International Journal of Robotics and Autonomous Systems, Vol. 39, No. 2-3, May, 2002

Neural Predictive Control for a Car-like Mobile Robot

Dongbing Gu and Huosheng Hu

Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, UK
Fax: +44-1206-872788
Email: dgu@essex.ac.uk, hhu@essex.ac.uk

Abstract: This paper presents a new path-tracking scheme for a car-like mobile robot based on neural predictive control. A multi-layer back-propagation neural network is employed to model non-linear kinematics of the robot instead of a linear regression estimator in order to adapt the robot to a large operating range. The neural predictive control for path tracking is a model-based predictive control based on neural network modelling, which can generate its output in term of the robot kinematics and a desired path. The desired path for the robot is produced by a polar polynomial with a simple closed form. The multi-layer back-propagation neural network is constructed by a wavelet orthogonal decomposition to form a wavelet neural network that can overcome the problem caused by the local minima when training the neural network. The wavelet neural network has the advantage of using an explicit way to determine the number of the hidden nodes and initial value of weights. Simulation results for the modelling and control are provided to justify the proposed scheme.

Key words: Model predictive control, Wavelet neural network, Path tracking

1. INTRODUCTION

Car-like mobile robots are mainly used in industry, ports, planet exploration, nuclear waste cleanup, agriculture and mining since they have the necessary loading capability. The control scheme for such mobile robots is traditionally decomposed into three subtasks: trajectory generation, position estimation and path tracking. Although behaviour-based approaches have been very popular since Brooks' invention in 1980s[2], their sole implementation on these mobile robots for many complex applications causes some difficulty. Instead, a behaviour-based approach can be incorporated with a traditional planning-based system to form a hybrid system to deal with unexpected situations [1].

The aim of trajectory generation is to generate a feasible path that allows the robots to move smoothly. A path consisting of straight lines and circular arc segments is easy to generate, but may not be suitable for a car-like mobile robot which has the non-holonomic constraint and its steering angles at the line-arc transition points have discontinuous curvature. Although a Clothoid's trajectory [10] can provide smooth transitions with a continuous curvature, its disadvantage is lack of the closed-form expression. Nelson proposed a continuous-curvature polar polynomial for path generation [14], which has a closed form expression and is easy to calculate. Pinchard improved Nelson's polar polynomial with variable speeds, leading to a fifth-order polynomial for a continuous-curvature path [16].

Position estimation plays a key role in path tracking. Many approaches have been proposed in terms of the different sensors on robots. Odometry is simple, inexpensive, and easy to implement in real time, but suffers from the unbounded accumulation of errors. The triangulation approach is appropriate for initialising position, but is too sensitive to angle measurements when robots move around. Grid-based position [12] is a successful method in dealing with sensory uncertainty, but is not suitable for model-based predictive control due to the time it takes. A Kalman filter is a feasible estimation approach that can fuse multiple sensory measurements to provide relatively accurate results.

Path tracking for a car-like robot is in fact a feedback control problem to which model-based predictive control (MPC) has been an effective mechanism [4][19]. Most MPC control applications are based on linear models of dynamic systems to predict outputs over a certain horizon. Future sequences of control signals are evaluated by minimising a cost index that takes account of the future output prediction errors over a reference trajectory, as well as control efforts. However, when the systems are non-linear and are operated over a large dynamic range, the use of linear models becomes impractical, and the identification of non-linear models for control becomes absolutely necessary.

A Neural network is an attractive tool to model complex non-linear systems due to its inherent ability to approximate arbitrary continuous functions. Conclusive proofs were given that feed-forward neural networks with one hidden layer are capable of approximating any continuous function on a compact set in a very precise and satisfactory sense [8]. Some researchers have successful applied feed-forward neural networks as system models to MPC, namely neural predictive control (NPC). For example, NPC has been used in manipulator control [3][5][20][21], chemical process control [17][24], mobile robot path tracking [18][28], and other applications [11][23]. In these research works, the control schemes of NPC were changed into non-linear optimisation problems. The optimisation constraints were the system dynamics represented by neural networks. The main drawbacks are undesirable local minima and slow convergence of back-propagation learning. Moreover, the implementation of multiple feed-forward neural networks suffers from the lack of efficient constructive approaches, both for determining parameters of neurones and for choosing network structures. Other algorithms such as random search techniques and genetic algorithms have been adopted to overcome the local minima caused by gradient descent rules [11][18]. However, the computational cost of these algorithms will degrade the performance of real-time control systems.

Recently, wavelet decomposition emerged as a new powerful tool for function approximation in a manner that readily reveals properties of the arbitrary L^2 function (energy-finite and continuous or discontinuous). Combining wavelets and neural networks can result in wavelet neural networks with efficient constructive approaches [15][25]. Because wavelet neural networks can approximate arbitrary functions belonging to L^2 space, they have results in a wide variety of application areas, like speech recognition [22] and digital communication [6]. They were also applied in robot applications where robot kinematics is a continuous function and operates in a bounded range [20]. The wavelet neural networks can further result in a convex cost index to which simple iterative solutions such as gradient descent rules are justifiable and are not in danger of being trapped in local minima when choosing the orthogonal wavelets as the activation functions of nodes in neural networks [25].

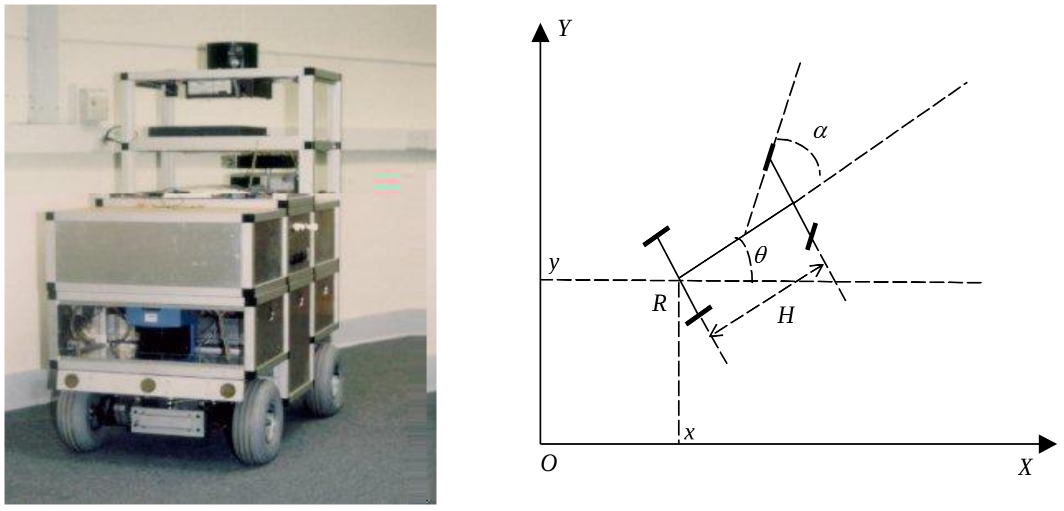


Figure 1 A car-like mobile robot

The rest of this paper is organised as follows. A car-like mobile robot being modelled in this research is outlined in section 2, including a brief explanation of robot kinematics, trajectory generation based on a polar polynomial, and position estimation by a Kalman filter. Section 3 investigates the wavelet neural network modelling of robot kinematics. Neural predictive control for path tracking is described in section 4. The simulation results for the proposed control scheme are given in section 5, including modelling results and control results. Finally, section 6 presents a brief conclusion and future work.

