A Comparison of Traditional and Soft-Computing Methods in a Real-Time Control Application

B. Sick, M. Keidl, M. Ramsauer, S. Seltzsam
University of Passau (Prof. Dr. K. Donner, Prof. Dr.-Ing. W. Grass)
Innstr. 33, 94032 Passau, Germany (email: sick@fmi.uni-passau.de)

Abstract

The paper presents a performance comparison of traditional (controller based on a physical model) and soft-computing methods (neural and fuzzy controller) applied to a real-time control problem. The pros and cons of the controller paradigms will be investigated in this paper by means of an application example. In this example a ball has to be navigated through a maze mounted on a board. The board can be tipped in any direction using two step motors. An image of the board is supplied by a CCD-camera and two signal processors are used to process the image sequences and to generate motor step sequences, respectively. The results show that the choice of a suitable controller for a specific task mainly is a trade-off between the accuracy of the motion, the ball's speed and the execution time of the particular control algorithm.

1 Introduction

In the past years more and more soft-computing techniques (based on e.g. neural networks or fuzzy systems) have been developed for the solution of control problems. The reason is that many real-world control problems can hardly be solved by means of conventional techniques because needed information is not available or the systems under consideration are not well defined. The principle of soft-computing is: exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness, low solution cost and even better rapport with reality [1]. In this sense, soft computing imitates certain human abilities. However, soft-computing techniques are not a panacea. Even if they are often very easy to handle, results have to be validated carefully. This paper presents a comparative evaluation of three different traditional and soft-computing techniques applied to a real-world control problem.

In this application example, a ball has to be moved through an arbitrary maze mounted on a board. The motion of the ball is achieved by tipping the board using two step motors. A video camera is placed above the board to observe the effect of the board's motion. At first sight the problem might appear playful, but similar or related problems (e.g. motion or balance problems) arise in many industrial applications. Fig. 1 shows the hardware components (left) and a photography of the board (right). The overall problem is solved in two subsequent steps. In the first step (pre-processing) it is necessary to detect the position of the ball and to find a passable way through the maze. The solution for this problem has been described in [2]. The result is a path from the starting to the target point of the maze in form of a polygonal. The second step is the real-time phase which is responsible to navigate the ball along the path by approaching the corner points of the polygonal one after the other. Initially, the new coordinates of the ball's center are estimated by considering some past positions of the ball. Then, the exact center is localized in an area ("sliding window") around the estimated center using e.g. a prototype fitting method (see [2]). The current position of the ball and the position of the next target point (corner of the polygonal) are used to calculate the motor step sequences. There are different possibilities to handle this task.

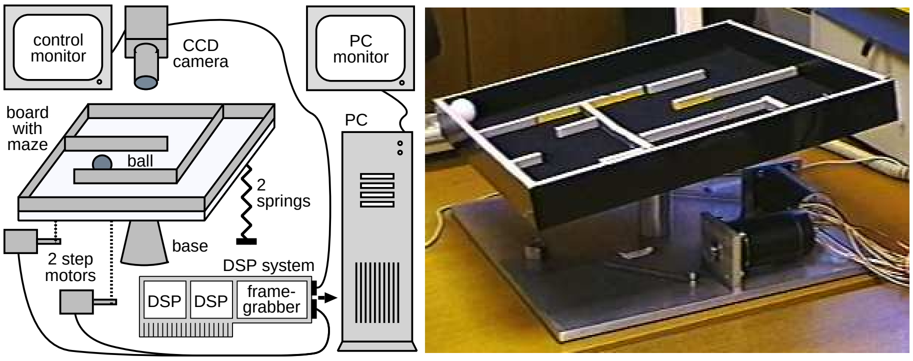

Figure 1: The hardware equipment of the application example

2 Controller Paradigms

Three types of controllers have been investigated: a fuzzy controller, a neural network and a conventional controller which is based on a physical model of the motion of the ball. These approaches are discussed in detail below.

