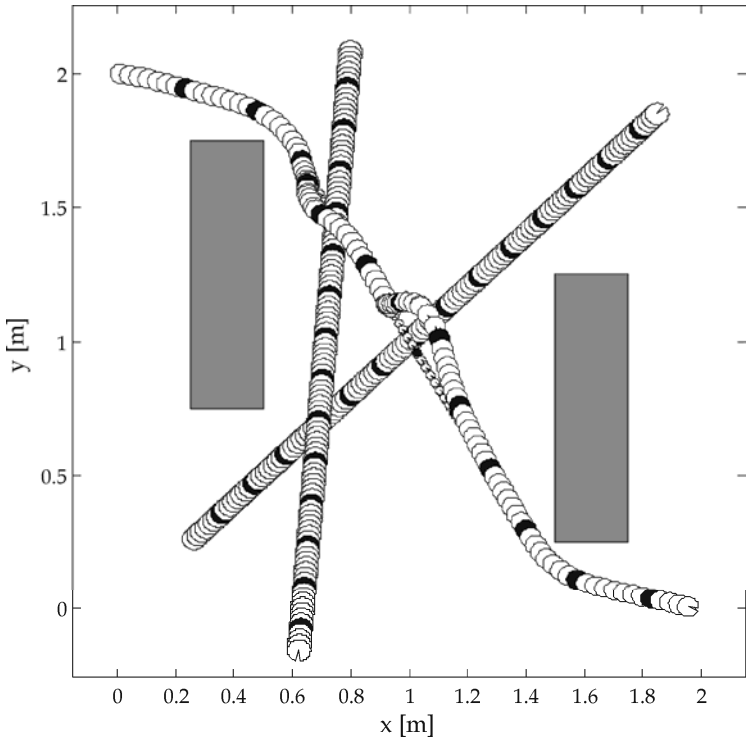


Fig. 15. Navigation and control of the robot in an environment with moving obstacles

The advantages and drawbacks of each of the devices in the integrated control system reflect on the overall performance of the system. Hence the improvement of each of them separately matters to the integrated system. The planner not only generates the collision-free path, but optimizes it in the sense of the criterion "the shortest path". However, for the sake of the gradient nature of the planning algorithm, it suffers from a significant drawback – the trapping in local minima of the potential function, which could frustrate the planning. The emotional intervention on the Braitenberg's architecture No.3c significantly improves it and makes it very proper as a sensor-based local obstacle avoidance controller. It has been reported in the literature (Tsankova, 2008; Tsankova, 2009) that the performance could be improved once more if an artificial immune network is used as innate action selection mechanism, but in compromise with the computation complexity. The trajectory tracking

controller, designed using the approach proposed by Gomi and Kawato (1993), ensures the adaptability of the robot to the uncertainties or changes in its dynamics. Here the contribution to this controller is the use of RBF NNC and especially the learning algorithm. The advantage of using weights adaptation by a recursive least squares technique based on Givens QR decomposition is a faster and easier solution of the problem than the cases, when: (1) a genetic algorithm has been used for fuzzy-neural network learning (Topalov et al., 1997); (2) a NN with backpropagation algorithm has been applied for the same purpose (Topalov et al., 1998).

7. Conclusions

A complete integrated system for navigation and control of a mobile robot in a partially known environment is developed. The neural network in the path-planning algorithm can automatically build up the collision penalty function (as an obstacles' potential function approximation). The training procedure uses the information of the environment, no matter how many obstacles there are and how they look like. This makes the algorithm effective, but because of its gradient character, it meets the local minima problem. The local obstacle avoidance controller performs very well under accidentally encountered obstacles. Probably this is due to the property of the artificial amygdala to be the short term memory for the 'fear' from the encountered obstacles. The trajectory tracking controller is highly adaptable to uncertaities and changes in the robot dynamics. The recursive least squares technique using Givens QR decomposition is a comparatively fast and easy way for training the RBF net's weights. Future work will include the development of a modification of the path-planning algorithm, which aims at filling the local minima along the path to calculate and thus to ensure reaching the goal reliably. This modification, as well as the overall integrated control system, will be further verified on a real mobile robot.

Acknowledgements

The paper presents research and development, supported by Ministry of Education and Science in Bulgaria (National Scientific Fund) under the Research Project BY-TH-108/2005.

