Neural Motion Controller for Robots Daisy and Ester [*]

Zbyněk Winkler, Iveta Mrázová, Jiří Iša, Jakub Krchák

Charles University, Faculty of Mathematics and Physics,
Department of Software Engineering
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

e-mail: zbynek.winkler@mff.cuni.cz

Abstract

In this paper, we present a novel adaptive strategy for robust motion control of differentially steered robots Daisy and Ester. Daisy and Ester were originally built for the Eurobot 2004 competition and their architecture differs essentially – very high versus very low acceleration, tracked versus wheeled vehicle etc. Our approach employs an artificial neural network to drive the robots' wheels. Given the required velocities of the wheels the controller calculates the amount of power needed to move towards a set direction. In this sense, our approach differs from previous ones as it directly computes the required powers for the motors. Results of extensive experiments performed so far confirm that despite of the huge differences both tested robot architectures were able to follow the prescribed path and avoid collisions reliably.

1 Introduction

In this paper, we present a novel adaptive strategy for robust motion control of differentially steered robots. The strategy employs an artificial neural network to drive the robots' wheels. Unfortunately, practical applications of neural networks are often critically dependent on the way the task is represented. For this reason, our approach encompasses also an effective algorithm to compute the velocities for the wheels. Given the required velocities of the wheels the controller calculates the amount of power needed to move towards a set direction. In this sense, our approach differs from previous ones as it directly computes the required powers for the motors.

Most traditional controllers either assume a relatively simple system behaviour (e.g. linear) or require complicated settings of vague constants to work properly. Therefore a simple adaptive solution to this problem would be very important for a reliable control of mobile robots. Neural controller employed in our experiments was based on a multi-layer feed-forward neural network of the back-propagation type. This kind of neural networks is adapted by supervised learning requiring a set of input/target output samples. Trained networks are characterized by a relatively high robustness and efficient performance even for previously unknown data patterns – good generalization. A good generalization ability corresponds to a correct behaviour of the trained system also for previously unseen input patterns (i.e. the accuracy of the trained system on a test set of patterns not contained in the training set).

Results of extensive experiments performed so far have shown that the new proposed controller can steer efficiently both our robots Daisy and Ester even at speeds of up to 1 m/sec. Daisy

[*]This research was partially supported by the grant No. 201/04/2102 of the GA ČR and by the Program "Information Society" under project 1ET100300517.

and Ester were originally built for the Eurobot 2004 competition and their architecture differs remarkably – very high versus very low acceleration, tracked versus wheeled vehicle etc. Despite these huge differences both of them were able to follow the prescribed path and avoid collisions reliably.

2 Related Works

The traditional robot path-tracking problem has been studied in the last decade for use in automated office, hospital, and factory floor operations. An early approach to designing path-tracking controllers was to develop a specific dynamic model of the robot assuming straight line paths and a constant, slow velocity. Actions depended on the magnitudes of lateral, and orientation path-tracking errors [2]. However, velocity control was considered as a minor problem, especially if the velocity is kept sufficiently low.

Some studies that extend the robot's tracking ability to the cases of general curved paths and changing velocities have been carried out. An optimal controller capable of maintaining path following stability, and flexible for speed changes was implemented in [4]. In [3] was proposed a controller that is memoryless function of the lateral, heading, and velocity offsets. This controller takes into account dynamic and kinematic properties of mobile wheeled robots. Under the conditions of small offsets and constant tracking velocity, the controller can be simplified to a PID controller by means of decoupling and nonlinear feedback linearization.

Unlike these traditional approaches, recent work increasingly involves dynamically changing environments. Fuzzy logic control techniques imitating human drivers have been applied in robot path-tracking [9]. This control design is adaptive so as to cope with the characteristics of outdoor environments for which full dynamical models of the robot/terrain interactions do not generally exist. Velocity control is considered in their study, and error is calculated with respect to desired velocities. One new approach incorporates a switching policy that engages sequences of controllers to respond to non-stationary and unpredictable environments.

There has also been work in applying reinforcement learning to continuous domains [1]. In [10] a set of impedance controllers is used for both state estimation and tracking control on a mobile robot. Reinforcement learning was used to create switching policies that optimize time or energy in a path tracking task while optimizing a secondary objective (such as time or energy). It is even possible to learn policies for sequences of controllers that perform better than any individual controller and the learned policies perform well in novel situations. Paths in this framework can come from planners (or potential functions) and can be in service to any general task.

A rapidly increasing demand in complex dynamical control systems under significant uncertainty makes the application of neural networks very attractive. Neural networks commonly used in control are the so-called BP-networks with weights adjusted using the back-propagation algorithm. When a BP-network is trained as a controller, the desired output of the network might be not readily available but has to be induced from the known desired system output. In such a case, it is assumed that the inverse dynamics of the system can be represented by the network. The desired system output is then presented to the network as its input, and the actual network output is used as an input for the system.