2.1 Fuzzy Controller

The fuzzy controller doesn't need a precise physical model of the ball's behaviour. An intuitive description of the desired behaviour of the controller by means of linguistic variables, linguistic terms and fuzzy rules is sufficient [3]. The distance between the current position of the ball's center and the next corner of the polygonal and the current speed of the ball have been considered to be appropriate input variables for the controller. It is easy to see that the controller's behaviour in x- and y-direction has to be the same. Therefore, it is possible to use two identical controllers, one for each direction. In the same sense, the motion in positive direction is symmetrical to the motion in negative direction. For that reason, controllers with only one output variable (the signed number of steps in one of the two directions) have to be designed.

After defining the input and output variables of the control algorithm, it is possible to describe the transfer function of the controller by means of linguistic terms for the input and output variables and fuzzy rules. As described above, the linguistic variables are distance (in "pixels"), speed (in "pixels per image", i.e. per 1/50 *sec*) and number of motor steps. The corresponding linguistic terms (or fuzzy sets) are simple triangular functions. To enable the fuzzy controller to calculate the required number of motor steps, a set of fuzzy rules like "if low negative speed and high negative distance then many negative steps" has to be defined. The set of fuzzy rules describes the transfer function of the fuzzy controller. The controller can be considered to be a static, non-linear, and non-adaptive system [4]. We started with a more or less canonical definition of linguistic terms and rules and optimized the controller manually.

Further improvements of the controller's performance have been achieved by changing the way of tracking the path. With the original version of the fuzzy controller problems arise when the ball drifts away from the path, because the ball moves in the direction of the next target corner and not back to the path. To avoid this undesired behaviour the ball is led back to the path by calculating a virtual corner. If the center of the ball is far away from the path this virtual corner is the nearest point of the current path segment. The nearer the ball is to the path the nearer the virtual corner gets to the target corner. To prevent the ball from "lurching" from one side of the path to the other instead of moving in a straight line an additional PID controller has been connected with the output of the fuzzy controller. The PID variables have been optimized manually. With this PID, the overall control algorithm shows a dynamic behaviour.

2.2 Neural Controller

The main reason to use a neural controller is the possibility for an automatic adaption of the controller to other environmental conditions (e.g. another ball with different weight or other board surfaces; not evaluated here) and an improved performance due to a differentiable transfer function.

An offline pre-trained 3-layer feedforward neural network (multilayer perceptron, fully connected, four input neurons, 10 hidden neurons and 2 output neurons) is used as a starting point for a continuous, supervised online training [5]. Again, the inputs are the speed and the distance in x- and y-direction. The two output neurons give the number of motor steps. By using a pre-trained network, the duration of the training is shortened and the hardware is prevented from getting damaged in an early stage of the training (this would be the result of using randomly initialized weights). The learning patterns for the pre-training have been obtained by evaluating the transfer function of the fuzzy controller. Therefore, the neural controller and the fuzzy controller show quite the same behaviour after the offline training.

In order to improve the controller's performance two new input variables have been introduced: the inclination of the table and the deviation of the ball from the given path. Therefore four input neurons had to be added. The corresponding *additional* weights have been initialized with randomly selected values (unlike the original weights). In order to obtain "good" target output values for a supervised online training it is necessary to assess the obtained output values. By modifying the weights of the network in an appropriate way using the backpropagation learning algorithm, the behaviour of the neural controller can be improved continuously. The network should perform better in two ways: first, the speed of the ball should be increased and second, deviations from the given path should be minimized. For that reason, the target output values are calculated as a weighted sum of three values: the real output, a speed adjustment factor and a "lead the ball back to the path" adjustment factor. The speed adjustment is the difference between the desired speed and the real speed, and the lead-back values penalize major deviations of the ball from the given path. It is possible to determine whether the controller should be more accurate or faster by setting the weights in this sum in an appropriate way.

With the described online adaption of the synaptical weights, this controller type can be considered to be a static, non-linear, and adaptive system [4].

