Int. J. Appl. Math. Comput. Sci., 2006, Vol. 16, No. 4, 475–484

A LEARNING PARADIGM FOR MOTION CONTROL OF MOBILE MANIPULATORS

FOUDIL ABDESSEMED*, ERIC MONACELLI**, KHIER BENMAHAMMED***

* UNIVERSITE de Batna
Institut d'Electronique, Rue Chahid Boukhlouf
Batna – 05000, Algeria
e-mail: fodil_a@hotmail.com

** Laboratoire de Robotique de Paris
1–12 Avenue de l'Europe, 78140 Velizy-Paris, France
e-mail: eric.monacelli@liris.uvsq.fr

*** Université de Setif, Institut d'Electronique
Setif – 19000, Algeria
e-mail: khierben@lycos.com

Motion control of a mobile manipulator is discussed. The objective is to allow the end-effector to track a given trajectory in a fixed world frame. The motion of the platform and that of the manipulator are coordinated by a neural network which is a kind of graph designed from the kinematic model of the system. A learning paradigm is used to produce the required reference variables for each of the mobile platform and the robot manipulator for an overall coordinate behavior. Simulation results are presented to show the effectiveness of the proposed scheme.

Keywords: mobile manipulator, neural network, backpropagation, obstacle avoidance

1. Introduction

In recent years, path planning and motion control of mobile manipulators have emerged as a very significant aspect in the area of autonomous robotics. However, few solutions have been provided. A mobile manipulator system is a robotic manipulator mounted on a mobile platform. This combination permits manipulation tasks in unlimited work spaces. However, since the platform and the manipulator have independent movements, a particular point in the workspace may be reached in multiple configurations, which results in a redundant system. This can be helpful when it is desirable to perform tasks in a cluttered environment, or to optimally configure the system. Research on investigating the capabilities of mobile platforms with onboard manipulators has attracted considerable attention. Dubowsky and Tanner (1988) associated a holonomic platform to a robotic manipulator. A framework is used to deal with motion planning and control to minimize a given criterion. Yamamoto and Yun (1994) considered the manipulator as being fixed during platform motion, and the platform to be anchored during manipulator motion. They proposed a control algorithm for the platform, so that the manipulator is always positioned at preferred configurations. Separate control of the end-effector and the mobile platform was also considered. This approach, proposed by Ficher *et al.* (1996), is based on the natural separation of the two motion subsystems, using a fuzzy based locomotion control strategy, including criteria evaluating the desired system configurations. The problem of controlling simultaneous motions of the mobile platform and the robot manipulator has been the area of research of several authors. Pin *et al.* (1996a) proposed the FSP method to optimally solve the inverse kinematics problem for redundant systems in the presence of applied constraints and a behavioral criterion. Lee and Cho (1997) proposed a motion planning method to execute a sequence of tasks. They formulated the motion-planning problem as a global optimization one, and simultaneously obtained the motion trajectory set and commutation configurations. Zhao *et al.* (1997) investigated simultaneous motions of a mobile platform and a manipulator. They developed a genetic algorithm to solve the optimal sequence of mobile platform positions and manipulator configurations given a series of task specifications. Chen and Zalzala (1997) proposed a genetic algorithm for multi-criteria motion planning of mobile manipulator systems. They took account of dynamics and nonholonomic constraints. Seradji (1995) obtained the required mobile platform and robot manipulator motions by solv-

ing a set of differential kinematic equations resulting from the combination of the nonholonomic platform constraint, the desired end-effector motion, and additional constraints specified by the user. Based on space formulation, Khatib (Khatib, 1993; Khatib *et al.*, 1996) developed a model for a mobile manipulator viewed as a combination of two subsystems. The mobile platform was considered as a macromechanism with a coarse, slow dynamic response, and the manipulator was treated as a fast and accurate minidevice. Their approach is reduced mainly to controlling redundant systems by obtaining the end-effector dynamic model by projecting mechanism dynamics into the operational space, and a dynamically consistent force/torque relationship that provides decoupled control of motions in the null space associated with the redundant mechanism. Nassal (1996) focused on the motion of mobile two-arm systems. The manipulator controller uses the desired end-effector position and the current platform position to compute the frame of the end effector with respect to the robot base. This frame is mapped to a vector of joint angles that are evaluated by the mobile platform, which uses a cost function for that purpose. The cost function is mapped to a gradient that serves as a vector error signal to the coordinated motion controller. But, when focusing on the different research works, the schemes proposed hitherto could be classified linto global and local approaches. This depends on whether we want to compute the optimal configuration with respect to the whole path, see, e.g., (Pin *et al.*, 1996b) or with respect to the transition of tasks, see, e.g., (Nagatani, 1996). In contrast to the classical methods (Abdessemed, 2004; Nakamura, 1991; Renaud and Dauchez, 1999), a pseudoinverse algorithm could be used to achieve the primary task of the end-effector trajectory while the redundant degrees of freedom are used to attempt to satisfy any additional subtasks, such as avoiding obstacles or preventing the saturation of the joints.

In this paper, the motions of the platform and the manipulator are coordinated by a neural network. This neural network is an adaptive graph of operations that provides reference output values of the desired motion to the mobile manipulator system. The main contribution here is the organization of the kinematic model of the mobile manipulator into a neural network in order to apply the Levenberg-Marquardt backpropagation learning rule to generate the appropriate parameter-weighting vector. Once determined, this vector is used to compute inputs to the mobile platform and to the manipulator so that the end-effector trajectory, specified in a fixed world frame, is tracked with a minimum error.

Now, what happens if an object is found in the surroundings or in the path of the end point trajectory? This is considered as an obstacle to be avoided and we can find some approaches that deal with this subject (Brock and Khatib, 2000; Brock *et al.*, 2002). In order to cope with this situation, in this paper we propose a method to solve this problem by adding a term that accounts for this new situation and which is switched each time the system detects an obstacle in the path way of the robot end point. The control algorithm is tested and the simulation results show the efficiency of the proposed approach.

This paper is organized into six sections. In Section 2, the mobile manipulator description is presented. Section 3 deals with the design procedure of the coordinated and manipulation motions, while Section 4 details the learning rule. In Section 5, an obstacle avoidance method is presented. Simulation results are presented in Section 6. Section 7 summarizes and concludes the paper.

2. Analysis of the Mobile Manipulator System

In this section we analyze the mechanical system made up of a nonholonomic platform upon which a robot manipulator with three rotational degrees of freedom is mounted as shown in Fig. 1. In this section we briefly discuss its kinematic model.

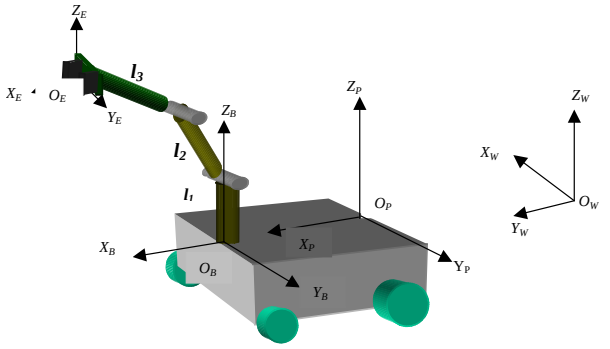


Fig. 1. Mobile manipulator configuration.

