Pergamon

Control Eng. Practice, Vol. 4, No.9 pp 1221-1230,1996

0967-0661/96 $15.00 + 0.00

PII:S0967-0661(96)00128-1

LEARNING FEEDFORWARD CONTROLLER FOR A MOBILE ROBOT VEHICLE

J.G. Starrenburg, W.T.C. van Luenen**, W. Oelen***, J. van Amerongen***

** Department of Electrical Engineering and Mechatronics Research Centre Twente, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands (amn@rt.el.utwente.nl)*
***Unilever Research Laboratory, P.O. Box 114, 3130 AC Vlaardingen, The Netherlands*
****CMG Noord Nederland, Steenhouwerskade 8, 9718 DA Groningen, The Netherlands*

(Received April 1995; in final form May 1996)

Abstract: This paper describes the design and realisation of an on-line learning pose-tracking controller for a three-wheeled mobile robot vehicle. The controller consists of two components. The first is a constant-gain feedback component, designed on the basis of a second-order model. The second is a learning feedforward component, containing a single-layer neural network, that generates a control contribution on the basis of the desired trajectory of the vehicle. The neural network uses B-spline basis functions, enabling a computationally fast implementation and fast learning. The resulting control system is able to correct for errors due to parameter mismatches and classes of structural errors in the model used for the controller design. After sufficient learning, an existing static gain controller designed on the basis of an extensive model has been outperformed in terms of tracking accuracy.

Keywords: learning, feedforward, control, neural, intelligent, automated guided vehicles, autonomous, mobile robots, robot

1. INTRODUCTION

Conventional approaches to designing a controller are based on a mathematical model of the system to be controlled. In some situations, this system may have time-varying parameters, such as the payload mass or friction of a robot manipulator. In this case, the use of adaptive control systems enables considerable improvements to be made compared to conventional controllers (Berghuis, 1993). Nevertheless, the controller is based on a mathematical description of the process.

In many practical systems, parts of the process model are hard to describe mathematically. Friction effects are an example of such a phenomenon. Coulomb friction, for instance, may depend on unmeasured parameters, like payload, oil temperature in the bearings or wear. In addition, not only the amplitude but also the shape of a function describing the effect may change. Appropriate compensation for such effects on the basis of an accurate mathematical description is either impossible or involves extensive modelling and identification efforts.

Recently, neural networks like the multilayer perceptron (Rumelhart et al., 1986) and radial basis function networks (Poggio and Girosi, 1989), have been introduced in the control field. These networks are able to approximate a non-linear continuous function using large numbers of identical basis functions. Their application in the control field has triggered the suggestion that these networks could be used for approximating the unknown parts of the process model.

The approach taken in this paper combines the robustness and easy design of a PD-feedback controller with a learning control strategy for improved tracking performance. A feedback controller is designed on the basis of a simplified model. This controller must have a sufficiently large stability margin to be robust for the possible shortcomings in the a priori model. This

Fig. 1. The Mobile Autonomous Robot Twente

results in lower feedback gains, i.e. larger tracking errors. A learning feedforward part is added which gradually reduces these tracking errors. In this way, a controller, obtained with limited modelling and design effort, is provided with a learning component which is able to enhance performance.

2. THE PROCESS: A MOBILE ROBOT

The process to which the learning controller is applied is the Mobile Autonomous Robot Twente (MART). The MART is the result of a mechatronic design project in which a new, flexible assembly-line concept is worked out. Multiple mobile robots can collect parts from supply stations and assemble these parts, if necessary during driving. (Schipper, 1991) Within this case study, the tracking control of the position and orientation of the vehicle will be considered. The vehicle has three wheels, two of which are driven independently. The third is a castor wheel, that is able to rotate freely. The vehicle consists of an upper frame and a lower frame connected by air springs and dampers to obtain a proper suspension for the manipulator. (Graaf, et al., 1993)

2.1. Coordinate definitions

The geometry of the vehicle and the coordinate definitions for the vehicle moving in a plane, are shown in Figure 2. Figure 2a shows a schematic top view of the vehicle, consisting of the outline and the three wheels, including the centre of gravity. In order to specify the motion of the vehicle, it is convenient to define the *pose* of the vehicle as the position together with the orientation. Figure 2b shows the pose p of the vehicle in coordinates of a world-fixed frame $p(x_w, y_w, j_w)$, together with linear velocity v (in m s^{-1}) and the angular velocity w (in rad s^{-1}). Figure 2c shows the definition of a coordinate frame fixed to the vehicle. The subscript w indicates a representation in world-fixed coordinates. The subscript veh is used for vehicle coordinates.

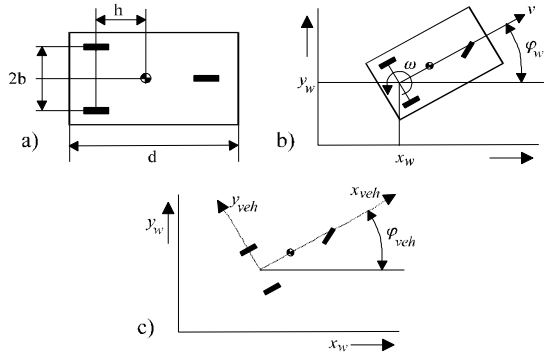

Fig.2. Vehicle geometry and coordinate definitions

When moving along a trajectory, a path generator provides the vehicle with a reference trajectory in world-fixed coordinates. The reference trajectory consists of the desired pose $p_{ref} = [x_{ref}, y_{ref}, j_{ref}]$, velocities $[v_{ref}, w_{ref}]$ and accelerations $[\dot{v}_{ref}, \dot{w}_{ref}]$. The actual pose p_{meas} and the actual velocities are measured by a measurement system that combines intermittent accurate position fixes with odometry (Oelen, 1995). The difference between the reference and measured poses is the *pose error*: $\Delta p = p_{ref} - p_{meas}$. The pose error can be expressed in world coordinates as well as vehicle coordinates. Representing the pose error in vehicle coordinates is most convenient, as Δx_{veh} equals the error in the driving direction, Δy_{veh} equals the lateral error, and $\Delta j_{veh} = \Delta j_w$ is the orientation error.