2.3 Conventional Controller

The conventional controller is based on a extremely simplified physical model of the ball/board system. Because an appropriate measuring hardware has not been available, only a few values have been determined experimentally: γ_n (the inclination increment of the board when one impulse is sent to a step motor), $a_{\max}$ (the maximum acceleration which can be achieved due to the limited inclination of the board), $v_{\min}$ and $v_{\max}$ (the minimum and maximum speed of the ball), and $\delta_{\max}$ (a certain short distance to the target point where the ball should begin to slow down). The acceleration of gravity is g_t.

The control algorithm consists of two parts: First the desired speed v_f is calculated and then the number of impulses which have to be sent to the board to achieve this speed is computed. The formulas described below are identical for each axis. The input values of the controller are the distance to the target point δ ($d = \mathrm{sgn}(\delta)$ defines the direction), the current speed of the ball v_c, and the number of steps the table diverges from the starting position t_c. The desired speed of the ball v_f can now be computed by

$$
\begin{aligned}
0 \leq |\delta| \leq \delta_{\max} &: v_f := \delta \cdot \tfrac{1}{sec} \quad \text{(Slow down the ball.)} \\
|\delta| > \delta_{\max} &: \text{Adjust the ball's speed so that } v_f \in [v_{\min}, v_{\max}]: \\
&\qquad (0 \leq |v_c| < v_{\min}) \vee (|v_c| > v_{\max}) \ : \ v_f := d \cdot v_{\min} \\
&\qquad\qquad (v_{\min} \leq |v_c| \leq v_{\max}) \ : \ \text{Don't move the board.}
\end{aligned}
$$

Finally, the number of motor steps s is given by

$$s := \frac{\gamma_f}{\gamma_n} - t_c,$$

where

$$
\begin{aligned}
a_c &:= \sin\!\left(\tfrac{t_c \cdot \gamma_n \cdot \pi}{180^\circ}\right) \cdot g_t \quad \text{(current acceleration)}, \\
a_f &:= \mathrm{sgn}(a'_f) \cdot \min\{|a'_f|, a_{\max}\} \text{ with } a'_f := \tfrac{v_f - v_c}{1/50 sec} - a_c \\
&\quad \text{(required acceleration to speed up the ball from } v_c \text{ to } v_f\text{)}, \\
\gamma_f &:= \arcsin\!\left(\tfrac{a_f}{g_t}\right) \cdot \tfrac{180^\circ}{\pi}.
\end{aligned}
$$

The time to execute a certain number of steps is limited to $\frac{1}{50}$ sec, so only a limited number of steps may be executed. Therefore s is scaled to $s \in [-s_{max}, s_{max}]$ where s_{max} is the maximum number of motor steps.

As in the case of the fuzzy controller (and for the same reason), a PID controller is placed behind the conventional controller. With this PID the overall controller's behaviour changes slightly. The overall conventional controller is a dynamic, non-linear and non-adaptive system [4].

3 Results

To assess the different controller types, each controller has been tested using the same maze and the same path. To compare the controllers in a fair way, each test has been repeated twenty times and **mean** μ and **standard deviation** σ of the following rating criteria have been determined (see tab. 1):

- the covered *distance* of the ball (measured in pixels),
- the *run time* between beginning (start point) and end (last target point) of the motion (measured in seconds),
- the *average deviation* (measured in pixels),
- the *max. deviation* of the ball (measured in pixels),
- the average number of processor *cycles* per calculation,
- the number of *motor steps.*

The path through the maze had been hard-coded in order to guarantee exactly the same path for each test and, therefore, comparable results. This "reference path" had a length of 1139 pixels. Results for the online trained network are shown **after** a certain time of online training (with online training the number of processor cycles is about 67000).

paradigm	fuzzy controller		pre-trained NN		online trained NN		conv. controller	
	μ	σ	μ	σ	μ	σ	μ	σ
distance	1478	26.3	1459	27.3	**1410**	31.7	1560	24.0
run time	70.2	1.21	66.7	1.24	**52.9**	0.88	59.4	1.02
avg. dev.	2.50	0.21	**1.88**	0.16	3.47	0.19	3.24	0.22
max. dev.	12.1	1.40	**10.4**	1.89	13.9	1.36	16.6	4.02
cycles	3669	0.73	10992	0.07	10911	0.06	**465**	5.31
mot. steps	12316	177	11244	134	**9048**	193	9520	284