Consider Fig. 1, where four principal coordinate frames are shown: the world frame $\boldsymbol{O}_W$, the platform frame $\boldsymbol{O}_P$, the manipulator base frame $\boldsymbol{O}_B$, and the end-effector frame $\boldsymbol{O}_E$. Then the manipulator's end-effector position/orientation with respect to $\boldsymbol{O}_W$ is given by

$$T_E^W = T_P^W T_B^P T_E^B,$$

such that the matrix T_P^W is determined by some matrix $A(q)$, T_B^P is a fixed matrix and T_E^B is determined by the joint variable vector

$$\theta = \begin{pmatrix} \theta_1, & \theta_2, & \ldots, & \theta_{n_m} \end{pmatrix}^T,$$

and n_m represents the degree of freedom of the arm manipulator. The position of the end-effector x_E^W is a nonlinear function of the configuration vector $q = (p^T, \theta^T)^T \in \mathbb{R}^n$, where $n = 3 + n_m$. The joint coordinates of the manipulator are $\theta = (\theta_1, \theta_2, \theta_3)^T$ (thus $nm = 3$). Therefore,

the generalized coordinates of the mechanical system are

$$q = (q_{1,}, q_2, \ldots, q_6)^T = (x_B, y_B, z_B, \theta_1, \theta_2, \theta_3)^T.$$

Hence, the generalized space dimension of the mechanical system is equal to $\lambda = 6$. Now, for a given mechanical configuration system $\boldsymbol{q}$, its structure imposes η position and orientation constraints on its end-effector. In our case, only the end-effector position is considered. Therefore, the number of constraints is reduced to $\sigma = 3$. On the other hand, we can observe that the system is nonholonomic, and taking into account the constraint of the nonholonomy of the mobile platform, we can deduce the order of redundancy, which is equal to $(\lambda - \sigma - 1) = 2$. This redundancy helps increasing manipulator dexterity, prevents the arm from singular configurations, and keeps the system away from obstacles while completing a given task. On the other hand, the control of such mechanisms becomes much harder.

If we refer to Fig. 1 and follow the D–H parameterization, the outputs of the neural network are given by the following set of equations:

$$\begin{aligned} x_E^W &= x_B^W + \cos(\theta)\Big[l_2\cos(\theta_2) + l_3\cos(\theta_2+\theta_3)\Big], \\ y_E^W &= y_B^W + \sin(\theta)\Big[l_2\cos(\theta_2) + l_3\cos(\theta_2+\theta_3)\Big], \\ z_E^W &= z_B^W + l_1 - l_2\sin(\theta_2) - l_3\sin(\theta_2+\theta_3), \end{aligned} \tag{1}$$

where

$$\theta = \theta_1 + \varphi, \tag{2}$$

ϕ is the heading angle of the mobile platform, and l_1, l_2 and l_3 are the lengths of the three links forming the manipulator arm. Here $x_B^W,\ y_B^W \text{and} z_B^W$ are the coordinates of the point $\boldsymbol{B}$ located at the front of the mobile platform with respect to the world frame $\{\boldsymbol{W}\}$. The system (1) defines the Cartesian coordinates of the task variable E, with respect to the world frame $\{\boldsymbol{W}\}$. In a closed form, this can be written as $\boldsymbol{X}_E(t) = \boldsymbol{F}(\boldsymbol{q}(t))$, where $\boldsymbol{F}$ represents the direct kinematic mapping from the joint space to the task space and $\boldsymbol{X}_E(t) = (x_E^W,\ y_E^W, z_E^W)^T$. In the sequel, for simplicity, we assume that z_B^W equals zero. The goal is to find the generalized trajectory $\boldsymbol{q}(t)$ for a given task space trajectory $\boldsymbol{X}_E(t)$ such that $\boldsymbol{F}(\boldsymbol{q}(t)) = \boldsymbol{X}_E(t)$ is satisfied. Since the system is redundant, the number of solutions is infinite. To realize a generalized task of the mechanical system, one has to derive a set of λ generalized coordinates. In this context, an approach is suggested to investigate and solve this problem when we make a complete motion of the end-effector resulting from a combined operation of the two subsystems working in a coordinated manner.

3. Motion Control of a Mobile Manipulator

The control strategy combines mobile platform behavior and manipulator behavior to produce an integrated system that performs a coordinated motion and manipulation. If we refer to the arm manipulator as *agent1* and the mobile platform as *agent2*, then the architecture shown in Fig. 2 illustrates the actions on the environment performed by the two agents.

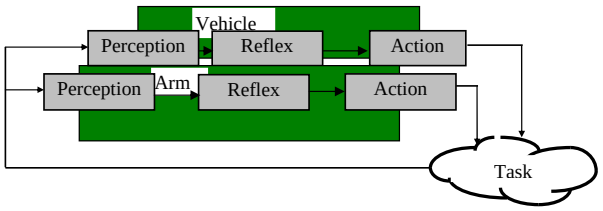


Fig. 2. Configuration of a coordinated motion and manipulation of the robotic system architecture.

To provide a solution to the mobile manipulation motion, we have arranged the direct geometric model equations into a sort of adaptive graph of operation made up of three layers (Fig. 3), which forms a neural network. Each layer has a number of nodes and each node represents a function of two or more variables. The expression of these functions is defined by (3). This neural network, being the kernel of our proposal, is very interesting and uncommon in robot trajectory generation. This setting will facilitate the implementation of the back propagation algorithm as a learning rule to adapt the weights so that the output values of the neural network come close to the desired reference values describing the task space trajectory. In this case,

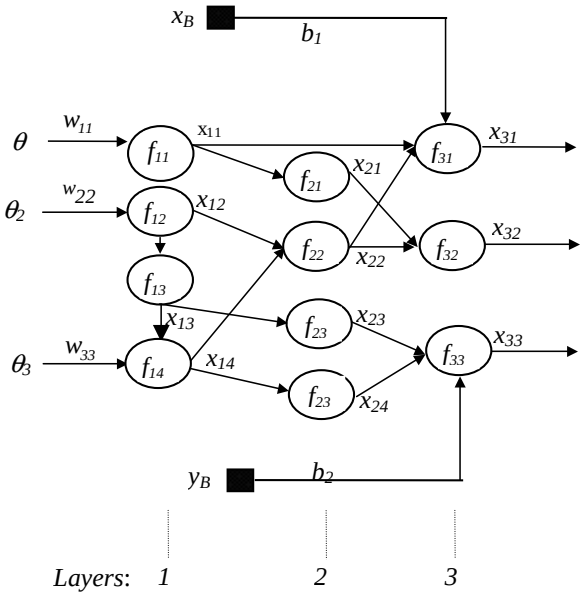


Fig. 3. Neural network of operations.