2.2. A-priori model

For the design of the control system, a simple model of the vehicle is used. The model describes the dynamic behaviour of the vehicle in two dimensions. The following physical properties are assumed to be known:

$M = 500\,[Kg]$	the total mass of the vehicle
$J = 60\,[Kgm^2]$	moment of inertia around centre of mass
$b = 0.30\,[m]$	half the distance between the driving wheels (fig. 2)
$h = 0.42\,[m]$	the distance of the wheel axis with respect to the centre of mass (fig. 2)
$d = 1.4\,[m]$	the length of the vehicle (fig. 2)

A dynamic model based on this information is given by:

$$\dot{v} = \frac{F_r + F_l}{M} + hw^2$$
$$\dot{w} = \frac{b(F_r - F_l) - Mhw\,v}{J + Mh^2} \qquad (1)$$

with F_l the actuator force on the left wheel and F_r the actuator force on the right wheel, in [N]. F_l and F_r are the control input signals.

Note that this model does not include details of the actuators, friction effects, the presence of the castor and the presence of an upper and a lower frame in the vehicle.

2.3. Constraints on the implementation

The specifications for the MART provide some practical constraints on the implementation of a controller. The constraints with respect to the vehicle motion are (Oelen, 1995):

maximum linear velocity: $v_{\max} = 1.0$ [m/s]
maximum linear acceleration: $\dot{v}_{\max} = 1.0$ [m/s^2]
maximum angular velocity: $w_{\max} = 1.0$ [rad/s]
maximum angular acceleration: $\dot{w}_{\max} = 1.0$ [rad/s^2]

The actuators are sufficiently powerful to achieve these maxima.

The *position tracking error* of the vehicle is defined as:

$$\Delta r = \sqrt{\Delta x^2 + \Delta y^2}$$

where Δx and Δy are the position error components, either in world coordinates or in vehicle coordinates. After learning, the learning feedforward controller must satisfy the following requirements on the tracking errors.

during driving at high speed: $\Delta r \le 100$ [mm]
during slow driving: $\Delta r \le 10$ [mm]
orientation tracking error: $\Delta j \le 0.02$ [rad]

The positioning accuracy of the vehicle at the final position is defined as the position error immediately after arrival. The required final position accuracy: $\Delta r \le$ 10 [mm].

The control signals (wheel forces) should be continuous functions of time with a bounded time derivative. This constraint is to ensure a smooth vehicle motion, thereby enabling the manipulator to work accurately during driving.

Because of the other tasks to be carried out, the control algorithm must share a single T800 transputer with the path generator and the odometry. The required sampling frequency is estimated to be equal to 200 [Hz]. The control algorithm may occupy up to 700 Kbytes of computer memory.

3. CONTROLLER DESIGN

The design of the control system will be divided into two main parts. First, the model-based part will be designed on the basis of the a priori available model. Second, the design of the learning feedforward controller is described. Figure 3, which shows the resulting controller scheme, should be used as a reference throughout this section.

3.1. The model-based controller

As a result of the wheel configuration of the vehicle, no motion is possible in the lateral direction. This is called a "non-holonomic" constraint. A solution to this problem (Oelen and Van Amerongen, 1994) can be found by introducing a *corrected orientation error* Δz, defined by

$$\Delta z = \Delta y_{veh} + a\,\Delta j_{veh} \cdot sign(v_{ref}) \qquad (2)$$

where a is a positive constant which can be chosen equal to the inverse of the vehicle length. (Thus, $a = 1/d \approx 1$). Controlling the vehicle such that $\Delta z \to 0$ will reduce the lateral error while driving along the trajectory. The vehicle requires a fixed covered distance, determined by a and independent of the vehicle's velocity, to reduce the lateral error by a certain factor. This property is called the "geometric convergence" of the lateral error (Oelen and Van Amerongen, 1994).

The feedback controller has been designed by considering the model given in eq.(1). As the errors of the model-based controller will be reduced by the learning component, the demands on feedback tracking performance are not high. For this reason, the feedback parameters have been chosen sufficiently small to obtain a safe stability margin. Also, an additional simplification has been allowed by disregarding the coupling between the translating and rotating motion of the vehicle, resulting in

$$\begin{aligned} F_r + F_l &\approx M\dot{v} \\ F_r - F_l &\approx \frac{(J + Mh^2)\dot{w}}{b} = J'\dot{w} \end{aligned} \qquad (3)$$

This allows translating and rotating motion to be considered as two independent linear second-order systems. For each of them, position and velocity feedback can be applied. One feedback controller uses the error in driving direction, Δx_{veh}. The other uses the corrected orientation error Δz. The controller equations, where the subscript fb indicates feedback contributions to the actuator forces, are given by:

$$\begin{aligned} F_{r,fb} + F_{l,fb} &= K_{px}\Delta x + K_{dx}\Delta\dot{x} \\ F_{r,fb} - F_{l,fb} &= K_{pj}\Delta z + K_{dj}\Delta\dot{z} \end{aligned} \qquad (4)$$