Table 1: Results of the controller experiments

It can be noticed, that the pre-trained neural network (which requires the transfer function of the fuzzy controller for the training) is the best to navigate the ball close to the given path (mean average dev. 1.88 pixels, mean maximum dev. 10.4 pixels). The online trained network guides the ball very fast through the maze (mean run time 52.9 seconds) and, therefore, the mean covered distance is the shortest (1410 pixels). Additionally, this controller needs the smallest number of motor steps (9048). Finally, the conventional controller has a very fast control algorithm (mean number of processor cycles 465).

4 Conclusion

It can be stated, that all controller types are able to control the motion of the ball in an excellent manner. The controllers (and the image processing algorithms, too) have to cope with a lot of disturbances in the control loop (see fig. 2). It is very difficult or even impossible to integrate these influences in a precise model of the system in order to consider them in the control algorithms.

The main advantage of the fuzzy controller is its robustness, e.g. dropped motor steps have only very little influence on good controlling results. On the other side this controller is very hard to adapt to different environments (weight of the ball, friction of the board's surface, etc.). The linguistic terms and other parameters have to be optimized manually. The advantage of the neural network is its ability to learn an input/output relationship and, therefore, to adapt to different environments (not evaluated here). In this report the learning didn't start with randomly initialized weights. The network has been pre-trained offline by means of the transfer function of the fuzzy controller. After an offline training the user may specify whether the controller should train online on accuracy or speed or a combination of both. The main disadvantage of the online training is the execution time. Another problem is to control the learning parameters like momentum and learning rate carefully to guarantee convergence in learning. The conventional controller based on a extremely simplified physical model of the ball/board system may be used as well. The motor steps are calculated very fast and the average speed is excellent. On the other hand, adapting this controller to other environmental conditions is hard due to the necessity of adjusting the parameters of the underlying model manually. However, the period of development for this controller has been surprisingly short compared with the other controllers. The results show, that the overall system behaviour is a trade-off between accuracy of the motion, the ball's speed and the execution time for the control algorithm.

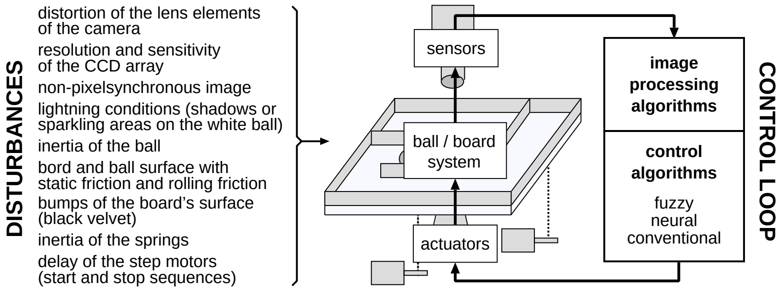

Figure 2: Disturbances in the control loop

Acknowledgements: The authors are grateful to the other members of the project team for the contribution of the interface hardware (R. Kickingereder, W. Maydl) and the image processing algorithms (G. Schichl, V. Vogelhuber).

References

[1] Zadeh LA. What is soft computing?. in: Soft Computing, vol. 1 no. 1, 1997

[2] Keidl M, Kickingereder R, Maydl M, Ramsauer M, Schichl G, Seltzsam S, Vogelhuber V. Praktikum im Vertiefungsgebiet Technische Anwendungen der Informatik – Dokumentation. technical report, University of Passau, 1997

[3] Kahlert J. Fuzzy Control für Ingenieure. Verlag Friedr. Vieweg & Sohn, Braunschweig, Wiesbaden, 1995

[4] Oppenheim AV, Willsky AS. Signale und Systeme. VCH, Weinheim, 1992

[5] Rojas R. Neural Networks – A Systematic Introduction. Springer-Verlag, Berlin, Heidelberg, New York, 1996