Comparing Control Strategies for Autonomous Line-Tracking Robots

Luís Almeida, Alexandre Mota, Pedro Fonseca
{lda,alex,pf}@ua.pt

Departamento de Electrónica e Telecomunicações
Universidade de Aveiro, P-3810 Aveiro, Portugal
tel.: +351-34-370 859 fax: +351-34-381 128

Abstract

Autonomous mobile robotics is a very exciting area for students particularly for those who attend courses on electronics. The authors have been involved in several activities in this area together with students of the

model of the line-following robot [1]. The model takes into account several real-world constraints and allows to predict the movement of the robot based on the electrical voltages applied to the motors.

Also in [1] the authors have described the geometry of the line-tracking process which was used to build a

used to keep the robot horizontally. The deviation of the robot from the reference path is measured by a set of sensors placed ahead of the robot which are, normally, infrared-light detectors.

Typically, closed-loop control of the wheels' velocity has not been done. The velocity of each wheel is controlled indirectly by applying voltages to the motors. This option may decrease the performance of the tracking algorithm but simplifies the final tuning. Remember that the use of closed-loop wheel speed control would require the tuning of two extra independent loops.

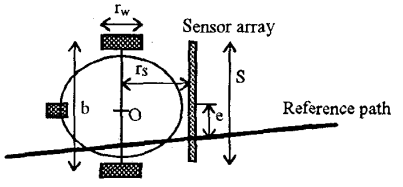

Figure 1. The basic robot

These characteristics have been used to derive a model for the line-tracking robot (fig. 2). To improve its accuracy the model takes into account inertia (mass (M) and moment of inertia (J)), friction coefficients (for translational (B_v) and rotational (B_ω) movements), electric motors parameters (the resistance (R) and the motor constant (K_m)), additive noise (in the sensor readings) and physical limitations of the robot such as the length of the line-sensors (S) and the maximum voltage that can be applied to the motors (V_{max}). The model is described in [1] and allows to calculate both linear (v) and angular (ω) velocities of the robot based on the voltages applied to the motors (V_{av} - average, and V_{dif} differential).

2.2 The Line-Tracking Simulator

The robot model referred to above, was complemented with a geometric analysis of the line-tracking problem. This problem falls within the general path-tracking problem which has been treated in the literature, e.g. [2]. In particular, the simulator presented in this article

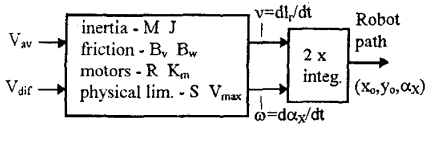

Figure 2. The robot model

follows a reactive approach to track an unknown line as opposed to the planning approach of tracking a path previously planned and thus, known in advance.

In [1] a geometric analysis is also shown that allows to calculate the next deviation from the line (e) based on the present deviation, wheels velocities and angular position of the robot relative to the line. The robot is used as referential. However, in order to better define the reference trajectory and to visualise the robot trajectory, another model was built in which the robot position was referred to an absolute referential. In this geometric model, the next deviation from the line (e) is calculated based on the robot absolute position and the wheels velocities. Knowing the robot position ($x_o, y_o, \alpha x$) it is possible to calculate the intersection of the sensor array with the line (x_e, y_e) which then allows to calculate the deviation e (fig. 3). The resulting dependency of e relative to the position of the robot is non-linear.

The velocities are used to calculate the robot displacement ($dl_r, d\alpha_x$) during an infinitesimal time interval (dt). If this interval is kept small enough then it is irrelevant if the linear movement is considered separately from the angular movement and which of them is considered first. In the experiments that were conducted, an interval such that a trajectory point was calculated every 5mm was small enough to obtain the same trajectory whether angular or linear movements were considered first.