2. A CAR-LIKE MOBILE ROBOT

The robot considered in this research is a car-like mobile robot with four pneumatic tyre wheels. Two front wheels serve for steering and two rear wheels serve for tracking, as shown in figure 1.

The robot has a modular structure such that it can be readily extended and easily configured to allow the use of a range of sensor modules or actuator modules. Two 500 count/revolution rotary encoders are fixed onto two motor shafts for position servo control by a PID controller. Another two 500 count/revolution rotary encoders have been connected to the rear axes near the rear wheels using 3:1 ratio pulley and belts. The reason we use these two extra encoders is to compensate for the position errors caused by inaccuracy of the steering angle. The robot has a weight of 100 Kg and a maximum speed of 1 m/s. Sensors include optical encoders, a sonar array, a laser scanner, an infra-red proximity, and a stereo head [7].

2.1 Robot kinematics

Assume that the midpoint of the rear axis of the mobile robot is a reference point (R as shown in figure 1) in the global frame *XOY*. A state vector $\mathbf{x}(k) = [x(k), y(k), \theta(k)]^T$ is used to represent the position (x, y) and the orientation (θ) of the robot. Its discrete kinematics equation can be described as follows:

$$\mathbf{x}(k+1) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + v(k)\Delta t \cos(\theta(k))\cos(\alpha(k)) \\ y(k) + v(k)\Delta t \sin(\theta(k))\cos(\alpha(k)) \\ \theta(k) + v(k)\Delta t \sin(\alpha(k)) / H \end{bmatrix} \tag{1}$$

where $\mathbf{u}(k) = [v(k), \alpha(k)]^T$ is the control vector for motion tracking, Δt is the sample period and H is the wheelbase between front and rear axes. The two control variables are the speed of robot ($v(k)$) and the steer angle ($\alpha(k)$).

Such a non-linear system is open loop controllable, which can be linearised in order to use traditional linear feedback control to regulate the robot. But if the robot operates over a large range in its state space, especially when the robot turns around corners, the linearisation of the kinematics will lead to the loss of controllability. Since the MPC’s models are based on linear regressions, they are not suitable to non-linear robotic systems. Furthermore, the kinematics is difficult to model precisely because there exist many physical sources of uncertainty, for example backlash, friction, load, and pneumatic tyres. Adaptive control should be employed to deal with the uncertainty.

2.2 Position estimation

Two kinds of sensors are used to estimate the robot state in this research. The optical encoders mounted on two rear wheels are used for odometry calculation. The problems for odometry are (i) surface roughness and undulation may cause the distance to be over-estimated. (ii) wheel slippage can cause distance to be under-estimated. (iii) variations in load can distort the odometry wheels and introduce additional errors. Because odometry leads to unbounded accumulation of the errors, it is clear that absolute position estimation would be necessary. Another kind of sensors is the laser scanner on the robot interacting with artificial beacons mounted around the operating space, which can be used for the absolute position estimation. A priori information about the beacons can be viewed as a form of world model. With this model, the state of the robot is determined by integrating odometry with the bearing of the beacons measured by the laser scanner. To do this, an Extended Kalman Filter (EKF) is used to estimate the state based on information from the odometry and the laser scanner under the assumption that the errors of measurement follow a Gaussian distribution [7].

2.3 Path Generation

A path consisting of straight lines and circular arc segments is easy to generate, but may not be suitable for a car-like mobile robot which has non-holonomic constraints and its steering angles at the line-arc transition points can generate discontinuities of the trajectory curvatures. A Clothoid's trajectory [10] provides smooth transitions with a continuous curvature. Its disadvantage is lack of closed-form position expression. Nelson proposed a continuous-curvature polar polynomial for path generation [14], which is a closed form expression and easy to calculate.

Pinchard [16] improved the Nelson's polar polynomial with variable speeds, leading to a fifth-order polynomial for a continuous-curvature path, which is used here for trajectory generation. The position, tangent and curvature constraints yield a fifth-order polynomial for a continuous-curvature path. Its general form is:

$$r(\phi) = a_0 + a_1\phi + a_2\phi^2 + a_3\phi^3 + a_4\phi^4 + a_5\phi^5 \tag{2}$$

where r is the polar radius and ϕ the polar angle shown in figure 2. The initial and final conditions of the transition between paths are:

$$r=R_1,\ r'=0,\ c=0, \quad at\ \phi = 0$$
$$r=R_2,\ r'=0,\ c=0, \quad at\ \phi = \Phi$$

where c is curvature. Φ is the angle the robot will turn. R_1 and R_2 are the initial and final radii. v_1 and v_2 are the initial and final velocities. With these conditions, the coefficients of a general polar polynomial can be given as follows:

$$\begin{aligned}
&a_0 = R_1, a_1 = 0, a_2 = R_1/2 \\
&a_3 = \frac{-20R_1 + 20R_2 - 3R_1\Phi^2 + R_2\Phi^2}{2\Phi^3} \\
&a_4 = \frac{30R_1 - 30R_2 + 3R_1\Phi^2 - 2R_2\Phi^2}{2\Phi^4} \\
&a_5 = \frac{-12R_1 + 12R_2 - R_1\Phi^2 + R_2\Phi^2}{2\Phi^5}
\end{aligned} \tag{3}$$

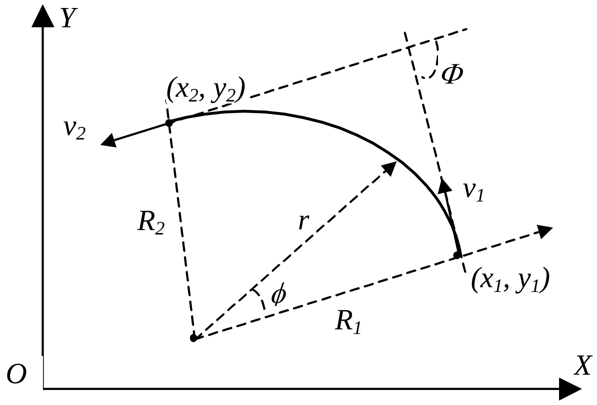


Figure 2 Path generation

2.4 NPC Path tracking scheme

To control a robot to track a path, the desired path and the measured path are two basic information sources used for a controller. The trajectory generation module shown in figure 3 produced a desired path represented by a set of states $\mathbf{x}_r(k)=[x_r(k), y_r(k), \theta_r(k)]^T$. The controller, either MPC controller or PID controller shown in figure 3, yields the control vector $\mathbf{u}(k) =[v(k), \alpha(k)]^T$. The motor movement leads to position changes that are estimated by an EKF, based on the sensory information. The current state of the robot obtained from the position estimation module is estimated as $\boldsymbol{x}_m(k)=[x_m(k), y_m(k), \theta_m(k)]^T$. PID control is a simple and feasible choice. It uses the state errors ($\boldsymbol{x}_r(k)$- $\boldsymbol{x}_m(k)$) as feedback signals, which are further decomposed into tangential and normal errors. The dash line loop, which is in parallel with a MPC controller in figure 3, shows the PID control scheme. More details for the PID control scheme in path tracking can be found in our previous paper [7].