Since the network represents in essence the system inverse, the system input computed in this way causes the desired behaviour of the whole system output. A major problem with inverse identification is that the system inverse is not always well-defined, that is, more than one system inputs can produce the same output. In this case, the network attempts to map the same network input to many different target responses [6].

An interesting application of neural networks to navigation and path planning can be found e.g. in [5]. The presented method is based on two neural networks. The first network is used to determine free space around the robot using ultrasound range finder data. The second network selects the movement direction. To make its final decision it combines the output of the first network with the current position of the robot and its target goal. Even though the results obtained in computer simulations look promising it is hard to assess the usefulness of the method once confronted with the real world.

3 Learning Algorithm and Analysis

A BP-training algorithm is an iterative gradient descent procedure which minimizes the value of an objective function expressing the desired behaviour of the trained BP-network. The network is trained by initially choosing small random weights and then presenting all training patterns repeatedly. Weights are adjusted using an additional information specifying the correct network output until weights converge and the objective function is reduced to an acceptable value. An essential component of the algorithm is the iterative method described in detail in [8] that propagates the error terms required to adapt weights backwards from neurons in the output layer to neurons in lower layers.

For the standard BP-training algorithm, the objective function E represents the total error between the desired and actual outputs of all the output neurons in the BP-network:

$$E \quad = \quad \frac{1}{2} \sum_{p} \sum_{j} (y_{j,p} \; - \; d_{j,p})^2 \tag{1}$$

where p is an index over all training patterns (will be omitted further in the text) , j is an index over all output neurons, y is their actual and d their desired output value. For presented input patterns, the network first computes its actual output and compares it with its desired output. Then, for this considered training pattern, the weights of the network are adjusted by:

$$w_{ij}(t+1) \quad = \quad w_{ij}(t) \;+\; \alpha\, \delta_j\, y_i \;+\; \alpha_m \,(w_{ij}(t) \;-\; w_{ij}(t-1)) \tag{2}$$

In this expression, the terms for δ_j correspond to:

$$\delta_j \quad = \quad \begin{cases} (d_j \; - \; y_j)\, y_j \,(1 \; - \; y_j) & \text{for an output neuron} \\ y_j \,(1 \; - \; y_j)\, \sum_{k} \delta_k \, w_{jk} & \text{for a hidden neuron} \end{cases} \tag{3}$$

w stands for weights and thresholds, i and k index neurons in the layers below and above the neuron j, respectively. d_j is the desired and y_j the actual output value of the neuron j. $t+1$, t and $t-1$ index next, present and previous weights, respectively, α is a constant representing learning rate. α_m is a constant between 0 and 1 which determines the effect of past weight changes on the current direction of movement in the weight space. This provides a kind of momentum that effectively filters out high-frequency variations of the error surface.

Unfortunately, a good training accuracy does not necessarily guarantee a satisfactory robustness and/or generalization capabilities of the trained network. Often, a good training accuracy can be achieved only by forming complex decision boundaries, which in turn requires a large

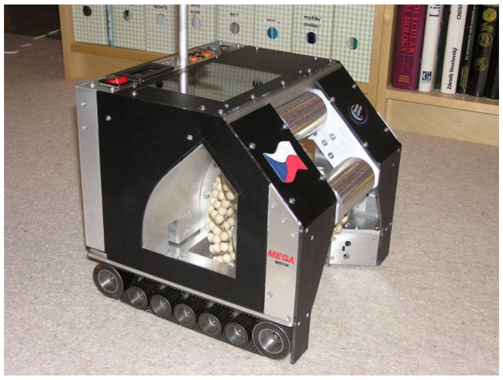
Figure 1: Ester

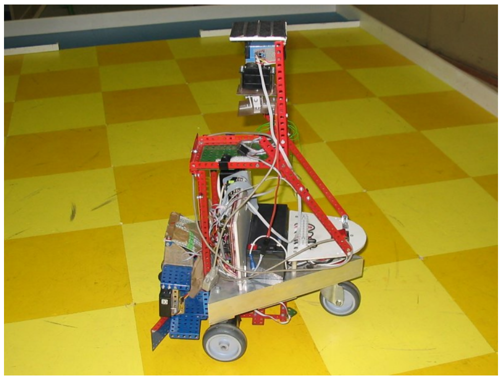
Figure 2: Daisy

network size and a time-consuming training process. Over-fitted networks are generally characterized by mappings which have a lot of structure and a relatively high curvature. This provides some indirect motivation for weight-decay regularizers as a way of reducing the curvature of the network function.