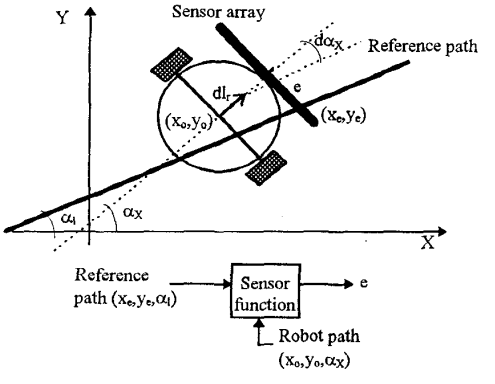

Figure 3. The line-tracking geometric model

The geometric model used allows for reference lines composed of straight segments and arcs of circunference smoothly joined one after another. Although it might seem limitative, it allows to create almost any sort of smooth trajectory by using arcs with different radius.

2.3 The Reference Path

The simulations that were carried out in this work used a reference path that was composed of straight segments interleaved with arcs of circunference of 1m radius and 90° or 180° aperture. Such path is depicted in fig. 4 and had a total length of aprox. 30m.

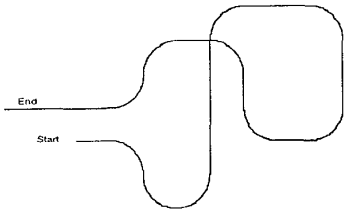

Figure 4. The reference path.

2.4 The Robot Parameters

The robot used as the basis for the simulations presented in this article represented the University of Aveiro in the edition of '96 of the International Mobile Robotics Championship that took place in La Ferte-Bernard, France, in May of that year. For details concerning the parameters below refer to [1,3].

Weight M=3.2 Kg
Moment of inertia J=0,7 Kg.m2
Max. voltage applicable to motors V_{max}=11.5 V
Motor parameter (R=7 Ω and K_m=0.86 N/A)
Wheels diameter r_w=0.03 m
Distance between wheels b=0.27 m
Linear movement friction coefficient B_v=40 Kg/s
Angular movement friction coefficient B_ω=0.25 Kg.m^2/s
Type of sensor array - linear with saturation
Width of sensor array S=0.18 m

3 Comparing Control Strategies

As can be seen from fig. 2, the robot model has two inputs, V_{av} and V_{dif} which are the average and differential voltages applied to the drive motors. However, there is only one error signal which is the deviation e of the robot from the reference path as perceived by the sensor array. If the robot is always moving forward, it can be seen that any control strategy that minimises e will make the robot converge to the reference path.

Since the differential voltage V_{dif} is the one that determines the angular movement of the robot, allowing it to change direction so that it converges to the line, one simple possibility is to use e to directly control V_{dif}. In this case, since the final purpose is to achieve the maximum speed over the reference path, the average voltage V_{av} can be set to its maximum value $\mathbf{V_{max}}$. However, the physical voltages applied to the drive motors are limited to $\mathbf{V_{max}}$. Reflecting this correction onto the average and differential voltages yields V_{av}' and V_{dif}' which will be actually supplied to the robot model.

Also, the output of the sensor function is corrupted with additive noise (normal distribution and std.dev.=2.5mm). This noise allows to incorporate such effects as imperfections on the line or floor, electrical interferences in the sensor readings and limited precision of the sensors. To facilitate comparisons the noise vector was kept the same for all runs from the *Start* to the *End* points.

The control is digital, with a sampling period of 100 ms.

The complete control system is depicted in figure 5. The reference input is the path to be tracked. The error signal is the deviation read by the sensor array.

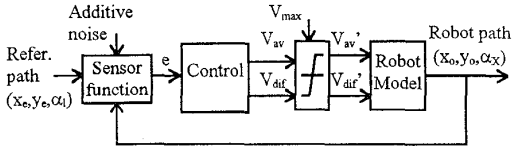

Figure 5. The complete control system

In this simple model the controlling function can be written as in expression (1).

$$V_{dif} = f(e) \quad \text{and} \quad V_{av} = \mathbf{V_{max}} \qquad (1)$$

In very fast robots it can be of interest to use a V_{av} as a function of e as well. For example, this could be used to slow down the robot while describing a curve and to accelerate it along straight segments. Nevertheless, the robots that have typically been built are not very fast, running at less than 0.5m/s. Thus, in the rest of this paper the simple approach as in (1) will be used.