By substituting (4) in (3), the following closed-loop transfer functions are obtained.

$$\begin{aligned} \frac{x}{x_{ref}} &= \frac{K_{px} + sK_{dx}}{Ms^2 + sK_{dx} + K_{px}} = \frac{M^{-1}\left(K_{dx}s + K_{px}\right)}{s^2 + 2z_x w_x s + w_x^2} \\ \frac{j}{j_{ref}} &= \frac{K_{pj} + sK_{dj}}{J's^2 + sK_{dj} + K_{pj}} = \frac{J'^{-1}\left(K_{dj}s + K_{pj}\right)}{s^2 + 2z_j w_j s + w_j^2} \end{aligned} \qquad (5)$$

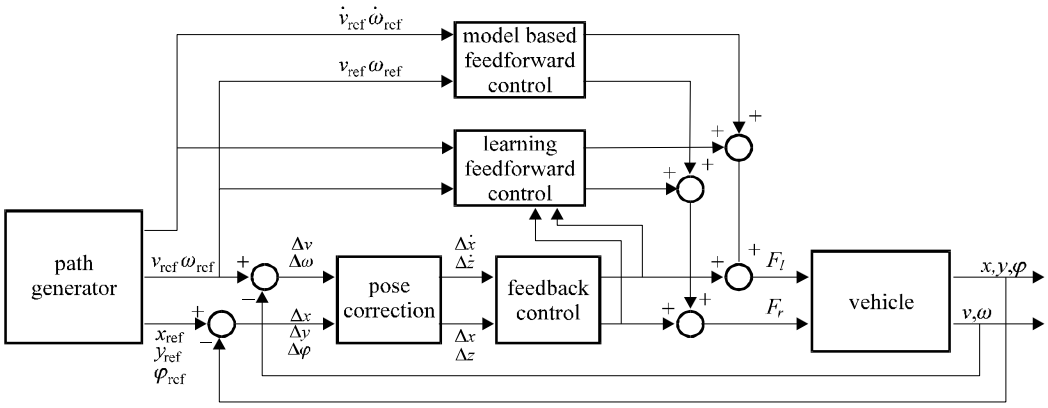

Fig.3.Learning feedforward control combined with model-based control

In order to stay below the resonance frequency of the vehicle suspension of 3[*Hz*], the natural frequencies w_x and w_j are chosen approximately ten times lower. The dampings z_x and z_j are chosen to be at least 0.7. The following values have been taken.

$K_{px} = 2\cdot 10^3[N/m]$ $\qquad$ $K_{dx} = 2\cdot 10^3[Ns/m]$

$K_{pj} = 2\cdot 10^3[N]$ $\qquad$ $K_{dj} = 2\cdot 10^3[Ns]$

The resulting pole-zero images indicate a stable, well-damped closed-loop behaviour for both systems.

The model used for design of the feedback can also be used to extend the control law with a model-based feedforward component. Equation (1) is rearranged such that F_l and F_r are given as explicit functions of the (reference) path, yielding the feedforward control law given below. The subscript ff indicates feedforward components of the actuator forces.

$$F_{l,ff} = \frac{1}{2}\left[M\dot{v}_{ref} - Mhw_{ref}{}^2 \frac{(J + Mh^2)\dot{w}_{ref} + Mhw_{ref}v_{ref}}{b} \right]$$
$$F_{r,ff} = \frac{1}{2}\left[M\dot{v}_{ref} - Mhw_{ref}{}^2 + \frac{(J + Mh^2)\dot{w}_{ref} + Mhw_{ref}v_{ref}}{b} \right] \quad (6)$$

3.2. Design of the learning control component

In Section 3.2.1, the reason for using feedforward learning is explained. In 3.2.2, convenient feedforward inputs are deduced. Section 3.2.3 presents the learning criterion. Sections 3.2.4 and 3.2.5 motivate the choice of the neural network and describe the chosen network.

3.2.1. Why feedforward learning?

In a tracking control system, learning can take place either inside the feedback loop (for instance by adaptation of the feedback gains) or outside the feedback loop, resulting in feedforward learning. As the stability of the controlled system is determined by the feedback loop, feedback learning may endanger stability. When the structure of a model of the process is known, stable feedback learning or adaptation mechanisms can be constructed. In this case study, incomplete knowledge of the structure of the plant has been postulated. For this reason the feedforward approach has been chosen.

3.2.2. Inputs and outputs

The model-based feedforward given in eq.(6) already indicates that the required control signals depend strongly upon desired velocities and accelerations. Friction effects are also a function of the vehicle velocities. The dynamic behaviour of the vehicle is independent of the vehicle's position and orientation. This is expressed in the following equation:

$$F_{ff} = f\left(v_{ref}, w_{ref}, \dot{v}_{ref}, \dot{w}_{ref}\right) \quad (7)$$

The learning feedforward will have two outputs, corresponding to the force contributions for the two actuated wheels. Note that by explicitly applying the first two derivatives of the reference pose, a static mapping between the inputs and outputs of the feedforward component results.

3.2.3. Learning-error measure

Ideally, the total feedforward structure should eventually contain the mapping from the desired velocities and accelerations to the corresponding control signals, such that there are no tracking errors if these control signals are applied to the process. As long as this mapping is not perfect, tracking errors will occur which will be compensated for by the feedback controller. Therefore, the feedback control signals have been used as the output error measure for the learning feedforward structure. When learning proceeds sufficiently slowly, feedback signals caused by disturbances or noise will have no significant effect on the learning process.