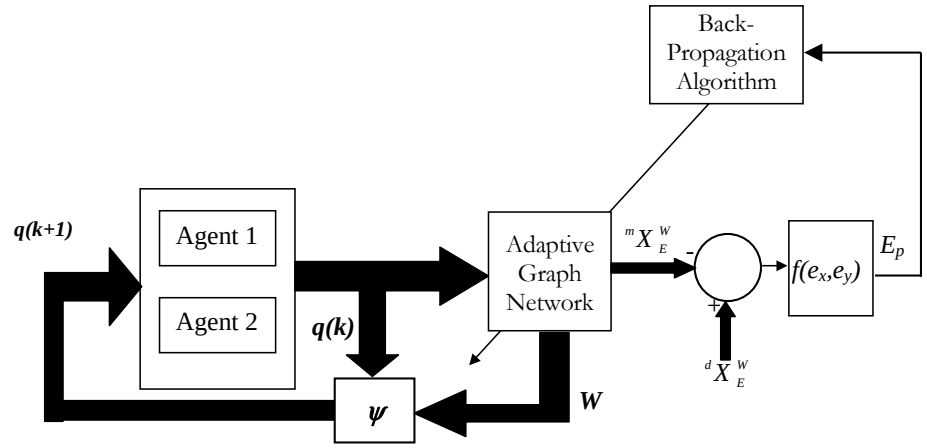


Fig. 4. Configuration of the controlled system including a neural network.

two mechanical structures are considered as a unique entity. The accomplishment of the task is the result of the permanent movement of the two structures, for which the success is based on satisfying of some criterion. Figure 4 illustrates the model architecture of the combined structure.

The functions f_{ij} are defined as follows:

$$\begin{aligned}
f_{11}(\theta, w_{11}) &= x_{11} = \cos(w_{11}\theta),\\
f_{12}(q_{m2}, w_{22}) &= x_{12} = \cos(w_{22}q_{m2}),\\
f_{13}(x_{12}) &= x_{13} = \cos^{-1}(x_{12}),\\
f_{14}(x_{13}, q_{m3}, w_{33}) &= x_{14} = (x_{13} + w_{33}q_{m3}),\\
f_{21}(x_{11}) &= x_{21} = \sin(\cos^{-1}(x_{11})),\\
f_{22}(x_{12}, x_{14}) &= x_{22} = l_3\cos(x_{14}) + l_2x_{12},\\
f_{23}(x_{13}) &= x_{23} = l_2\sin(x_{13}),\\
f_{24}(x_{14}) &= x_{24} = l_3\sin(x_{14}),\\
f_{31}(x_{11}, x_{22}) &= x_{31} = x_{11}x_{22} + b_1x_A,\\
f_{32}(x_{21}, x_{22}) &= x_{32} = x_{21}x_{22} + b_2y_A,\\
f_{33}(x_{23}, x_{24}) &= x_{33} = l_1 - x_{23} - x_{24}. \qquad (3)
\end{aligned}$$

For convenience, we define x_{ij} as the outputs of the nodes, where x_{31}, x_{32}, x_{33} are the network outputs. They constitute the Cartesian coordinates of the task variable E, i.e., (x_E^W, y_E^W, z_E^W).

Let $\boldsymbol{q}(k)$ be the input vector such that

$$\boldsymbol{q}^T(k) = [\theta, \theta_2, \theta_3, x_B, y_B],$$

and $^{m}X_E^W$ be the measured output vector such that $^{m}X_E^W(k) = [^{m}x_E^W, {}^{m}y_E^W, {}^{\mathrm{m}}z_E^W]^T$. Morover, introduce the weighing vector $\boldsymbol{W}$ such that

$$\boldsymbol{W}^T(k) = [w_{11}, w_{22}, w_{33}]\,.$$

If we set the criterion $\boldsymbol{E}_p$ as the tracking error,

$$\begin{aligned}
E_p &= \sum_{i=1}^{N(3)} (x_{3i} - r_i)^2\\
&= \left[(x_{31} - r_1)^2 + (x_{32} - r_2)^2 + (x_{33} - r_3)^2\right], \qquad (4)
\end{aligned}$$

where $^{\mathrm{d}}\mathrm{X}_{\mathrm{E}}^{\mathrm{W}} = (r_1, r_2, r_3)^T = (^{d}x_E^W, {}^{d}y_E^W, {}^{d}z_E^W)^T$ represents the desired operational coordinates and $^{m}X_E^W = (x_{31}, x_{32}, x_{33})^T = (^{m}x_E^W, {}^{m}y_E^W, {}^{m}z_E^W)^T$ defines the operational coordinates measured in the world frame, then the control objective is to design a control law which guarantees that $E_p \to 0$ as $k \to \infty$, k being running time. The effect of adjusting the weighing vector $\boldsymbol{W}$ to the error E_p is determined by the ordered derivatives $\partial^+ E_p/\partial W(k)$ (Werbos, 1974).

Now, we apply the backpropagation learning rule to generate the appropriate parameter-weighing vector $\boldsymbol{W}(k)$ (Rumelhart *et al.*, 1986). Once determined, the weights are used to update the input vector $\boldsymbol{q}$. The elements of this vector will serve as inputs to the low level controllers of the two agents as illustrated by the block diagram of Fig. 4.

The reference states of the plant at the time $k+1$ are functions of the reference states at the time k and the computed weights at the time $k+1$, and can be symbolically expressed as

$$\boldsymbol{q}(k+1) = \boldsymbol{\psi}(\boldsymbol{q}(k), \boldsymbol{W}(k+1)). \qquad (5)$$

4. Back-Propagation Learning Rule

4.1. Output Layer. The error signal for the j-th output node can be calculated directly as follows:

$$\varepsilon_{3,i} = \frac{\partial^+ E_p}{\partial x_{3,i}} = \frac{\partial E_p}{\partial x_{3,i}}. \qquad (6)$$

Therefore,

$$\varepsilon_{31} = 2(x_e^d - x_{31}),$$
$$\varepsilon_{32} = 2(y_e^d - x_{32}),$$
$$\varepsilon_{33} = 2(z_e^d - x_{33}).$$

4.2. Internal Layers. The error signals of these internal nodes in the j-th position are calculated using the following equation:

$$\varepsilon_{l,i} = \underbrace{\frac{\partial^+ E_p}{\partial x_{l,i}}}_{\substack{\text{Error}\\ \text{signal of}\\ \text{Layer 1}}} = \sum_{m=1}^{N(l+1)} \underbrace{\frac{\partial^+ E_p}{\partial x_{l+1,m}}}_{\substack{\text{Error}\\ \text{signal of}\\ \text{Layer 1+1}}} \cdot \frac{\partial f_{l+1,m}}{\partial x_{l,i}}, \tag{7}$$

$$\varepsilon_{l,i} = \sum_{m=1}^{N(l+1)} \varepsilon_{l+1,m} \frac{\partial f_{l+1,m}}{\partial x_{l,i}}, \tag{8}$$

such that $0 \le l \le L-1$,

$$\varepsilon_{2,j} = \sum_{m=l}^{2} \varepsilon_{3,m} \frac{\partial f_{3,m}}{\partial x_{2,j}}, \quad j = 1, 2. \tag{9}$$