4 Supporting Experiments

The goal for our experiments was to train a BP-network to drive a differentially steered robot. Our ultimate idea was to create an autonomous robot able to adjust its behaviour both to its changing dynamics (e.g. charged/discharged batteries, dirty wheels) and its environment (e.g. wooden floor/carpet). Here, we intend to apply an off-line training regime. In practice this means to let the robot move in its environment first. Then, its neural controller has to be trained according to the data collected in the off-line phase. With the trained controller, the robot should be able to move autonomously in the now-known environment. Experiments discussed below were done with Ester. Ester is a tracked robot, but even though Daisy is a wheeled robot, very similar results were obtained as well.

The frequency for steering the robot Ester is 250 Hz. In principle, this means that 250-times a second we are asking the network to output an adequate value for the power on both motors while presenting it the value for current velocities and for the velocities required in the following step. An important issue has been connected with the resolution of the robot's wheel encoders. One pulse of the encoder corresponds to the distance of 0.7 mm. A usual acceleration reached by the movement of Ester is around 2 m/s^2.

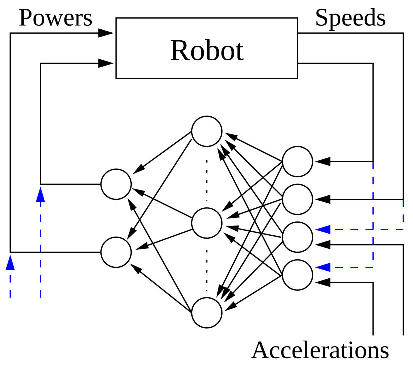

Figure 3: The proposed robot controller

An example of the data provided by the encoder is shown in Figure 4. However, this data is discrete and heavily corrupted by quantization noise. To obtain an adequate approximation for the speed and acceleration of the trained robot, it was necessary to use a second-order polynomial to approximate the quantized data. The first-order derivative of the data was used as an approximation for velocity and its second-order derivative as an approximation for acceleration. They are shown in Figure 4 as the dashed and dashed-and-dotted curves, resp.

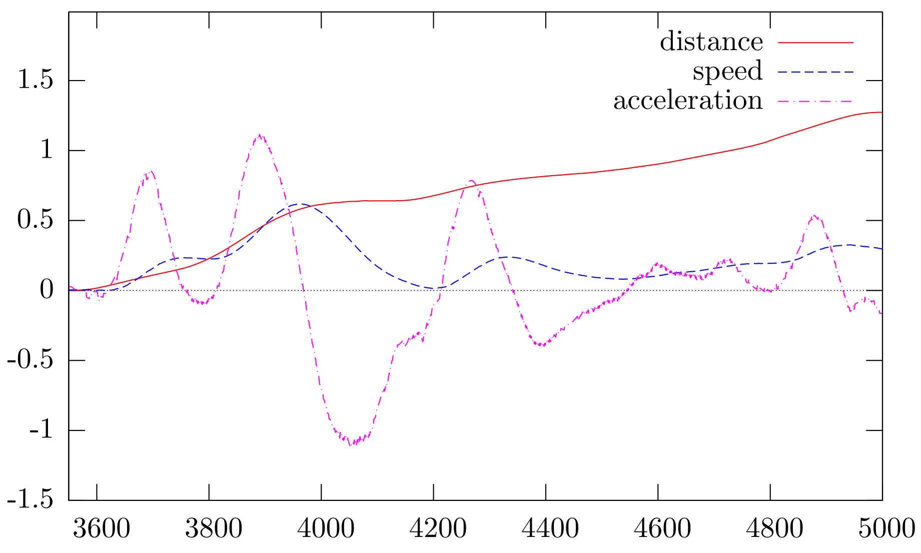

Figure 4: Data obtained by an encoder. The dashed curve corresponds to the approximated velocity and the dash-and-dotted curve corresponds to the approximated acceleration.

Figure 3 shows how to apply a BP-network to control the inverse dynamics of a mobile robot. For training, a large stream of real data was provided by the robot (27298 patterns in our case). To form the training set, the sequence of tuples $\ldots, [power_i, velocity_i], [power_{i+1}, velocity_{i+1}], \ldots$ has been transformed into a sequence $\ldots, [[velocity_i, acceleration_{i+1}], power_{i+1}], \ldots$ of training patterns. The transformed data has the following interpretation: given the current velocity $velocity_i$ of the wheels and accelerations $acceleration_{i+1}$ required for the next step, the network is trained to return the proper powers $power_{i+1}$ for both motors to achieve the required acceleration. During training, the training patterns are selected randomly from the above transformed sequence.