3.2.4. Choice of network

The chosen inputs and outputs allow the use of a static structure for the learning feedforward controller. The (possibly non-linear) mapping from
reference velocities and accelerations to forces will have to be learned. Recall that the constraints on this design choice (on sample frequency, memory usage and smoothness of the mapping) are mentioned in Section 2.3 The following mappings have been considered.

- A *lookup table* can be used to implement a non-linear mapping. The output signals will be discontinuous functions of the inputs, which is not satisfying.
- A *single-layer spline-network* (Brown and Harris, 1991) consists of a single layer with many units, each containing a B-spline basis function. The network output is a weighted combination of these basis functions. Spline functions are polynomial functions, which can be evaluated efficiently. The order of the spline functions determines the smoothness of the mapping. Splines have a restricted domain on which they differ from zero. For learning or reading the network, only the parameters corresponding to splines with non-zero contribution need to be addressed, resulting in an efficient implementation.
- A *radial basis function network* (RBF) (Poggio and Girosi, 1989) consists of a single layer with many units, each containing radial basis functions, e.g. Gaussian functions. The structure resembles the above-mentioned spline network, but the basis functions used are different. Gaussian functions take more time to evaluate than spline-functions and do not have a restricted domain on which they differ from zero. This results in a much larger number of parameters to be addressed for learning or reading the network. The properties of Gaussian functions enable the proof of certain network properties, whereas spline functions prohibit this.
- A *multilayer perceptron* (MLP), (Rumelhart *et al*., 1986) significantly reduces memory usage compared to the previous alternatives. On the other hand, MLP's learn *very* slowly compared to single-layer networks and give a less accurate mapping in terms of mean-squared output error (Van Luenen, 1993). Therefore, this option has been rejected.

The sampling frequency constraint requires a fast implementation, indicating that the use of the single-layer spline network is to be preferred over an RBF network.

3.2.5. Single-layer spline network

A single-layer spline network can be used to realise a static mapping between k inputs $x_1 \ldots x_k$ and a single output y, on a bounded domain of the input space. This is achieved by placing a finite number of basis functions, B-splines, on this domain. The desired mapping is represented as a linear combination of these basis functions.

An n-th order B-spline function consists of pieces of (n-1)th order polynomials, such that the resulting function is (n-1) times differentiable; a linear combination of n-th order splines is also (n-1) times differentiable. Figure 4 shows examples of one-dimensional B-spline functions. A spline function differs from zero over a finite interval.

On each input axis, one-dimensional spline functions can be defined. A grid g_{x_i} is specified on each input axis x_i. The number of grid points specified on input axis x_i is indicated by n_{x_i}. A possible grid definition with corresponding splines is shown in Figure 5 for second-order splines and $n_x = 6$. Multidimensional spline functions can be constructed by mutually multiplying the spline functions defined on the different axes. These are indexed $b_{i_1 \ldots i_k}(\underline{x})$. The following property holds for all $\underline{x} = [x_1 \cdots x_k]$ within the domain of the mapping

$$\sum_{\substack{0 \le i_1 < n_{x_1} \\ \cdots \\ 0 \le i_k < n_{x_k}}} b_{i_1 \cdots i_k}(\underline{x}) \equiv 1 \qquad (8)$$

Each of these multi-dimensional functions can be assigned a *weight* $w_{i_1 \ldots i_k}$. The output of the network is defined by

$$y(\underline{x}) = \sum_{\substack{0 \le i_1 < n_{x_1} \\ \cdots \\ 0 \le i_k < n_{x_k}}} b_{i_1 \cdots i_k}(\underline{x}) \cdot w_{i_1 \cdots i_k} \qquad (9)$$

By assigning the appropriate values to the weights, a desired target relationship between $\underline{x}$ and y can be approximated. If the number of basis functions is increased, this approximation will be more accurate. Consider an input $\underline{x}$ together with a desired output $y_t(\underline{x})$. When the weights are adjusted according to the following learning rule, the actual output will approach the desired output.

$$w_{i_1 \cdots i_k, new} = w_{i_1 \cdots i_k, old} + g \cdot \{y_t(\underline{x}) - y(\underline{x})\}\, b_{i_1 \cdots i_k}(\underline{x}) \qquad (10)$$

with *learning rate* g satisfying $0 < g < 2$. Only a limited number of the $b_{i_1 \cdots i_k}(\underline{x})$ are nonzero for a given input $\underline{x}$. Only those weights corresponding to nonzero basis functions need to be adjusted every learning step.

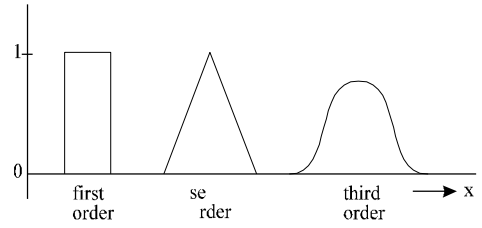

Fig.4. examples of spline functions

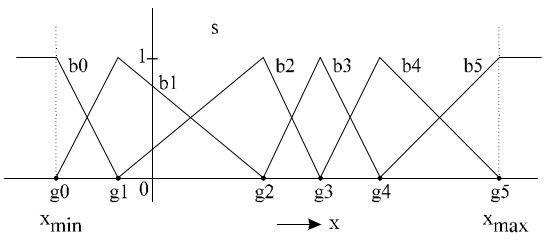

Fig.5. example grid with corresponding splines

This computational efficiency is a significant advantage of the use of B-splines over radial basis functions.