Therefore, the error signals of the nodes in the internal layer are as follows:

$$\varepsilon_{2,1} = \varepsilon_{3,2} x_{2,2},$$
$$\varepsilon_{2,2} = \varepsilon_{3,1} x_{1,1} + \varepsilon_{3,2} x_{2,1},$$
$$\varepsilon_{2,3} = -\varepsilon_{33}, \varepsilon_{2,4} = -\varepsilon_{3,3}.$$

4.3. Input Layer. The first layer contains four neurons arranged in the manner presented in Fig. 3. The general form for the error signal is given by

$$\varepsilon_{1,i} = \sum_{m=1}^{4} \varepsilon_{2,m} \frac{\partial f_{2,m}}{\partial x_{1,i}}. \tag{10}$$

Explicitly, the error signals are

$$\varepsilon_{1,1} = -\varepsilon_{2,1} \frac{x_{1,1}}{\sqrt{1 - x_{1,1}^2}} + \varepsilon_{3,1} x_{2,2}, \tag{11}$$

$$\varepsilon_{1,2} = -\varepsilon_{2,2} l_2 - \varepsilon_{1,3} \frac{1}{\sqrt{1 - x_{1,2}^2}}, \tag{12}$$

$$\varepsilon_{1,3} = \varepsilon_{2,3} l_2 \cos(x_{1,3}) + \varepsilon_{1,4}, \tag{13}$$

$$\varepsilon_{1,4} = l_3 \Big[\varepsilon_{2,4} \cos(x_{1,4}) - \varepsilon_{2,2} \sin(x_{1,4}) \Big]. \tag{14}$$

4.4. Weight Adjustment. To adjust the weights, we make use of the following update:

$$w_{ij}^l(k+1) = w_{ij}^l(k) - \mu \frac{\partial E_p}{\partial w_{ij}^l(k)} \bigg|_{w_{ij}^l(k)}, \tag{15}$$

where

$$\frac{\partial^+ E_p}{\partial w} = \frac{\partial^+ E_p}{\partial x_{l,i}} \frac{\partial^+ f_{l,i}}{\partial w} = \varepsilon_{l,i} \frac{\partial f_{l,i}}{\partial w}. \tag{16}$$

Therefore, the weights are altered according to the following equations:

$$w_{11}(k+1) = w_{11}(k) + \eta \varepsilon_{1,1} \theta \sin(w_{11}\theta), \tag{17}$$
$$w_{22}(k+1) = w_{22}(k) + \eta \varepsilon_{1,2} \theta_2 \sin(w_{22}\theta_2), \tag{18}$$
$$w_{33}(k+1) = w_{33}(k) - \eta \theta_3 \varepsilon_{1,4}, \tag{19}$$
$$b_1(k+1) = b_1(k) - \eta x_B \varepsilon_{3,1}, \tag{20}$$
$$b_2(k+1) = b_2(k) - \eta y_B \varepsilon_{3,2}. \tag{21}$$

Here the last two equations represent the updates of the biases b_1 and b_2. However, the steepest descent algorithm is slow for on-line applications. For this reason, we use the Levenberg-Marquardt algorithm, which has proven to be an effective way of accelerating the convergence rate (Levenberg, 1944; Marquardt, 1963). It only needs information about the first and second derivatives and avoids the inversion of the Hessian matrix. The expression for updating the weights is

$$w_{k+1} = w_k - \left[J^T J + \mu I \right]^{-1} J^T \varepsilon, \tag{22}$$

where I is the identity matrix, J stands for the Jacobian matrix, μ denotes the learning rate and e stands for the error. The weights are updated in each iteration in accordance with the following formulas:

$$\begin{aligned}
w_{11}(k+1) &= w_{11}(k) - \frac{j_{11}}{j_{11}^2 + \mu} \varepsilon_{1,1}, \\
w_{22}(k+1) &= w_{22}(k) - \frac{j_{22}}{j_{22}^2 + \mu} \varepsilon_{1,2}, \\
w_{33}(k+1) &= w_{33}(k) - \frac{j_{33}}{j_{33}^2 + \mu} \varepsilon_{1,4}, \\
b_1(k+1) &= b_1(k) - \frac{j_{44}}{j_{44}^2 + \mu} \varepsilon_{3,1}, \\
b_2(k+1) &= b_2(k) - \frac{j_{55}}{j_{55}^2 + \mu} \varepsilon_{3,2},
\end{aligned} \tag{23}$$

where

$$\begin{aligned} j_{11} &= -(\theta_1+\varphi)\sin(w_{11}(\theta_1+\varphi)),\\ j_{22} &= -\theta_2\sin(w_{22}\theta_2),\\ j_{33} &= \theta_3,\\ j_{44} &= x_B,\\ j_{55} &= y_B. \end{aligned} \tag{24}$$

Thus, the reference state variables at the time $k+1$ are given by

$$\begin{aligned} \theta(k+1) &= w_{11}(k+1)\theta(k),\\ \theta_2(k+1) &= w_{22}(k+1)\theta_2(k),\\ \theta_3(k+1) &= w_{33}(k+1)\theta_3(k),\\ \theta_1 &= \theta-\varphi,\\ x_B(k+1) &= b_1(k+1)x_B(k),\\ y_B(k+1) &= b_2(k+1)y_B(k). \end{aligned} \tag{25}$$

5. Obstacle Avoidance

Obstacle avoidance behavior is added to the previous design in case the effector moves inside a region where an obstacle is present. If we assign a spherical safety region with a radius R_o around an obstacle P_o and another with a radius R_E around the end point P_E, as depicted in Fig. 5, then we can define the detection region by a sphere with a radius R_d, such that $R_d = R_E + R_o$, cf. Fig. 6. The objective is to track the desired reference trajectory while avoiding unforeseen obstacles on the path. Such reactive behavior is easily obtained by adding a second term to the cost function given by Eqn. (4) as the second term that is directly related to the distance between the actual position of the robot and that of the obstacle. Since our goal is to keep the end-effector away from the obstacle, a straightforward choice of the cost function is

$$E_o = \alpha\exp\left(-\beta\|d\|\right), \tag{26}$$

where d is the Euclidean distance between the point of the end-effector and the obstacle such that

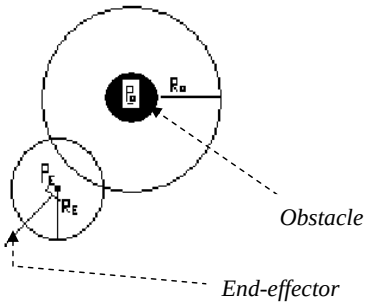
$$d = \left[(x_{31}-o_1)^2+(x_{32}-o_2)^2+(x_{33}-o_3)^2\right]^{1/2}, \tag{27}$$

and $o_1 = x_0, o_2 = y_0$ and $o_3 = z_0$ are the coordinates of the obstacle. Furthermore, α is an inhibitor parameter,

$$\alpha = \begin{cases} 1 & \text{if the end-effector belongs to the detection region.}\\ 0 & \text{otherwise} \end{cases}$$

The overall cost function is then written as

$$E_p = \sum_{k=1}^{N(3)}(x_{3k}-r_k)^2 + \alpha\exp\left(-\beta\,\|d\|\right), \tag{28}$$



Reflex

Fig. 5. Spherical safety regions associated with the end-effector and the obstacle.