MPC is based on an assumed model of the robot kinematics and on an assumed scenario for the future control signals. The solid line parts in figure 3 show the MPC control scheme. With the desired trajectory and measured path mentioned above, it could produce a sequence of control signals. Only the first one is applied to the robot. Due to the reasons in modelling kinematics discussed in subsection 2.1, a neural network is employed to model the kinematics expressed in (1). It uses the measurement states $\boldsymbol{x}_m(k)$ and control variables $\boldsymbol{u}(k)$ to produce an adaptive kinematics model that provides a substrate for the MPC controller. Section 4 will present the details about the algorithm.

The neural network used to model the kinematics is trained according to $\boldsymbol{u}(k)$ and $\boldsymbol{x}_m(k)$. Initially, the PID controller controls the robot to track the path in order to produce the training data for neural network learning. During the training stage, the MPC controller is replaced with the PID controller shown as a dash line in figure 3. Once the neural network has been trained, it can be used for the MPC controller and will be adapted during the path tracking process.

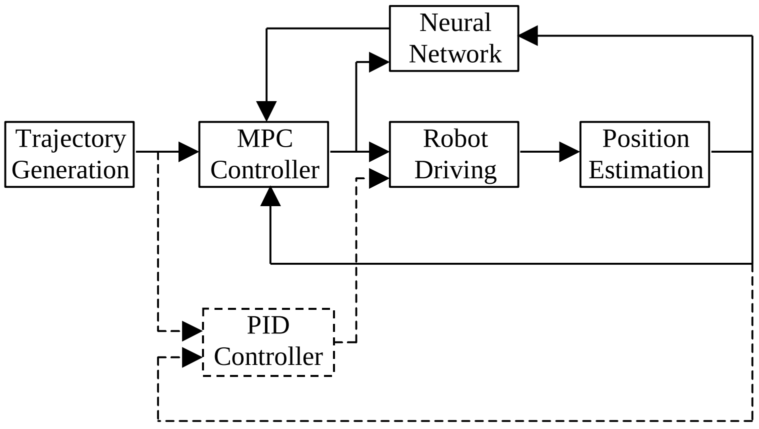


Figure 3 NPC for path tracking

3. MODELLING ROBOT KINEMATICS

Due to the similarity between wavelet decomposition and hidden layers in neural networks, wavelet neural networks were proposed by combining wavelet and neural networks in [25]. These wavelet neural networks can remedy the weakness of a large dimension of wavelet decomposition and the constructive problem of neural networks. In this section, the way to model the robot kinematics by using wavelet neural networks is proposed.

3.1 Wavelet neural network

In multi-resolution signal analysis, a square integral function *f(t)* in L^2 space can be decomposed into a summation of a series of weighted wavelet components[25]:

$$f(t)=\sum_{m,n}\left\langle f,\psi_{m,n}\right\rangle\psi_{m,n}(t) \tag{4}$$

where $\psi_{m,n}(t) = 2^{m/2}\psi(2^m t - n)$ is the orthogonal wavelet function. The weight in each wavelet component is $\langle f, \psi_{m,n} \rangle$ that denotes the inner product of function *f(t)* and wavelet $\psi_{m,n}$. The formula (4) can be rewritten in the following form according to the wavelet theory:

$$f(t) = \sum_{n} \langle f, \varphi_{M,n} \rangle \varphi_{M,n}(t) + \sum_{m>M,n} \langle f, \psi_{m,n} \rangle \psi_{m,n}(t) \tag{5}$$

where $\varphi_{M,n}(t)$ is called scaling function. Most of the physical functions used in engineering have bounded energy and can be regarded as square integral functions. Therefore they can be decomposed into an expression as shown in (5).

What is more important for the function decomposition is that for a sufficiently large *M*, any square integral function *f(t)* can be approximated arbitrarily closely by the first term of (5)[25]. That is for any $\varepsilon > 0$

$$\left\| f(t) - \sum_{n} \langle f, \varphi_{M,n} \rangle \varphi_{M,n}(t) \right\| < \varepsilon$$

It shows that the function approximation for *f(t)* in (5) can be expressed by the truncated wavelet decomposition as:

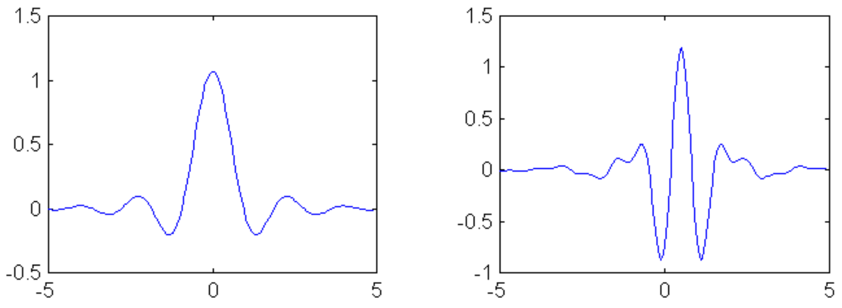
$$f(t) \approx \sum_{n} \langle f, \varphi_{M,n} \rangle \varphi_{M,n}(t) = \sum_{n} C_n \varphi_{M,n}(t) \tag{6}$$

It means that some fine components (high frequency) that can be represented by a series of the wavelet functions for the function *f(t)* are neglected and some coarse components (low frequency) that can be represented by a series of the scaling function are preserved to approximate the originate function under the *M* scale.

How to choose a wavelet function is very important in wavelet decomposition. Most orthogonal wavelets lack the finite support set that can guarantee good time resolution. Daubechies' wavelets possess the finite support set, but are not sufficiently smooth to guarantee good frequency resolution. Lemarie-Meyer's wavelets are orthogonal and posses good frequency and time resolution, but there is no simple closed form expression required by the real-time applications in this paper. Walter and Zhang in [27] found that two classes of wavelets can be represented by the simple closed form expressions and are orthogonal, which can be seen as the particular examples of Lemarie-Meyer's wavelets. We choose one of them as the function approximation base in this paper. Its scaling function expression is

$$\varphi(t) = \frac{\sin \pi(1-\beta)t + 4\beta t \cos \pi(1+\beta)t}{\pi t(1-(4\beta t)^2)}$$

Figure 4 shows its scaling function and its wavelet where *β=1/4*.