3.2.6. Single-layer spline network for the MART

To obtain continuous control signals with bounded time derivatives, at least second-order splines are required. Higher-order interpolation (and hence additional smoothness) can be obtained at the cost of extra computational efforts. A decision was made in this work to experiment with second- and third-order spline functions.

As the learning controller structure for the MART has two outputs, two single-layer spline networks have been applied in parallel. The two networks can have the same basis function grid definition. This results in a fast application, as the basis functions have to be evaluated only once. To calculate the B-splines, the recursive algorithm mentioned by Brown and Harris (1991) has been implemented, given by:

$$b_{i,j}(x)=\frac{x-g_{i-j}}{g_{i-1}-g_{i-j}}b_{i-1,j-1}(x)+\frac{g_i-x}{g_i-g_{i-j+1}}b_{i,j-1}(x) \quad (j>1)$$
$$b_{i,1}(x)=\begin{cases}1 & \text{if } x\in[g_i,g_{i+1}>\\ 0 & \text{otherwise}\end{cases} \qquad (11)$$

where $b_{i,j}(x)$ is the i-th spline function on input axis x, of order j (j≥1). Note that the computation of higher-order splines (j>1) requires additional grid points outside the border values of x.

The memory requirement of the network is determined by the number of B-splines used. This is a rather arbitrary choice, since knowledge about the mapping that is to be learned is considered unknown. Recall from section 2.3 that 700 Kbytes of memory is available for the controller software. In the experiments, two network sizes have been tested: a small network with 9 B-spline functions per input variable, and a larger network with 19 B-spline functions per input. Due to restrictions in the path-generator algorithm, only positive linear velocities could be obtained, implying that the number of basis functions for this input could be reduced to 5 for the small and 10 for large network. Hence, the total size of the small network became $5*9^3$ = 3645 spline functions. Since the controller has two outputs, the total number of spline weights to be stored equals 2*3645=7290. This results in 29 Kbytes of memory usage if a 32-bit floating-point number representation is used. The large network contains $2*19^3*10$=137.180 spline weights, requiring 536 Kbytes.

The B-splines are placed on the input space according to a predefined grid which is shown in Figure 6. Because the placement could be critical for the small network, it was not taken as homogeneous. Since the specifications require a higher accuracy for low velocities of the vehicle, the placement for velocities has been changed accordingly.

The feedback signal is used as the measure of learning error, meaning that $y_t(\underline{x})-y(\underline{x})$ in eq.(10) is replaced by the feedback signal $u_{fb}(t)$. If this learning rule is applied every sampling interval, an increase in the sampling frequency of the controller will imply a proportional increase in the effective learning rate. Therefore, a sampling-frequency-independent learning rate $x\ [s^{-1}]$ is introduced, and the learning rule becomes:

$$w_{i_1\cdots i_k,new}=w_{i_1\cdots i_k,old}+(x\cdot\Delta t)\cdot u_{fb}\cdot b_{i_1\cdots i_k}(\underline{x}) \quad (12)$$

where Δt is the sampling interval in seconds. An appropriate value for ξ depends on the network size, as ξ can be interpreted as the number of weights that can be learned per output per second. Appropriate values for ξ have been determined by trial and error, resulting in ξ=4 for the small network and ξ=20 for the large network.

4. SIMULATION RESULTS

The control scheme as proposed in the previous section has been fine-tuned in simulation. A simulation model has been obtained by extending eq(1) with friction effects on the two driven wheels. The friction on the castor wheel has not been included. The friction force F_{fr} is assumed to consist of viscous friction (linear in the velocity) and Coulomb friction (depending on the sign of the velocity), given by

$$F_{fr,l}=20\cdot(v-bw)+40\cdot sign(v-bw)$$
$$F_{fr,r}=20\cdot(v+bw)+40\cdot sign(v+bw) \qquad (13)$$

Note that $v-bw$ and $v+bw$ represent the velocities of the left and right wheels (see Figure 2).

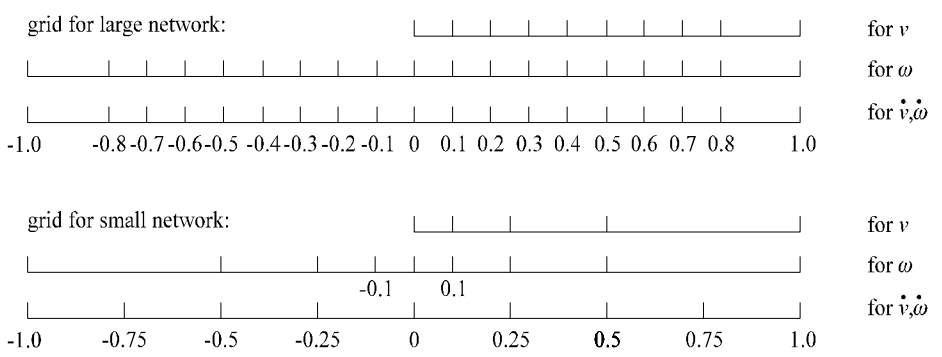

Fig.6. grid definition for both networks

A series of six closed trajectories has been used to test the motion control of the vehicle. The paths are closed in order to simplify later experiments on the real set-up. The trajectories are shown in Figure 7. The dots along the trajectories indicate time instances t_n, such that $t_0 = 0$, t_f is the total time in which the trajectory is covered and $t_n = t_f \cdot n/30$. As a result, the distance between two successive points provides an indication of the velocity along the corresponding part of the trajectory. The average velocities with which the trajectories are covered are as high as the restrictions on velocities and accelerations allow.