8. References

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*, MIT Press, Cambridge, MA

Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.E. & Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, ISBN: 0262033275, Boston

Fierro, R. & Lewis, F.L. (1995). Control of a nonholonomic mobile robot: backstepping kinematics into dynamics. *Proceedings of the 34th Conference on Decision and Control*, pp. 3805-3810

Gomi, H. & Kawato, M. (1990). Learning control for a closed loop system using feedback-error-learning. *Proceedings of the 29th Conference on Decision and Control*, pp. 3289-3294

Gomi, H. & Kawato, M. (1993). Neural network control of a closed-loop system using feedback-error-learning. *Neural Networks,* Vol. 6, pp. 933-946

Kanayama, Y.; Kimura, Y.; Miyazaki, F. & Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. *Proceedings of IEEE International Conference on Robotics and Automation*, Vol.1, pp. 384-389

Latombe, J.C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers, ISBN: 0792391292, Boston

LeDoux, J. (1996). *The Emotional Brain*, Simon & Schuster, New York

Meng, H. & P.D. Picton (1992). A neural network for collision-free path-planning. In: *Artificial Neural Networks,* Aleksander, I. & Taylor, J. (Ed.), Vol.2, pp. 591-594

Mochida, T.; Ishiguro, A.; Aoki, T. & Uchikawa, Y. (1995). Behaviour arbitration for autonomous mobile robots using emotion mechanisms. *Proceedings of IROS'95*, Vol.1, pp. 516-521

Platt, C.J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, Vol.3, No.2, pp. 213-225

Rosipal, R.; Koska, M. & Farkas, I. (1997). Chaos time-series prediction using resource-allocating RBF networks. *Proceedings of Conference Measurement '97*, Smolenice, Slovakia

Rosipal R.; Koska, M. & Farkas, I. (1998). Prediction of chaotic time-series with a resource-allocating RBF network. *Neural Processing Letters*, Vol.7, No.3, pp. 185-197

Salahshoor, K. & Jafari, M.R. (2007). On-line identification of non-linear systems using adaptive RBF-based neural networks, *International Journal of Information Science and Technology*, Vol.5, No.2, pp. 100-121

Topalov, A.; Tsankova, D.; Petrov, M. & Proychev, T. (1997). Intelligent motion planning and control for a simulated mobile robot, *Proc. of the 2nd IFAC Workshop on New Trends in Design of Control Systems*, pp. 338-343, Smolenice, Slovac Republic, 1997

Topalov, A.V.; Kim, J.-H. & Proychev, T.P. (1998a). Fuzzy-net control of non-holonomic mobile robot using evolutionary feedback-error-learning, *Robotics and Autonomous Systems*, Vol.23, pp. 187-200, Elsevier

Topalov, A.; Tsankova, D.; Petrov, M. & Proychev, T. (1998b). Intelligent sensor-based navigation and control of mobile robot in a partially known environment, *Proc. of the 3rd IFAC Symposium on Intelligent Autonomous Vehicles*, pp. 439-444, Madrid, 1998

Tsankova, D. (2008). Emotional intervention on stigmergy based foraging behaviour of immune network driven mobile robots, Chapter 27, pp. 517-542, In: *Frontiers in Evolutionary Robotics*, Iba, H. (Ed.), I-tech Education and Publishing, Vienna, Austria, www.i-technoline.com, ISBN 978-3-902613-19-6

Tsankova, D.D. (2009). Emotional intervention on an action selection mechanism based on artificial immune networks for navigation of autonomous agents, In: *Adaptive Behavior*, Vol.17, No.2, pp. 135-152, June 2009, Sage Press

Tsankova, D.D. & Topalov, A.V. (1999). Behaviour arbitration for autonomous mobile robots using immune networks. *Proceedings of the 1st IFAC Workshop on Multi-Agent Systems in Production (MAS'99)*, pp. 25-30, Vienna, Austria, 2-4 December, 1999

Yahja, A.; Singh, S. & Stentz, A. (2000). An efficient on-line path planner for outdoor mobile robots. *Robotics and Autonomous Systems*, Vol.32, pp. 129-143, Elsevier Science.

Zulli, R.; Fierro, R.; Conte, G. & Lewis, F.L. (1995). Motion planning and control for non-holonomic mobile robots. *Proc. IEEE Int. Symp. on Intelligent Control,* pp. 551-557

Mobile Robots Navigation

Edited by Alejandra Barrera

ISBN 978-953-307-076-6
Hard cover, 666 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

Mobile robots navigation includes different interrelated activities: (i) perception, as obtaining and interpreting sensory information; (ii) exploration, as the strategy that guides the robot to select the next direction to go; (iii) mapping, involving the construction of a spatial representation by using the sensory information perceived; (iv) localization, as the strategy to estimate the robot position within the spatial map; (v) path planning, as the strategy to find a path towards a goal location being optimal or not; and (vi) path execution, where motor actions are determined and adapted to environmental changes. The book addresses those activities by integrating results from the research work of several authors all over the world. Research cases are documented in 32 chapters organized within 7 categories next described.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821