(a) Scaling function (b) Wavelet function

Figure 4 Orthogonal wavelet

The function approximation expression (6) has a similar structure with a 3-layer neural network as shown in figure 5. The number of hidden nodes is decided by wavelet translation *n* that depends on the support set of *f(t)*. Here we can assume a positive integer *N* to form a range [-*N*, *N*] that can cover the support set of *f(t)*. For a multiple dimension case, the scaling functions or wavelets are generated by tensor products of one-dimensional scaling functions or wavelets.

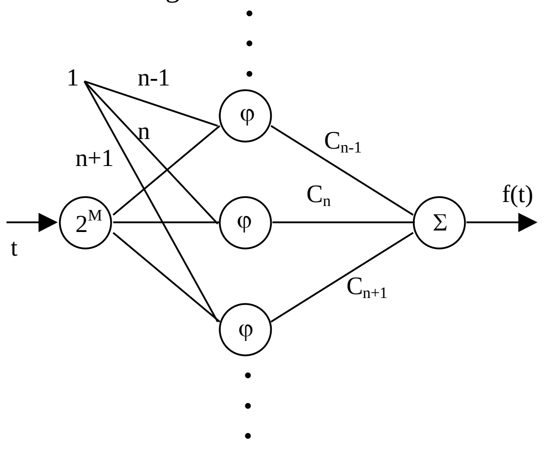


Figure 5 The wavelet neural network where $C_n = \langle f, \varphi_{M,n} \rangle$

3.2 Neural network modelling

We use a one-step-ahead predictive model to describe the robot kinematics (1). Thus, formula (1) can be rewritten as:

$$\mathbf{x}(k+1)=F(\mathbf{z}(k)) \tag{7}$$

where $\mathbf{z}(k)=[v(k), \alpha(k), x_m(k), y_m(k), \theta_m(k)]^T$ and $\mathbf{x}(k)=[x(k), y(k), \theta(k)]^T$. This predictive model can be represented by a wavelet neural network as shown in figure 6, where the input vector $\mathbf{z}(k)$ with its size $p=5$ consists of the control vector and the estimated state, and the output vector $\mathbf{x}(k)$ with its size $q=3$ is the predicted state. The activation function $\varphi_d(t)$ in the hidden layer is a multidimensional scaling function that is the tensor product of one-dimensional scaling functions. Other network parameters are:

- the number of the hidden nodes $r=2N+1$;
- the weight matrix from input to hidden layer is W_1;
- the weight matrix from input to hidden layer is W_2.

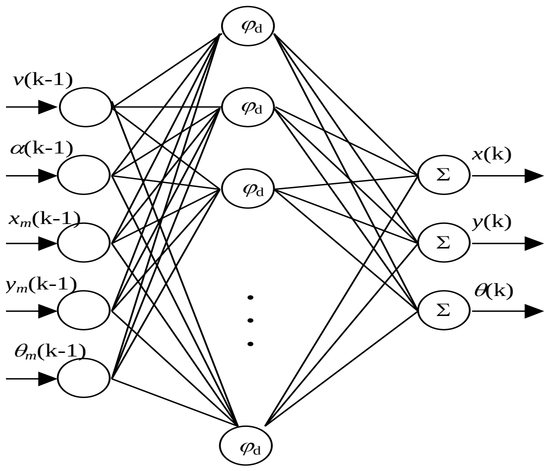


Figure 6 Wavelet neural network for robot kinematics

Based on wavelet neural networks, the robot kinematics can be represented by:

$$x(k+1)= W_2\Phi(W_1\mathbf{z}(k)) \tag{8}$$

where $\Phi=[\varphi_d, \dots, \varphi_d]^T$. The weight matrix W_1 is invariable during the training cycle and expressed as follows in terms of the wavelet decomposition.

$$W_1 = \begin{bmatrix} 2^M & \cdots & 2^M \\ \vdots & \vdots & \vdots \\ 2^M & \cdots & 2^M \end{bmatrix}_{(2N+1)\times 5} \tag{9}$$

Initial values of weight matrix W_2 can also be generated by the wavelet decomposition:

$$W_2(0) = \begin{bmatrix} C_{1,-N} & C_{1,-N+1} & \cdots & C_{1,N} \\ C_{2,-N} & C_{2,-N+1} & \cdots & C_{2,N} \\ C_{3,-N} & C_{3,-N+1} & \cdots & C_{3,N} \end{bmatrix}_{3\times(2N+1)} \tag{10}$$

It can be further updated by a steepest-gradient-decent algorithm.

$$W_2(k+1)= W_2(k)- \boldsymbol{\mu}\Phi\mathbf{e}^T \tag{11}$$

where $\boldsymbol{e} =[\boldsymbol{x}_m(k)-\boldsymbol{x}(k)]$ is the error between the estimated state from EKF and the predicted state from the wavelet neural network, and $\boldsymbol{\mu}$ is the update rate.

4. PREDICTIVE CONTROL SCHEME

A predictive controller is proposed here, based on the prediction model that predicts the future values of the robot state. The prediction model is constructed by the wavelet neural network shown in figure 6. The errors between predicted states and state estimates, combining with the control variations, are used to form a quadratic index. The process of minimising the quadratic index is to find an optimal control vector. At each control cycle during the path tracking, the wavelet neural network is first adapted based on the estimated state, and then optimisation control is executed.

4.1 MPC formation

The outputs of the neural network (8) presented in the previous section can be rewritten in the state space form as follows:

$$\mathbf{x}(k+1) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{z}(k)) \\ f_2(\mathbf{z}(k)) \\ f_3(\mathbf{z}(k)) \end{bmatrix} = F[\mathbf{z}(k)] = F[\mathbf{x}_m(k), \mathbf{u}(k)] \tag{12}$$

The MPC controller is to optimise a cost index $J(\boldsymbol{x}(k), \boldsymbol{u}(k))$ under the constraints of formula (12):

$$\min_{\mathbf{u}(k)} J(\mathbf{x}(k), \mathbf{u}(k))$$

The current control vector is chosen to minimise the state errors and control energy over several steps in the future so that the path tracking of the robot is smooth and stable. Therefore, the cost index can be expressed as

$$J = \frac{1}{2}\sum_{i=N_1}^{N_2} \left\| \mathbf{x}_r(k+i) - \mathbf{x}(k+i) \right\|^2 + \frac{1}{2}\sum_{i=1}^{N_\mu} \boldsymbol{\lambda}_i \left\| \Delta\mathbf{u}(k+i-d) \right\|^2 \tag{13}$$

where $\lambda(i)$ is a weighting matrix (2 by 2 positive definite symmetric), penalising the control effort. This cost index reflects the desire to drive the next robot state $\boldsymbol{x}(k)$ close enough to the desired state $\boldsymbol{x}_r(k)$ generated by the fifth order polynomial introduced in section 2.3 without excessive expenditure of control effort. The cost index is conditioned on data up to control cycle k, assuming no future measurements are available. At each control cycle, an optimal control sequence is calculated, but only the first one is applied to the robot. This process will be repeated at the next control cycle to form a *receding horizon* optimisation procedure.