To investigate the learning ability of the controller, in addition to the wheel friction, parameter mismatches have been included in the simulation model:

simulation model:		controller design model:
$M = 400[kg]$		$M = 500[kg]$
$J = 40[kgm^2]$	instead of	$J = 60[kgm^2]$
$h = 0.20[m]$		$h = 0.42[m]$

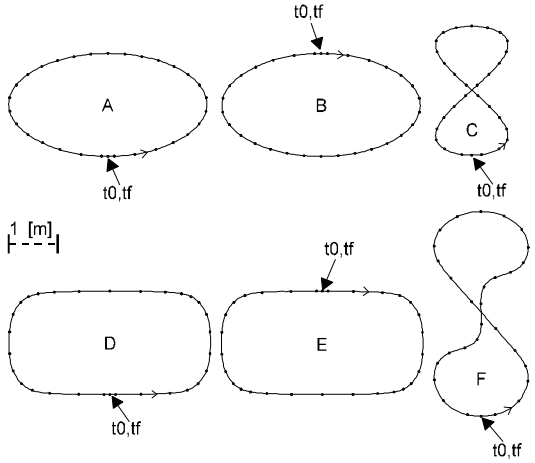

Fig.7.Test trajectories

Experiments investigating the influence of network size and spline order were performed both in simulations and on the actual setup. The results are similar in both cases. A single simulation will be presented for later comparison to measurements obtained on the real setup.

Starting with zero initialisation of the small network, trajectory F has been covered repeatedly. Figure 8 shows the resulting position-tracking error for the small network, using second-order splines, during the 1st, 3rd and 10th coverages of F. The vertical line indicates the end of the time for covering the trajectory. Figure 9 shows the maximum and average position tracking errors along F as a function of the number of times the path has been covered.

Figure 9 shows a drastic decrease in tracking errors, due to learning. The position error upon arrival is finally within the specified boundary of 10 mm. After arrival, the nonzero position error causes the learning controller to integrate the feedback control signal, due to which the error in driving direction approaches zero. Because learning is slow, the integration effect does not produce overshoot. The lateral error cannot be influenced by this effect due to the non-holonomic motion constraint of the vehicle.

The results indicate that the proposed learning controller is able to compensate for the effects of friction and parameter mismatches. Starrenburg (1993) showed that the learning behaviour does not suffer noticeably from the presence of a considerable amount of sensor noise.

Remark: The simulation experiments revealed a boundary condition problem in the path generator, resulting in improper reference signals at departure. This caused minor (5 mm.) positioning errors just after departure. The learning controller gradually decreased these errors as well.

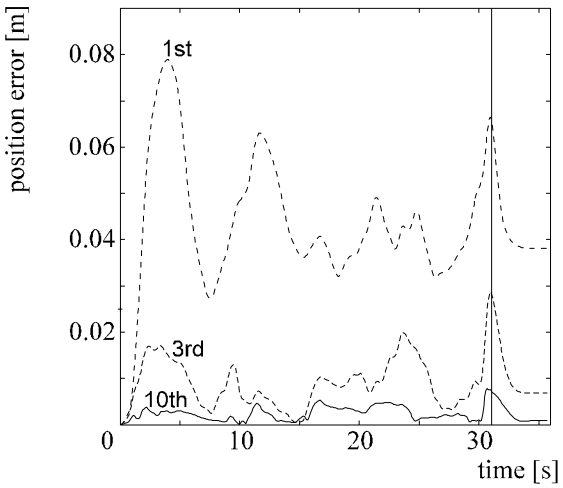

Fig.8. Simulated tracking errors along F

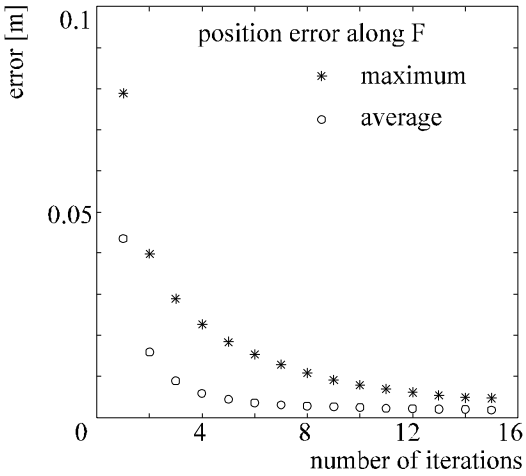

Fig.9. Simulated learning effect along F

This illustrates that the use of a learning controller may include the risk of compensating for unknown aspects of the system, resulting in a working solution but without providing an insight about what is actually learned.

5. EXPERIMENTAL RESULTS

This section gives a summary of the experiments on the actual MART. For a more detailed description, the reader is referred to (Van Luenen, 1993) and (Starrenburg, 1993).

The learning behaviour during repeated coverage of trajectory F was considered using the small and the large network with second-order splines (Figure 10 and 11).