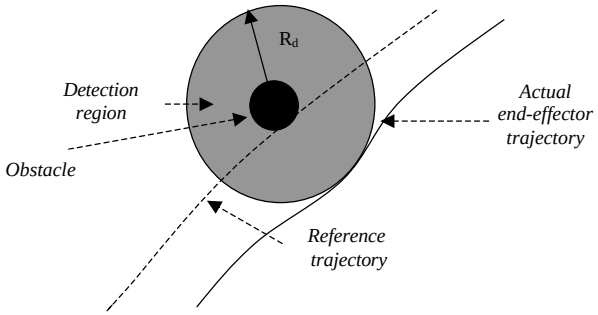


Fig. 6. Reference and obtained end-effector trajectories in the vicinity of an obstacle.

where β is the correcting factor used to adjust the shape of the exponential function. As soon as an obstacle is detected, both types of behavior are activated. But, since the obstacle fitness function dominates, the keep-away behavior is activated making the end-effector abandon the line to the goal. Later, when the path is clear and the obstacle is left behind, the obstacle behavior is inhibited and the tracking behavior regains importance.

6. Simulation Results

Simulation experiments have been performed in order to evaluate the developed approach. It is desirable to move the end-effector from its initial position $P1(1,1,0.2)$ to its desired final position $P2(5,5,0.5)$, by tracking instantaneously a specified linear trajectory of the end-effector generated by a uniform Cartesian movement. The neural network learns the presented desired values and adjusts the weights appropriately in order to present to the system the corresponding reference state variables. The convergence of the algorithm for finding the true values of the weights is proved to be exponentially fast. This convergence helps to enhance the performance of the neural network on motion control. The simulation results are shown in Figs. 7 to 10, and indicate how closely the Cartesian coordinates of the end-effector track their corresponding

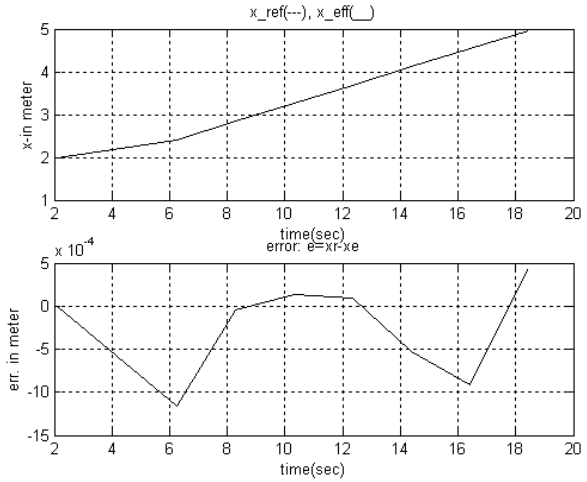


Fig. 7. Desired and measured *x*-trajectories and the resulting error.

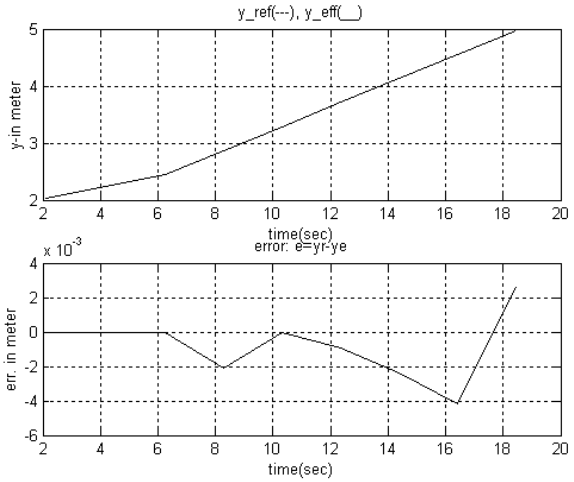


Fig. 8. Desired and measured *y*-trajectories and the resulting error.

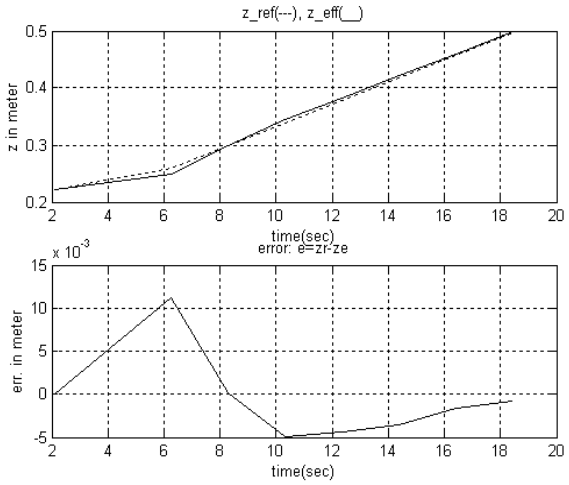


Fig. 9. Desired and measured *z*-trajectories and the resulting error.

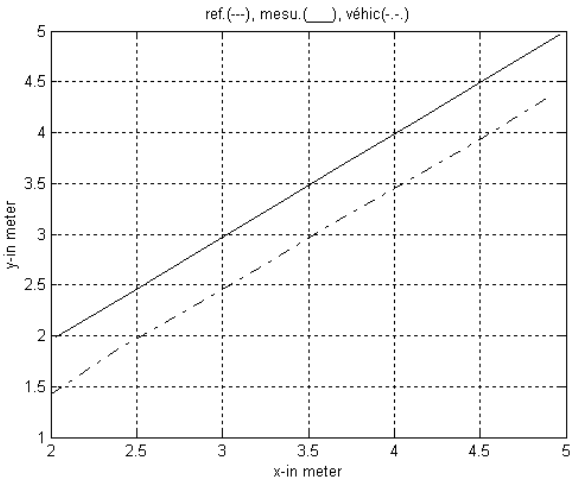


Fig. 10. *X*-*Y* plots of the end-effector and mobile platform trajectories.

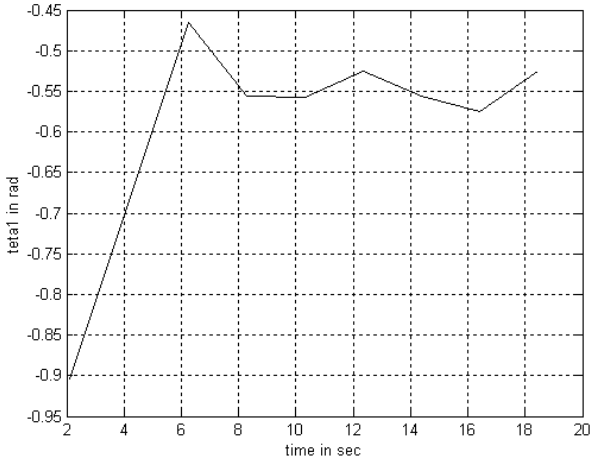


Fig. 11. Angular trajectory θ_1.