The notations used for equation (13) are

- N_1 and N_2 are the minimum and maximum *output horizons* respectively;
- N_μ is called the *control horizon*;
- $\Delta\boldsymbol{u}(k)$ is a set of control increments, $\Delta\boldsymbol{u}(k) = \Delta\boldsymbol{u}(k) - \Delta\boldsymbol{u}(k-1)$ and $\Delta\boldsymbol{u}(k+i)=0$, if $i>N_\mu-1$;
- d is the dead-time of the system.

In general, N_2 is chosen to encompass all the responses that are significantly affected by the current control. In practice, it is more typically set to approximate rise-time of the system. N_1 is selected according to dead time of the robot system, and is often taken as 1.

4.2 Optimisation algorithm

In each control cycle k, we need to compute a control increment matrix $\Delta\mathbf{U}(k)$ $(\Delta\mathbf{U}(k)=[\Delta\boldsymbol{u}(k), \ldots, \Delta\boldsymbol{u}(k+N_\mu-1)]^T$ that has K vectors and choose the first one as the actual control increments for outputs of the controller. For the non-linear system presented in (12), there are no analytic expressions available. We have to implement the non-linear optimal algorithm to find an optimal solution of a control vector during the interval k and $k+1$ on the fly. It should be noticed that the constraints on the cost index (13) are a convex expression shown in (8) due to the wavelet orthogonal decomposition. Therefore, these K control increments can be calculated recursively from the steepest gradient decent algorithm during the interval between k and $k+1$ without a local minimum.

Let j be the iteration step for the optimisation ($j=0, \ldots, T$) at the kth control cycle. T is the number of the discrete times and decided by the optimal terminal conditions. Figure 7 shows the calculating cycle relationship during the control and optimisation process where k is the current control cycle and the total number of control cycles in the path tracking is S. In our tracking case, we choose:

$$N_1 = 1,\ N_2 = L,\ d = 1,\ N_\mu = K$$

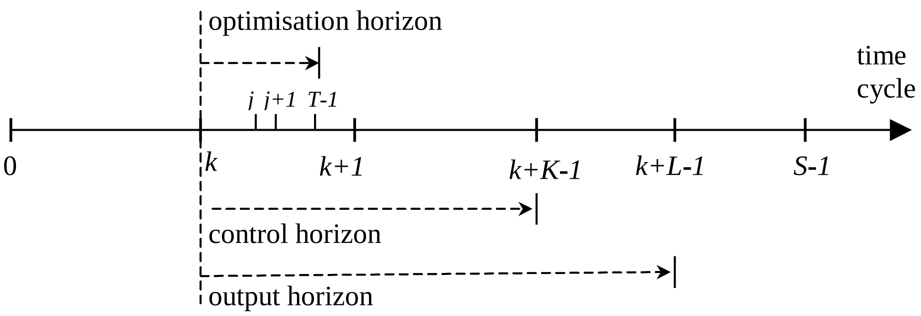


Figure 7 Time step for the predictive control

The control increment $\Delta\mathbf{U}(k)$ can be obtained based on the cost index (13) according to the gradient descent algorithm. Assume that μ is the optimising rate. We have:

$$\Delta\mathbf{U}^j = -\mu\delta J_{\mathbf{U}}^j \tag{14}$$

The cost index change $\delta J_{\mathbf{U}}^j$ is caused by the control increment $\Delta\mathbf{U}(k)$ and the state change $\delta\mathbf{X}_{\mathbf{U}}^j$. It can be represented in (15) by differentiating (13),

$$\delta J_{\mathbf{U}}^j = -\delta\mathbf{X}_{\mathbf{U}}^j\mathbf{E}^j + \mathbf{\Gamma}\Delta\mathbf{U}^j \tag{15}$$

where the state change $\delta\mathbf{X}_{\mathbf{U}}^{j}$ is also caused by control increment *ΔU(k)*. The state-changing gain and control-changing gain are $\mathbf{E}^{j}$ and $\mathbf{\Gamma}$ respectively. They can be expressed as:

$$\delta\mathbf{X}_{\mathbf{U}}^{j} = \begin{bmatrix} \dfrac{\partial \mathbf{x}^{j}(k+1)}{\partial \mathbf{u}^{j}(k)} & \cdots & \dfrac{\partial \mathbf{x}^{j}(k+K)}{\partial \mathbf{u}^{j}(k)} & \cdots & \dfrac{\partial \mathbf{x}^{j}(k+L)}{\partial \mathbf{u}^{j}(k)} \\ 0 & \cdots & \dfrac{\partial \mathbf{x}^{j}(k+K)}{\partial \mathbf{u}^{j}(k+K-1)} & \cdots & \dfrac{\partial \mathbf{x}^{j}(k+L)}{\partial \mathbf{u}^{j}(k+K-1)} \end{bmatrix}$$

$$\mathbf{E}^{j} = \left[\mathbf{x}_{r}(k+1) - \mathbf{x}^{j}(k+1) \quad \cdots \quad \mathbf{x}_{r}(k+L) - \mathbf{x}^{j}(k+L)\right]^{T}$$

$$\mathbf{\Gamma} = diag[\boldsymbol{\lambda}_{1}, \cdots, \boldsymbol{\lambda}_{K}]$$

Substituting (15) into (14), control increments during one control cycle can be expressed as:

$$\Delta\mathbf{U}^{j} = (I + \mu\mathbf{\Gamma})^{-1}\mu\delta\mathbf{X}_{\mathbf{U}}^{j}\mathbf{E}^{j} \tag{16}$$

At each optimisation step *j*, the predicted states $\boldsymbol{x}^{j}(k+i_{1})$ ($i_{1}=1, ..., L$) are produced by the neural network based on the control vector $\boldsymbol{u}^{j-1}(k+i_{2})$ ($i_{2}=0, ..., K-1$). For the first optimisation step $j=1$, $\boldsymbol{u}^{0}(k+i_{2})$ is initialised as $\boldsymbol{u}^{T}(k-1+i_{2})$ (the control vector at the last control cycle). After *T* iterations, an optimal control increment $\Delta\boldsymbol{U}^{*}(k)$ at the *k*th control cycle is produced. The main computation during this optimising procedure is the matrix calculation $\frac{\partial \mathbf{x}(k+i)}{\partial \mathbf{u}(k+l)}$ (*i=1, ..., L; l=0, ..., T-1*). With reference to the neural network model (8), one of the matrix elements is presented as:

$$\frac{\partial x(k+i)}{\partial \alpha(k+l)} = \frac{\partial f_{1}(\mathbf{z}(k+i-1))}{\partial \alpha(k+l)} = \frac{\partial \sum_{m=1}^{r} w_{1m}^{2}\varphi_{d}}{\partial \alpha(k+l)} = \begin{cases} \sum_{m=1}^{r} w_{1m}^{2}\varphi_{d}' 2^{M} & (i = l+1) \\ 0 & (i \neq l+1) \end{cases} \tag{17}$$