To compare the performance of each control approach two main measures have been used, the integral absolute error (IAE) and the integral squared error (ISE), integrated along the full reference path. Two other measures have also been used, the maximum absolute error ($|E_{max}|$) and the average speed achieved by the robot (v_{mean}).

3.1 Proportional Control

The simplest form of control is to use a proportional controlling function generating $V_{dif}=K_p*e$. Although simple, this method presents several problems. The best value for K_p is difficult to find (requires many trials) particularly in a non-linear system such as this one. Also, the best performance it can give is relatively poor because it is unable to compensate the lag caused by robot inertia. For the full reference path of figure 4, using a proportional controller with K_p=200, resulted in the deviation plot depicted in figure 6. Notice the typical oscillations resulting from the poor control obtained with a simple proportional approach. The IAE is 15.1 and the ISE is 0.33. The absolute maximum error is 49 mm and the average robot speed is 0.3ms^{-1}. In spite of this poor behaviour this type of control approach is useful for comparing purposes.

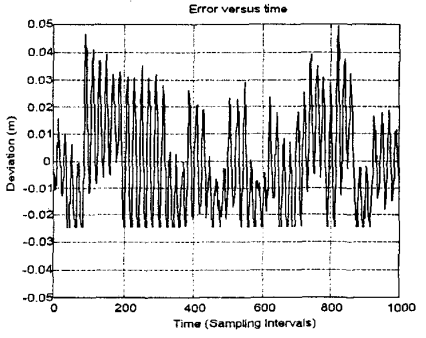

Figure 6. Using a proportional controller with K_p=200.

3.2 Proportional-Derivative

A simple way to compensate the lag caused by robot inertia, improving the performance of the proportional controller, is to add a derivative component (change in error *ce*) to the actuating signal. This creates a correction that opposes to the oscillations seen in fig. 6. The controller output is $V_{dif}=K_p*e+K_d*ce$.

Nevertheless, the existence of two independent parameters, K_p and K_d, makes the tuning of the controller even more difficult. The resulting performance is, in many cases, sufficient. Figure 7 shows the deviation obtained along the reference path with K_p=200 and K_d=200. The maximum deviation is 33 mm and the IAE is 9.5.

It is possible, by the use of an intensive off-line trial and error approach, to find the best values for K_p and K_d. It

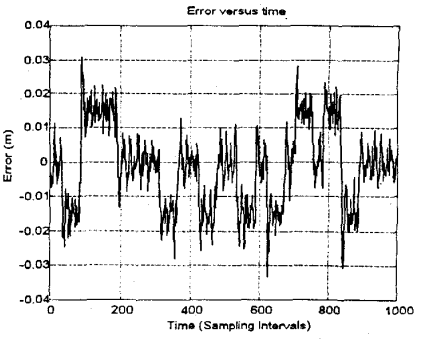

Figure 7. Using a PD controller with K_p=K_d=200.

was found that best results were obtained with K_p=400 and K_d=380. Figure 8 shows the deviation obtained along the reference path with these values. The absolute maximum deviation is 23 mm and the IAE is 6.6.

A note worth referring is the fact that the controller is capable of converging to zero deviation over straight segments but in curves with constant radius, the deviation converges to a non-zero value. Since the angle of the reference path is constant in straight segments (step input) and increases constantly in curves with fixed radius (ramp input) the line-tracking robot can be considered as a type 1 system. The same happens with the proportional controller.