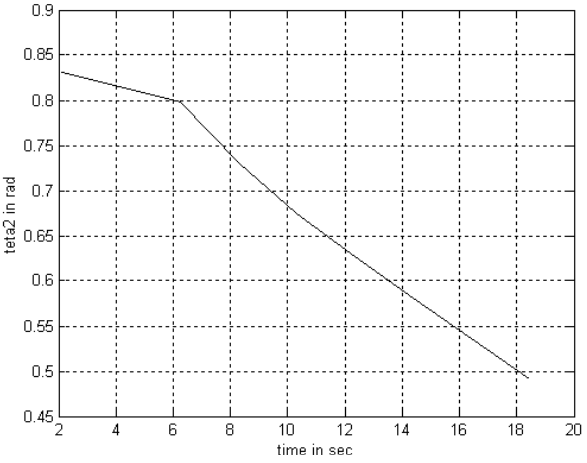


Fig. 12. Angular trajectory θ_2.

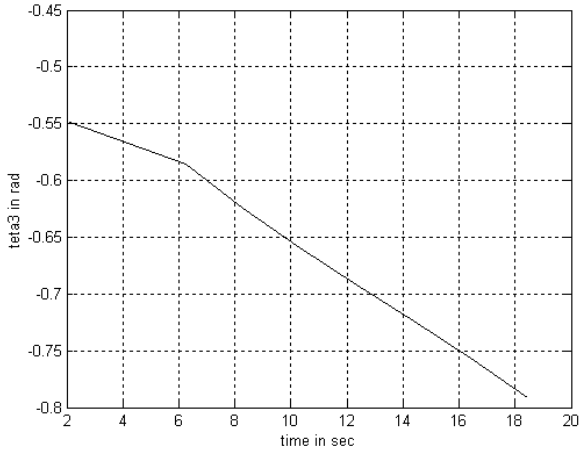


Fig. 13. Angular trajectory θ_3.

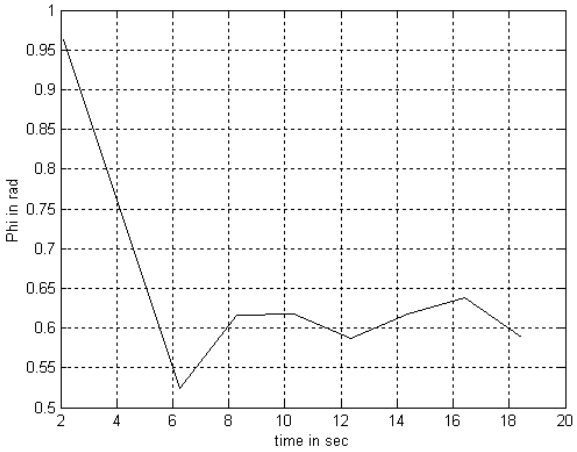


Fig. 14. Angular trajectory φ.

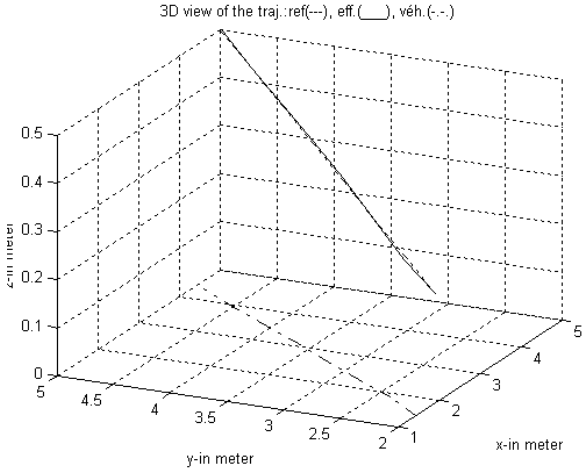


Fig. 15. *x*-*y*-*z* plots of the end-effector and mobile platform trajectories.

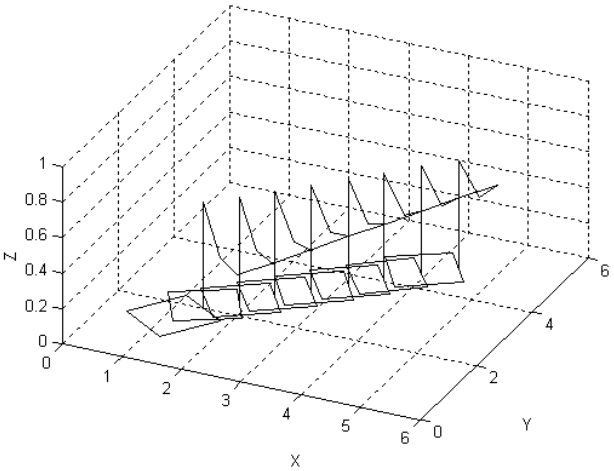


Fig. 16. 3D perspective of the simulation environment.

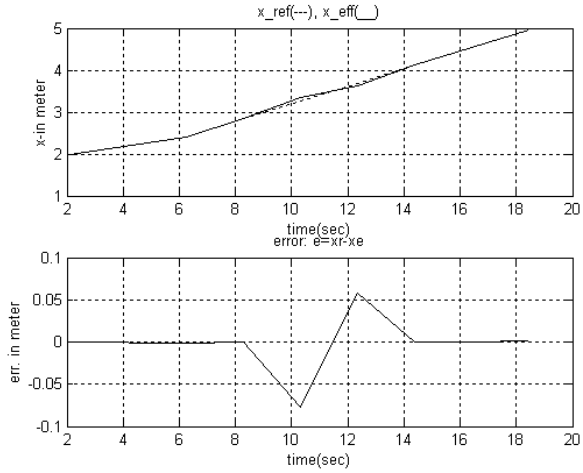


Fig. 17. Desired and measured *x*-trajectories and the resulting error.

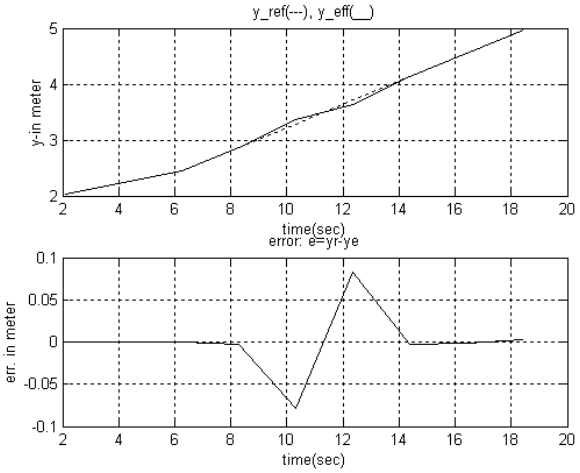


Fig. 18. Desired and measured *y*-trajectories and the resulting error.