The optimal control vector can be obtained by choosing the first vector in $\Delta\boldsymbol{U}^{*}(k)$:

$$\boldsymbol{u}^{*}(k)=\boldsymbol{u}^{*}(k-1)+[1, 0, ..., 0]\Delta\boldsymbol{U}^{*}(k) \tag{18}$$

5. SIMULATION RESULTS [1]

A simulator based on the robot kinematics (1) is developed to validate the proposed approaches. The Gaussian noise with zero mean value is added to the robot state to simulate the estimated state xm(k). As the first stage, we need to collect data of the robot kinematics to train the wavelet neural network. This can be done by employing a PID controller to track the path (see figure 3). Previously a PID control scheme has been used for the real robot in [7] where we estimated the position by an EKF and used the same trajectory generation. With the tedious trials, PID can track the path. However, this process has to be repeated when the robot travels on different road surface, or even on the same road at

[1] One important aspect in modelling is that the system has to be excited enough (the control signal has to be persistently exciting). However, the training data is generated based on the PID controller without any additional excitation noise. Because there is no drift term in the kinematics model (1), the lack of excitation is not a problem. This only holds if a model similar to (1) represents the dynamics of the true robot. The motivation to model the robot and not use (1) is to make the controller adaptable in case of changes in the system, such as increasing the load of the robot. This will lead to an increase in inertia resulting in lower accelerations and decelerations. These effects cannot be modelled in the kinematics model (which only deals with the speed). The modelling may try to incorporate this in the model by introducing a drift term (which is possible according to the general kinematics model in (7)). Without excitation noise, the model obtained along the PID trajectory may not be valid along the generated path, because the model does not have to generalise over a "larger" part of the state space.

different times for reasons of load change or tyre inflation. This is the main reason why we investigate adaptive control for path tracking.

Once the training data is collected, formula (11) is used to train the wavelet neural network until a certain number of the training trials, or the errors between the network output and the simulated position estimation are less than given thresholds. An initial kinematics is built up, which can be used for the NPC control scheme. In the NPC control process, the PID controller is replaced with the NPC controller. The control vector (18) output from the NPC controller is calculated at each control cycle, while on-line training of the network and the optimisation of the cost index are executed within the current control interval (see figure 7).

5.1 Wavelets

The scaling function is shown in figure 4. The dilation and translation of the scaling function $\varphi_{M,n}(t)= 2^M\varphi(2^M t\text{-}n)$ are centred in $n/2^M$. We transform the input and output ranges of the network into $[-1/2, 1/2]$. So, the number of hidden nodes can be determined roughly as $(2^M+1)^p$ and p is the number of inputs to the hidden nodes. For different outputs, p is selected according to the relationship between inputs and outputs described by the kinematics (1). To choose a proper M, a trial and error approach is used as follows [25]:

Step 1: Start from a small *M*.

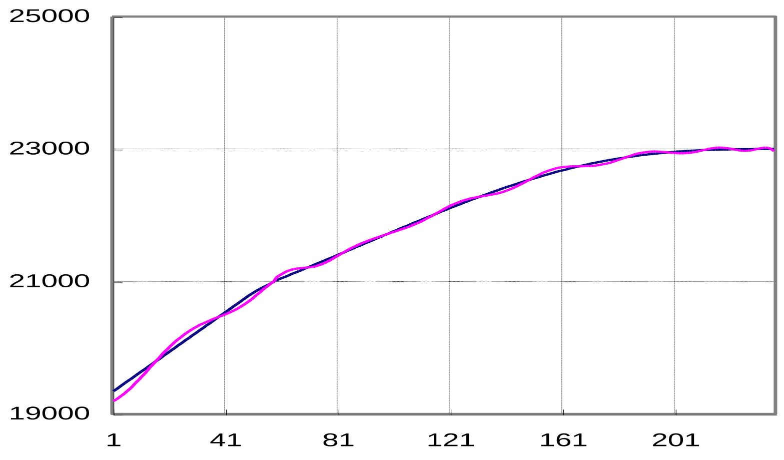
Step 2: Computed the mean square error (*MSE*).

Step 3: Increase *M* by 1 and repeat from *Step 1* if *MSE* is bigger than a threshold.

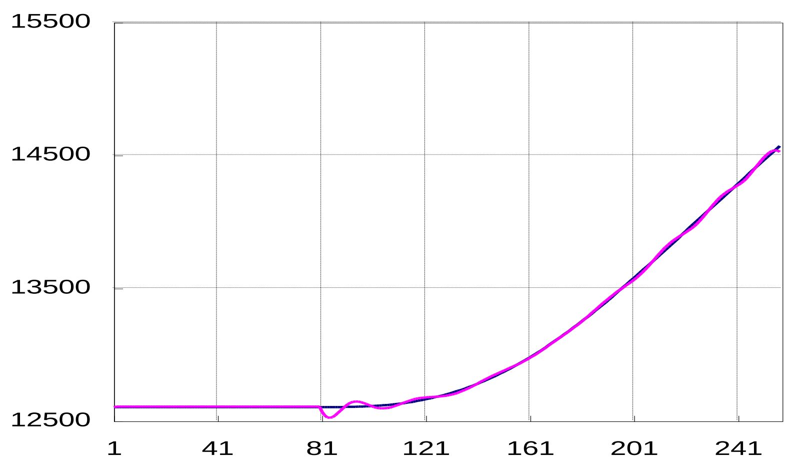
The sparse data in our path tracking case can further decrease the support range N of the network inputs, e.g. the steer angle is changed in a smaller region.