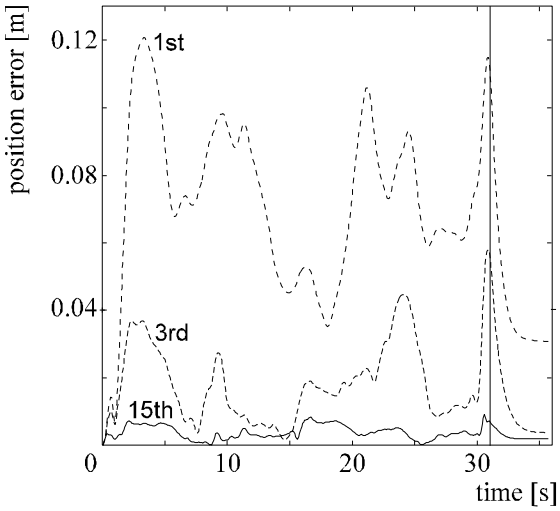

Fig.10. Experimental results while repeatedly covering F with the learning controller containing a small network

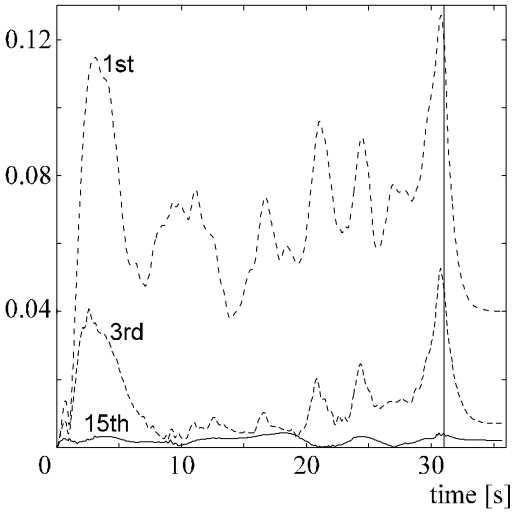

Fig.11. Experimental results while repeatedly covering F with the learning controller containing a large network

The results show that the learning controller indeed reduces the tracking errors significantly during learning. Comparing Figure 8 to Figure 10 shows that the errors during the first trial on the actual vehicle are larger than in simulation, while the shape of the error curve is similar. This indicates that the dynamic behaviour of the actual vehicle is similar to the dynamic behaviour of the simulation model; the friction parameters in the simulation model apparently were too small.

To compare the two networks, the errors obtained during the 15th trial for both the small and the large network have been compared. The errors of the large network are roughly a factor of two smaller than the errors of the small network. This confirms the intuitive impression that the accuracy of the network is proportional to the number of applied basis functions per input. However, it also shows that the number of basis functions increases rapidly if considerable increases in accuracy are required.

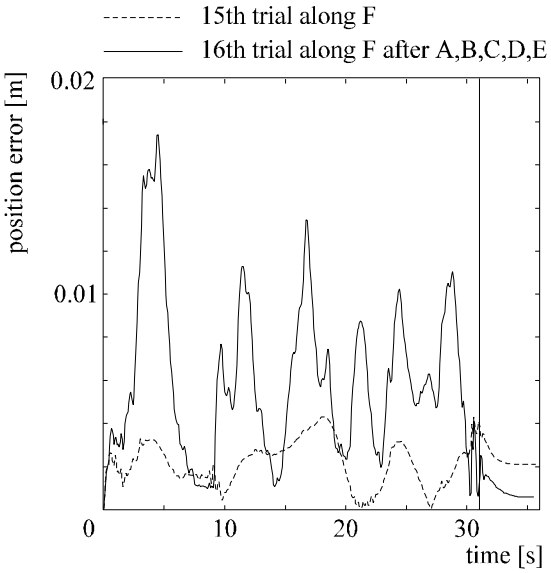

Fig.12. Experimental results demonstrating "forgetting"

The next experiments consider 'generalisation' and 'forgetting' phenomena. In the first experiment, the tracking error was recorded while covering F for the 15th time. Subsequently, trajectories A through E were driven and trajectory F was driven once again while recording the errors. The results for the large network are shown in Figure 12. These results indicate that by driving along various trajectories, a loss in accuracy may be obtained due to 'forgetting'. However, the resulting tracking error doesnot exceed 18 mm. The small network resulted in an error of less than 45 mm. Hence, the reduction of the 'forgetting' effect obtained with a large network is considerable.

In a second experiment, trajectory F is covered three times, once with zero initial values, the second time after having gained experience by driving the paths A through E and the third time after having driven A through E three times. Figure 13 shows the tracking errors along F. It shows that the experience from covering trajectories A through E is 'generalised' and improves the result on trajectory F.

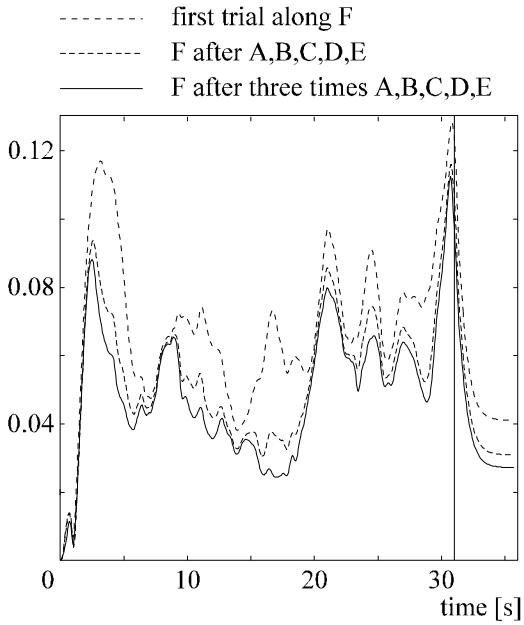

Fig.13. Experimental results demonstrating "generalisation"

The influence of the order of the spline function used in the network on the controller performance has also been tested. Second- and third-order B-splines have been compared. Experiments showed that the order did not significantly influence either the accuracy obtained or the behaviour of the vehicle during tracking. The influence on the achievable sampling rate was significant; with second-order splines 340 Hz. could be obtained, while the use of third-order splines reduced this to 220 Hz. An inspection of the control signals gave no evidence that the third-order splines yielded a smoother control signal. However, this inspection was hampered by the fact that, due to computer restrictions, only one out of 25 samples was available for inspection.