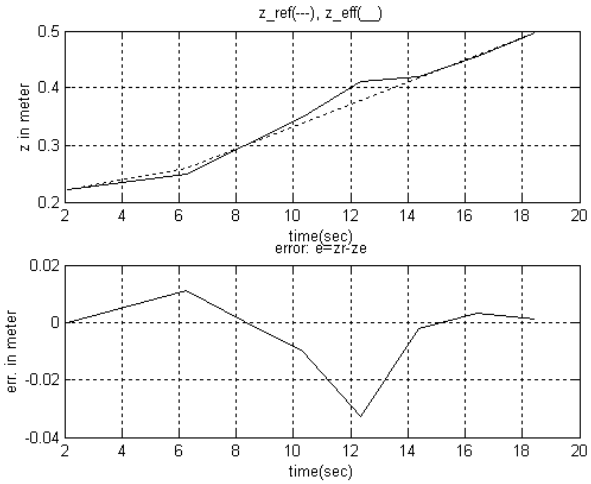


Fig. 19. Desired and measured z-trajectories and the resulting error.

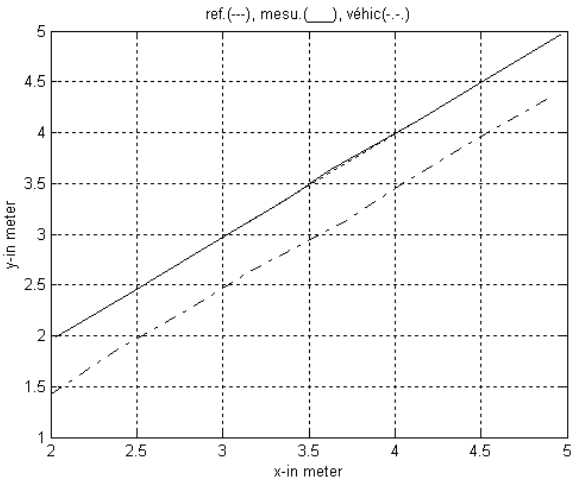


Fig. 20. x-y plots of the end-effector and mobile platform trajectories.

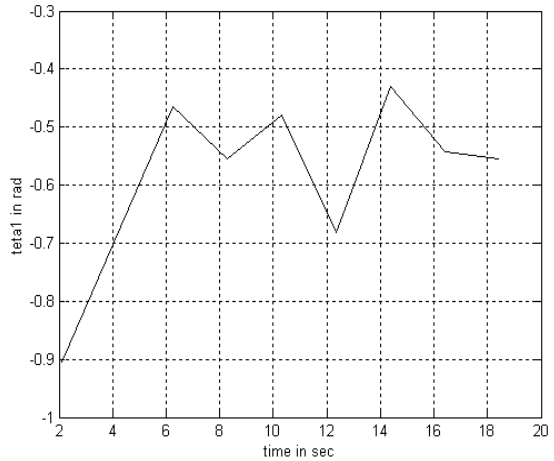


Fig. 21. Angular trajectory θ_1.

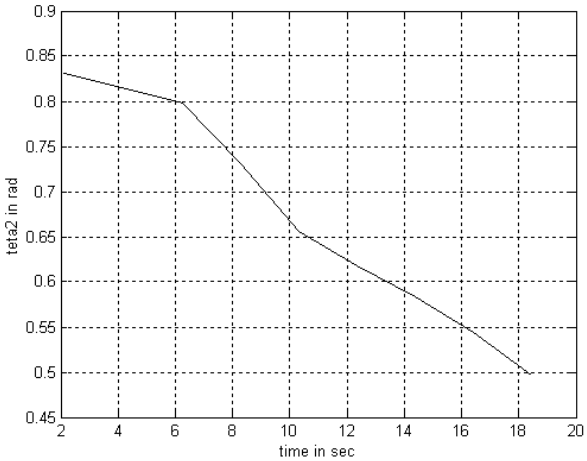


Fig. 22. Angular trajectory θ_2.

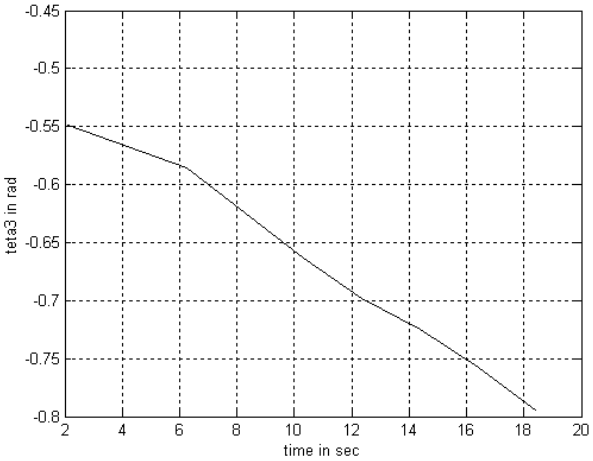


Fig. 23. Angular trajectory θ_3.

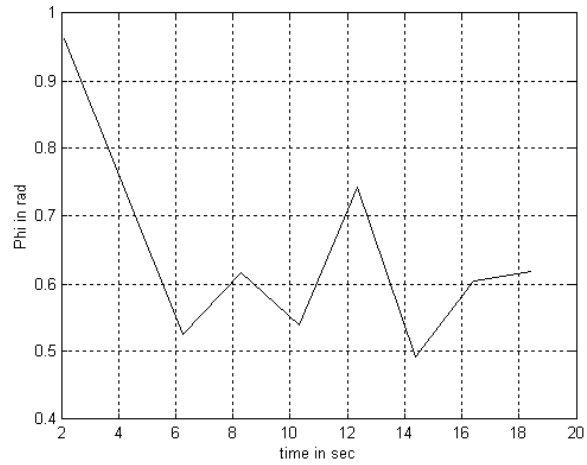


Fig. 24. Angular orientation trajectory φ.

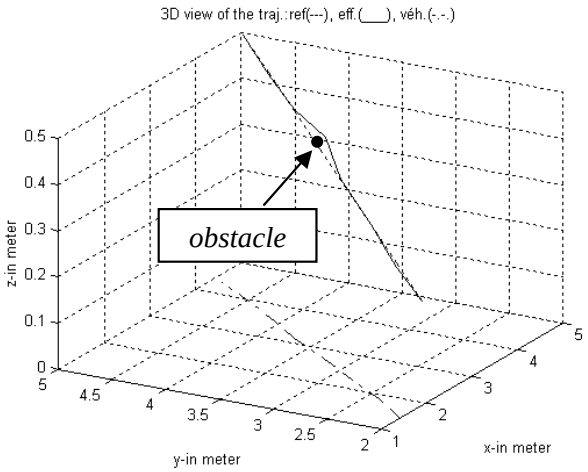


Fig. 25. *x-y* plots of the end-effector and mobile platform trajectories in the presence of an obstacle.