5.2 Modelling Validation

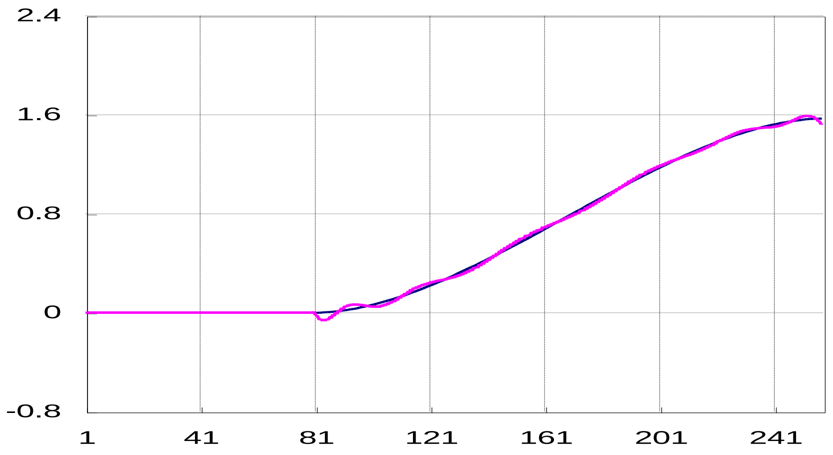
We implemented three path-tracking courses to collect the training data: straight line, turn right, and turn left. 65 hidden nodes were used to cover the range of the robot state. Only the weights in an output layer need to train (see figure 5). A path starting from (19000mm, 12600mm, 0rad) to (23000mm, 14600mm, π/2rad) is used to validate the model of the kinematics. The results are shown in figure 8, where the unit of x, y, and x error is *mm* and of θ is radians. The number on the horizontal axis is motion steps.



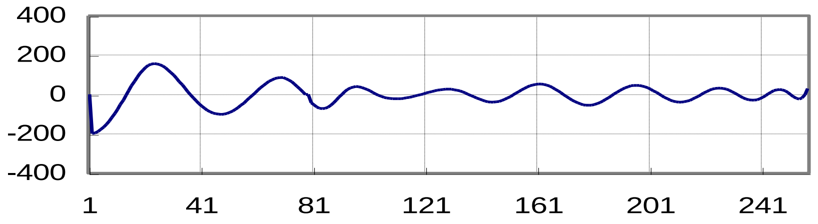
(a) A plot of *x*



(b) A plot of *y*



(c) A plot of θ



(d) A plot of *x* error

Figure 8 Neural modelling: (a) *x*, (b) θ, (c) *y*, and (d) *x* error.

This desired path was produced by the trajectory generation part shown in figure 3, which consists of two parts: a straight line starting from (19000mm, 12600mm, 0rad) to (21000mm, 12600mm, 0rad) and a left turning starting from (21000mm, 12600mm, 0rad) to (23000mm, 14600mm, π/2rad). The solid lines shown in the figure 8 (a)(b)(c) are the results of PID controller, which start to turn left around point (21400mm, 12600mm, 0rad) rather than (21000mm, 12600mm, 0rad) as desired since the PID controller is not a perfect controller. It did not affect our simulation results since the collected data is only used to train the wavelet neural network to model the kinematics. The dash lines in the figure 8 (a)(b)(c) are the modelling results of the wavelet neural network. The data shows that the results of the modelling can approximate to the dash lines. The approximate error of x is shown in the figure 8(d). They fall within a range of (-200mm, 200mm). In the straight-line part, the modelling error of x is relative large comparing with the errors of y and θ since only x increased while y and θ nearly kept constant. In the left turning part, all three variables increased to achieve their desired values.

5.3 Path tracking[2]

The trajectory generation produced a desired trajectory as shown in figure 9, which is the same as the one used in the modelling validation. The PID controller was first implemented to make the robot track this desired trajectory, namely "PID control" in figure 9. Based on the trained wavelet neural network the NPC controller was then implemented and the tracking result was also shown in figure 9, namely "NPC control". In the NPC tracking, we use $L=10, K=10, T=5$.

Although the modelling of the wavelet neural network was based on the data collected from the process of PID control, the NPC control has different performance with the PID control. There is not much difference in the straight-line part where only x error is large (this can be seen from the starting of the turning point). Then PID control turned the robot left with a larger angle than NPC control. This led to the PID control trajectory diverging from the desired path than the NPC control trajectory at the beginning of the turning. Later on, the deviation of the NPC trajectory from the desired path is caused mainly by x error, while the PID control trajectory deviates from the desired path in all three variables.

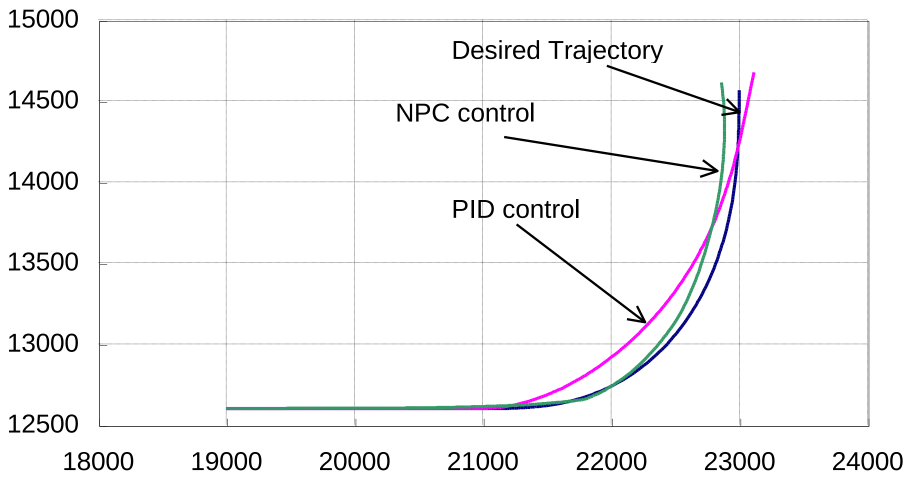


Figure 9 Path tracking

	PID	NPC
Position (x,y) ASE	98.06	57.29
Orientation θ ASE	0.167	0.168

Table 1 Comparison of ASE for PID AND NPC

[2] The general application domains of a MPC are slow systems such as process plants. The primary objective in these cases is not the "optimality" of the control algorithm but the safety. The main advantage of a MPC is that it can take into account arbitrary constraints. Not the first "optimal action" is applied, but the first optimal action that does not violate the constraints within the prediction horizon is applied. In the mobile robot context we can say that the robot is not a point as in the simulation, the true robot has a certain width. This would seem like a more natural motivation for the use of a MPC for mobile robots. Take the trajectories in figure 9 as an example. Suppose that the path is generated to make the robot turn around a corner. The PID result makes the robot move too much along the inside of the trajectory, which may lead to a collision of the robot with the corner. In this case the NPC result did not collide, but only because it followed the path very closely. The collision was not prevented by rejecting all actions that may have resulted in a collision.

In NPC control, maximum errors in *x* and y are 118mm and 48mm respectively. In the PID tracking, maximum errors in x and y are 300.48mm and 51mm. A numerical comparison between PID and NPC controllers through the average squared error (ASE) with reference to the desired trajectory is shown in table 1. The ASE is an average value for whole trajectory. The position (*x*,*y*) ASE is the root of sum of squared error *x* and *y*. The position (*x*,*y*) ASEs in table 1 show that the NPC control trajectory is much closer to the desired path than the PID control trajectory. The orientation ASEs show that average orientation deviations of the two trajectories from the desired are nearly the same in this section of the path.