Experimentally chosen architecture for the controller was $4 - 10 - 2$. The values for the learning rates and momentum corresponded to $\alpha = 0.02, \alpha_m = 0.01$. The mean squared error fell to 0.026 on the training set after 1000 iterations. For Ester, the time required to complete the off-line phase of training is about 3 minutes for both collecting the data and training the controller. Finally, the trained networks were used to drive Ester and Daisy. A comparison of a controller output and its reference values is shown in Figure 5. The required accelerations were provided externally, e.g. by the path planner.

5 Conclusions

In this paper, we presented a neural controller for mobile robots. The proposed controller is able to adjust quickly, both to changed dynamics of the robots and to their environment. Moreover, the network approximating the inverse dynamics is very small – 10 hidden neurons. This is an important issue especially when considering its application to control smaller robots with limited computational resources. The new motion controller has been tested extensively in a series of real-world experiments. Our further research will be focused to solve the problems connected with on-line training of the proposed controller. This would enable the robots to adapt even to rapidly changing environments without human intervention.

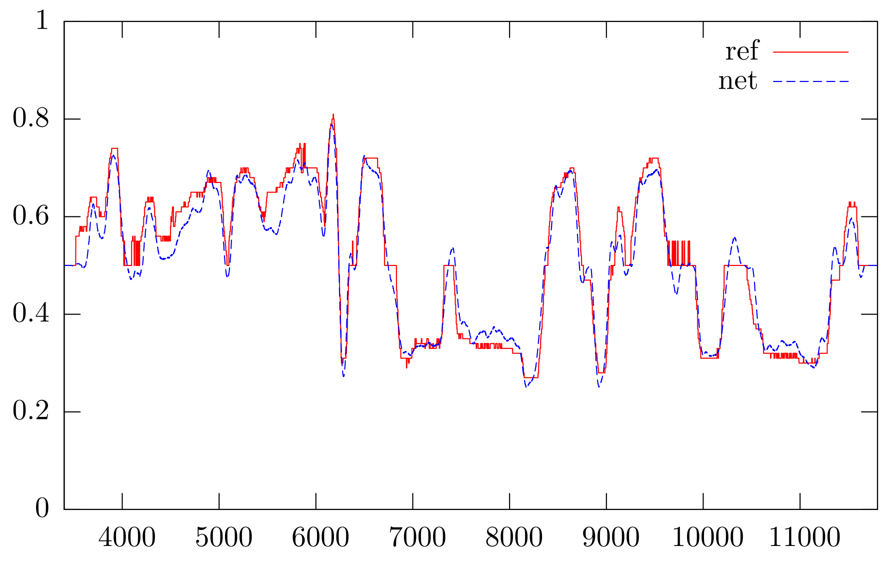

Figure 5: The graph shows the output of the trained controller and its reference output on a test set.

References

[1] J. A. Coelho jr., E. G. Araujo, M. Huber, R. A. Grupen: *Dynamic categories and control policy selection*, in: *Proc. of ISIC/CIRA/ISAS Joint Conf.*, Gaithersburg, MD, (1998).

[2] I. J. Cox, G. T. Wilfong, (eds.): *Autonomous Robot Vehicles*, Springer-Verlag, (1990).

[3] R. M. DeSantis: *Modeling and path-tracking control of a mobile wheeled robot with a different drive*, in: *Robotica*, Vol. 13, (1995), 401-410.

[4] A. Hemami, M. G. Mehrabi, R. M. H. Cheng: *Optimal kinematic path tracking control of mobile robots with front steering*, in: *Robotica*, Vol. 12, (1995), 563-568.

[5] D. Janglová: *Neural Networks in Mobile Robot Motion*, in: *International Journal of Advanced Robotic Systems*, Vol. 1, No. 1, (2004), 15-22.

[6] C.-T. Lin, C. S. G. Lee: *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*, Prentice Hall, (1996).

[7] V. M. de Oliveira, E. R. de Pieri, W. F. Lages: *Neural Networks in the Control of a Mobile Platform* in: *IFAC Symposium on Robot Control — Syroco'2000, Viena, Austria*

[8] D. E. Rumelhart, G. E. Hinton, R. J. Williams: *Learning representations by back-propagating errors*, in: *Nature*,(323) (1986) 533-536.

[9] O. Sanchez, A. Ollero, G. Heredia: *Adaptive fuzzy control for automatic path tracking of outdoor mobile robots - application to Romeo 3R*, in: *Proc. of FUZZ-IEEE 1997*, (1997) 593-599.

[10] Y. Wang, B. Thibodeau, A. H. Fagg, R. A. Grupen: *Learning Optimal Switching Policies for Path Tracking Tasks on a Mobile Robot* in: *Proc of IROS*, (2002)