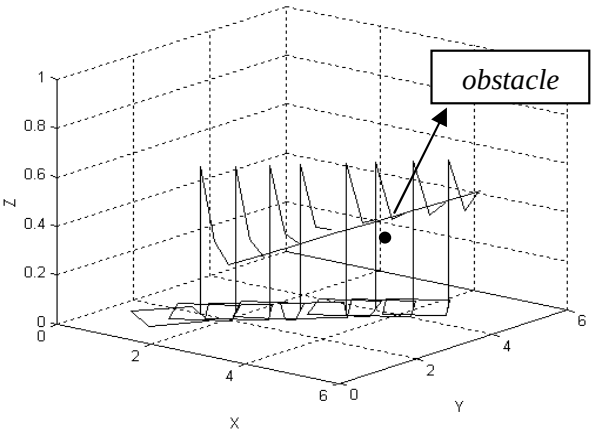


Fig. 26. 3D perspective of the simulation environment.

reference values. We notice that small departures from the reference trajectories are due to the cumulated tolerable errors from the learning process. The learning algorithm was run by using a learning rate of $\mu = 0.05$ for a period of time not exceeding real time control. All the weights were initialized to unity. At each step, the learning rate was updated depending on the obtained behavior. If the overall error was improved, then the learning rate was increased by the value $\mu = \mu \cdot \mu_{inc}$; otherwise, it was decreased by the value $\mu = \mu \cdot \mu_{dec}$. Initially, μ_{inc} and μ_{dec} took the values of 1.05, and 0.95, respectively. Figs. 11 to 14 show the plots of the manipulator angular values as well as the orientation of the mobile platform, and Fig. 15 clearly shows the trajectories of the end-effector and the mobile platform in the xyz-space. Figure 16 depicts a 3D perspective of the simulation environment. Similarly, when there is an obstacle in the middle path of the end-effector, the second term of the cost function given by (28) is activated. The neural network provides angular values necessary to push the end-effector away from the line to the obstacle. Later, when the path is clear and the obstacle is not in the detection region of the end-effector, the second term of (28) is inhibited and the neural network seeks appropriate angular values that make the end point of the robot manipulator track successfully the desired trajectory. Curves similar to those presented above are shown in Figs. 17 to 26. They reflect the results obtained in case an obstacle is met and the end-effector manages to avoid it. In this case, too, the obtained results show the effectiveness of the proposed controller scheme. The proposed neural network with the back propagation training rule is significantly simpler than the use of pseudoinverses, mainly when there is a need to avoid obstacles that are on the desired path of the end-effector.

7. Conclusion

In this paper we proposed a new scheme to motion control designed to perform mobile manipulation tasks. It combines the motion of the robot manipulator with that of the mobile platform to execute an end-effector tracking trajectory task. A neural network assures the mapping from the operational space to the generalized coordinate space. The latter was implemented in order to supply the low level controllers of each sub-system with appropriate generalized reference values when the operational coordinates of the end point of the manipulator are given. The results obtained by simulation show that the proposed method performed very well. This controller offers a general solution to a class of mobile manipulators executing a task in the operational space in an unknown environment.

References

Abdessemed F. (2004): *Control and planning of a mobile manipulator*. — Proc. 12-th Mediterranean Conf. *Control and Automation*, Kusadasi, Turkey.

Brock O. and Khatib O. (2000): *Real time replanning in high-dimensional configuration spaces using sets of homotopic paths*. — Proc. Int. Conf. *Robotics and Automation*, San Francisco, CA, USA, Vol. 1, pp. 550–555.

Brock O., Khatib O. and Viji S. (2002): *Task-consistent obstacle avoidance and motion behavior for mobile manipulator*. — Proc. IEEE Int. Conf. *Robotics and Automation*, Washington, DC, USA, pp. 388–393.

Chen M.W. and Zalzala A.M.S. (1997): *Dynamic modeling and genetic-based trajectory generation for non-holonomic mobile manipulators*. — Contr. Eng. Pract., Vol. 5, No. 1, pp. 39–48.

Dubowsky S. and Tanner A.B. (1988): *A study of the dynamics and control of mobile manipulator subjected to vehicle disturbances*. — Proc. 4-th Int. Symp. *Robotics Research*, pp. 111–117.

Fischer C., Buss M. and Shmidt G. (1996): *Soft control of an effector path for a mobile manipulator*. — Proc. Int. Conf. *On robotics and Manufaturing – ISRASM*, Montpellier, France, pp. 299–306.

Khatib O. (1993): *Mobile robotic manipulation*. — Proc. Symp. VDI/VDE Soc. *Measurement and Automation Intelligent Robot Control*, Langen, Germany, pp. 51–66.

Khatib O., Yokoi K., Chang K., Ruspini D., Holmberg R. and Casal A. (1996): *Vehicle/arm coordination and multiple mobile manipulator decentralised cooperation*. — Proc. Int. Conf. *Intelligent Robots and Systems, IROS96*, Osaka, Japan, pp. 546–553.

Lee J.K. and Cho H.S. (1997): *Mobile manipulator motion planning for multiple tasks using global optimization approach*. — J. Intell. Robot. Syst., Vol. 18, No. 2, pp. 169–190.

Levenberg K. (1944): *A method for the solution of certain nonlinear problems in least squares*. — Qty. Appl. Math., Vol. 2, No. 2, pp. 164–168.

Marquardt D.W. (1963): *An algorithm for the estimation of nonlinear parameters*. — SIAM J. Contr., Vol. 11, No. 2, pp. 431–441.

Nagatani K. and Yuta S. (1996): *Door-opening behavior of autonomous mobile manipulator by sequence of action primitives*. — J. Robot. Syst., Vol. 13, No. 11, pp. 709-721.

Nakamura Y. (1991): *Advanced Robotics: Redundancy and Optimization*.— New York: Addison-Wesley.

Nassal U.M. (1996): *Motion coordination and reactive control of autonomous multi manipulators systems*. — J. Robot. Syst., Vol. 13, No. 11, pp. 737–754.

Pin F.G., Morgansen K.A., Tulloch F.A., Hacker C.J. and Gower K.B.: (1996a): *Motion planning for mobile manipulator with a non-holonomic constraints using the FSP (Full Space Parametrization) Method*. — J. Robot. Syst., Vol. 13, No. 11, pp. 723–736.

Pin F.G., Morgansen K.A., Tulloch F.A., Hacker C.J. and Gower K.B. (1996b): *Motion planning for mobile manipulator with non-holonomic constraint using the FSP method*. — J. Robot. Syst., Vol. 13, No. 11, pp. 723–736.

Renaud M. and Dauchez P. (1999): *Modélisation et commande des manipulateurs mobiles à roues*. — Proc. Conf. *Journées Nationales de la Recherche en Robotique* (JNRR'99), Monpellier, France, pp. 179–192.

Rumelhart D.E, Hinton G.E. and Williams R.J. (1986): *Learning internal representations by error propagation*, In: Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. 1 (Rumelhart D.E. and McClelland J.L., Eds.). — Cambridge, MA: MIT Press Chap. 8, pp. 318–362.

Seraji H. (1995): *Configuration control of rover-mounted manipulators*. — Proc. IEEE Int. Conf. *Robotics and Automation*, Nagoy, Japan, pp. 2261–2266.

Werbos P. (1974): *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. — Ph.D. dissertation, Harvard Univ.

Yamamoto Y. and Yun X. (1994): *Coordination locomotion and manipulation of a mobile manipulator*. — IEEE Trans. Automat. Contr., Vol. 39, No. 6, pp. 1326–1332.

Zhao M., Ansari N. and Hou E. S. (1997): *Mobile manipulator path planning by a genetic algorithm*. — J. Robot. Syst., Vol. 11, No. 3, pp. 143–153.

Received: 17 March 2006
Revised: 29 September 2006