In order to relate the performance of the learning controller to other control approaches, a comparison has been made with an existing feedback controller based on a detailed model of the vehicle (Oelen and Van Amerongen, 1994). For this experiment, both controllers have covered trajectory F. The tracking error displayed for the learning controller was obtained after 15 trials along the trajectory. Figure 14 shows that the learning controller is able to outperform the feedback controller, approximately by a factor of 2, after sufficient exercise. Comparing Figure 12 with Figure 14 reveals that the feedback controller still beats the learning controller, approximately by a factor of 2, when arbitrary paths are covered.

Remark: During the experiments it was observed that the driving performance of the learning controller was very smooth compared to the feedback controller, due to the lower closed-loop stiffness of the former. For the MART, smooth driving behaviour

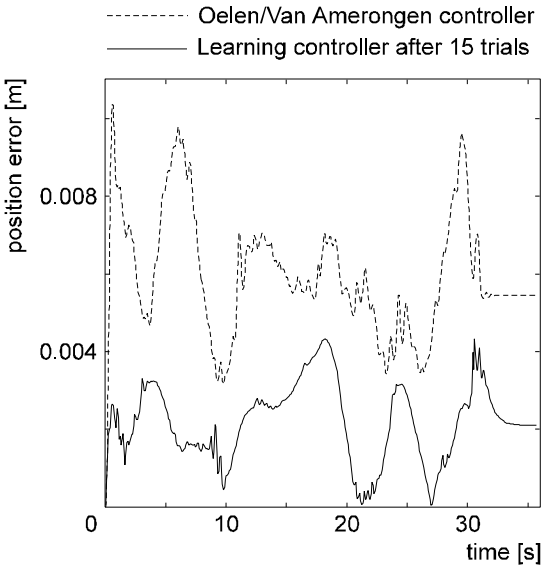

Fig.14. Experimental comparison with a feedback controller based on a detailed model

is advantageous for the performance of the manipulator on top of the vehicle.

6. CONCLUSIONS

This case study demonstrates the feasibility of an on-line learning control system using neural-network concepts. The controller design was based on a model of the vehicle that was known to contain structural and parameter errors, but provided a stable feedback controller.

The learning feedforward controller was tested in simulation against a more detailed model to investigate its robustness against structural errors due to non-linear friction, parameter errors and noise. The simulations made plausible that relatively small networks, containing spline functions, are able to provide an effective compensation for these errors. The experiments on the real set-up confirm this.

The advantage of the learning feedforward controller described in this paper, is that with little modelling a surprisingly good controller can be obtained. However, a consequence of this approach is a lower closed-loop stiffness, as the feedback component of the controller is designed on the basis of an inaccurate model. This turns out to be advantageous for this application, as it results in smooth driving behaviour.

A single-layer spline network is computationally efficient compared to radial basis function networks or multilayered networks. This allows a high sampling frequency in a digital implementation. However, if the desired accuracy of the spline network increases, the memory requirements increase drastically.

Experiments revealed that the use of higher-order splines noticeably improved neither the tracking accuracy nor the smoothness of the control signals. The

achievable sampling frequency decreases significantly with increasing spline-order.

Presumably, the method can be utilised to enhance the performance of any feedback tracking controller in cases where the derivatives of the reference outputs are explicitly available. Therefore, in a future experiment the learning feedforward controller presented here will be combined with a feedback controller, designed on the basis of full a priori knowledge.

7. REFERENCES

Berghuis, H. (1993), *Model-based robot control from theory to practice*, PhD Thesis, Department of Electrical Engineering, University of Twente, Enschede, the Netherlands

Brown, M. and C.J. Harris (1991), *A nonlinear adaptive controller: a comparison between fuzzy logic control and neurocontrol*, IMA Journal of Math. Control and Information, Vol. 8, pp. 239-265

Graaf, A.J. de, M.P. Koster, J.M. Nauta, W. Oelen, D. Schipper, H.G. Tillema (1993), *MART, an overview of the Mobile Autonomous Robot Twente project*, Memoranda Informatica 93-12, Department of Computer Science, University of Twente, Enschede, the Netherlands

Luenen, W.T.C. van (1993), *Neural networks for control, on knowledge representation and learning*, PhD Thesis, Department of Electrical Engineering, University of Twente, Enschede, the Netherlands

Poggio, T. and F. Girosi (1989), *A theory of networks for approximation and learning*, MIT, 1989

Oelen, W. and J. van Amerongen (1994), *Robust tracking control of two-degrees-of-freedom mobile robots*, Control Eng. Practice, Vol. 2, No. 2, pp. 333-340

Oelen, W. (1995), *Modelling as a tool for design of mechatronic systems*, PhD Thesis, Department of Electrical Engineering, University of Twente, Enschede, the Netherlands

Rumelhart, D.E., G.E. Hinton, R.J. Williams (1986), *Learning representations by back propagating errors*, Nature, Vol. 323, no. 9, October

Schipper, D.A. (1991), *MART-project, concept, doelstellingen en specificaties* (in Dutch), Internal report, no. WA-178/BSC-91R014, Department of Mechanical Engineering, University of Twente, Enschede, the Netherlands

Starrenburg, J.G. (1993), *Learning pose controller for the Mobile Autonomous Robot Twente*, MSc. thesis, report no. 93R118, Control Laboratory, Department of Electrical Engineering, University of Twente, Enschede, the Netherlands