6. CONCLUSIONS

The neural predictive control scheme proposed in this paper includes path generation, position estimation, and a wavelet neural network based MPC. Benefits resulted from wavelet neural networks will be the convex cost index without the local minima dilemma, appropriate initial value setting of weights, and a constructive approach for the hidden layer of feed-forward neural networks. Simulation results show the ability of approximation of wavelet neural networks to the non-linear kinematics and adaptive path tracking by this wavelet neural network based predictive controller.

This paper shows that adaptive neural network based MPC is one of the solutions for time-variable non-linear systems. The aim is to provide a constructive way to build up the neural networks used for modelling and limit the size of neural networks. Further research should be conducted to investigate the structurally dynamic wavelet neural networks that exploit the properties of wavelet decomposition to choose the wavelets on the fly. It is equivalent to dynamically choosing the hidden nodes in the neural networks based on the measurement of robot states [20]. In this way, the identification of non-linear systems can be preceded on both the model parameters and the model structure. The implementation of this neural predictive control scheme on the real robot will be also conducted in the next stage.

Acknowledgement: The authors wish to thank the anonymous reviewers for their constructive comments and suggestions which improved the quality of this paper.

REFERENCES

[1] R. C. Arkin, Behaviour-Based Robotics, The MIT Press, 1998.

[2] R. Brooks, A Robust Layered Control System for a Mobile Robot, IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, pages 14-23, 1986.

[3] J. Canderle and D. Caldwell, Generalised Predictive Control of Pneumatic Muscle Actuator, European Advanced Robotics Systems Master class and Conference-Robotics 2000, Salford, UK, April 2000.

[4] D. Clarke, C. Mohtadi, and P. Tuffs, Generalised Predictive Control-Part I. The Basic Algorithm, Automatica, Vol. 23, No. 2, pages 137-148, 1987.

[5] J. A. Gangloff and M. F. Mathelin, High Speed Visual Servoing of a 6 DOF Manipulator Using MIMO Predictive Control, Proc. of the 2000 IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, April 2000, pages 3751-3756.

[6] R. Gomez-Sanchez and D. Andina, A Wavelet Neural Network for detection of Signals in Communications, Proc. of SPIE, Vol. 3391, pages 265-272, 1998.

[7] D. Gu, H. Hu, M. Brady, F. Li, Navigation System for Autonomous Mobile Robots at Oxford, Proceedings of International Workshop on Recent Advances in Mobile Robots, Leicester, U.K., pages 24-33, 1-2 July 1997.

[8] S. Haykin, Neural Network - A Comprehensive Foundation, Prentice Hall, 1999.

[9] H. Hu, D. Gu, and M. Brady, A Modular Computing Architecture for Autonomous Robots, Int. Journal of Microprocessors and Microsystems, Vol. 21, No. 6, pages 349-362, March 1998.

[10] Y. Kanayama, and N. Miyaki, Trajectory generation for mobile robots, Proc. of the IEEE Int. Conf. Robotics and Automation, pages 333-340, France, 1985.

[11] M. Martinez, J. S. Senent, and X. Blasco, Generalised Predictive Control Using Genetic Algorithms (GAGPC), Artificial Intelligence, Vol. 11, 1998, pages 355-367.

[12] H. P. Moravec, Sensor Fusion in Certainty Grids for Mobile Robots, AI Magazine, 9(2), pages 61-74, 1988.

[13] K. S. Narendra and K. Parthasarathy, Identification and Control of Dynamical Systems, IEEE Trans. on Neural Network, Vol. 1, No. 1, March 1990, pages 4-27.

[14] W. L., Nelson, Continuous-curvature Paths for Autonomous Vehicle, Proc. of the IEEE Int. Conf. Robotics and Automation, pages 1260-1264, 1989.

[15] Y. C. Pati and P. S. Krishnaprasad, Analysis and Synthesis of Feedforward Neural Networks Using Discrete Affine Wavelet Transformations, IEEE Trans. on Neural Network, Vol. 4, No. 1, Jan. 1993, pages 73-85.

[16] O. Pinchard, A. Liegeois, and F. Pougent, Generalised polar polynomials for vehicle path generation with dynamic constraints, Proc. of the IEEE Int. Conf. Robotics and Automation, pages 915-920, Minneapolis, Minnesota, 1996.

[17] S. Piche, et al, Non-linear Model Predictive Control Using Neural Networks, IEEE Control Systems Magazine, Vol. 20, No. 3, June 2000, pages 53-62.

[18] D. Ramirez, et al, Non-linear MBPC for Mobile Robot Navigation Using Genetic Algorithms, Proceedings of the 1999 IEEE Int. Conf. on Robotics & Automation, Detroit, Michigan, May 1999, pages 2452-2457.

[19] J. B. Rawlings, Tutorial Overview of Model Predictive Control, IEEE Control Systems Magazine, Vol. 20, No. 3, June 2000, pages 38-52.

[20] R. M. Sanner and J. E. Slotine, Structurally Dynamic Wavelet Networks for the Adaptive Control of Uncertain Robotic Systems, Proceedings of the 34th Conf. on Decision & Control, New Orland, LA, Dec. 1995, pages 2460-2467.

[21] B. Song and A. Koivo, Neural Network Model Based Control of a Flexible Link Manipulator, Proc. of the IEEE Int. Conf. on Robotics & Automation, Leuven, Belgium, May 1998, pages 812-817.

[22] H. H. Szu, B. Telfer, and S. Kadambe, Neural Network Adaptive Wavelets for Signal Representation and Classification, Optical Engineering, Vol. 31, No. 9, September 1992, pages 1907-1916.

[23] Y. Tan and R. Keyser, Neural Network Based Adaptive Predictive Control, Advances in Model Based Predictive Control, Edited by D. Clarke, pages 358-369, 1994.

[24] J. M. Zamarreno and P. Vega, Neural Predictive Control: Application to a Highly Non-linear System, Artificial Intelligence, Vol. 12, 1999, pages 149-158.

[25] J. Zhang, G. Walter, Y. Miao, and W. Lee, Wavelet Neural Networks for Function Learning, IEEE Trans. on Signal Processing, Vol. 43, No. 6, June 1995, pages 1485-1495.

[26] Q. Zhang, Using Wavelet Network in Non-parametric Estimation, IEEE Trans. on Neural Network, Vol. 8, No. 2, March 1997, pages 227-236.

[27] G. Walter and J. Zhang, Orthonormal Wavelets with Simple Closed-Form Expressions, IEEE Trans. on Signal Processing, Vol. 46, No. 8, 1998, pages 2248-2251.

[28] J. G. Ortega, and E. F. Camacho, Neural Network GPC for Mobile Robot Path Tracking, Robotics and Computer-Integrated manufacturing, Vol. 11, pages 271-278, 1994.