3.3 Proportional-Integral-Derivative

This type of controller, known as PID, results from the previous one by adding an integral term to the actuating signal. This allows to bring the deviation to zero over any part of the line, either straight or curve. The deviation

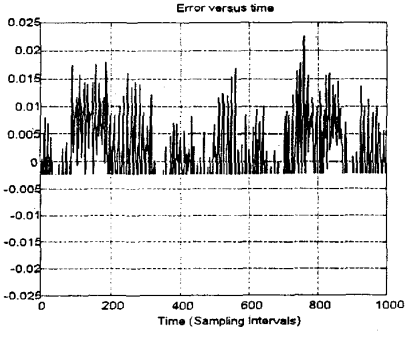

Figure 8. Using a PD controller with K_p=400 and K_d=380.

can be kept very small when the right parameters are used although it always increases in the beginning and ending of any curve. The controller output is $V_{dif}=K_p e+K_d ce+K_i I(e)$. With K_p=200, K_d=200 and K_i=100 it was possible to decrease the IAE (7.8) and the maximum absolute error (25 mm). No attempt was done to find the best 3 values. Anyway the results are better than with the non-optimised PD controller.

Although controllers of this type normally achieve a good performance, the tuning of the 3 constants is very difficult. The use of non-optimal constants may cause a considerable degradation in performance.

3.4 Fuzzy Logic Approach

The fuzzy logic approach can be an alternative to the previous strategies. Although it is more complex than either P, PD or PID approaches, it is still relatively easy to implement since it is based on intuitive rules explicitly given by the programmer [4].

In this case a fuzzy incremental controller with normalised universes of discourse and gaussian membership functions is used [5]. The controller inputs are the trajectory error (e) and its derivative (ce). The controller output is the differential voltage V_{dif}. The fuzzy control surface can be depicted on figure 9. Note the non--linear surface and the gradient near the center.

Two approaches were tried with this type of controller: rule-based and table-based algorithm. The first one uses functions from the MathWorks Fuzzy Logic Toolbox. The second is only a 2D Look-Up Table. The results are identical in terms of IAE, ISE and maximum absolute error. Some better results were obtained adding a linear integral term to the fuzzy algorithm (see table 1 for details). However the results were a little bit more modest than the ones obtained with the optimised PD controller. Trying to improve these results lead to the use of a self-organising fuzzy controller.

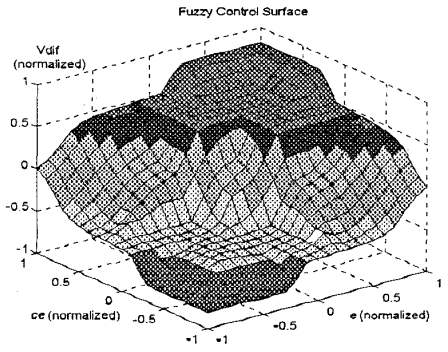

Figure 9. Fuzzy Control Surface.

3.5 Self-Organising Fuzzy Approach

The self-organising fuzzy controller (SOC) uses some kind of performance measure to update the rule base. The most common approach has a hierarchical structure in which the lower level is a table-based controller. The higher level monitors the error and the change in error and modifies the table, when necessary, through a modifier algorithm [6]. The performance measurement is carried out using expression 2.

$$P = (e + K_{ce} * ce) * G_p \qquad (2)$$

P is the performance measure or the penalty, that is added to the control table, e is the error and ce is the change in error. K_{ce} is a time constant and G_p is the learning rate factor. Starting with a table similar to the one used on the table-based controller it is possible, after 10 training sessions (of one full referece path each), to improve the overall performance up to the one obtained with the optimised PD controller. Figure 10 shows the IAE evolution along the 10 training sessions. Note that the training occurs "on-line" while the robot is actually moving along the line.

As well as with the simple fuzzy approaches, the addition of an integral action to the SOC allows to achieve even better results as can be seen in table 1.

3.6 Neural Networks Approach

Knowing that the robot model predicts non-linear, yet

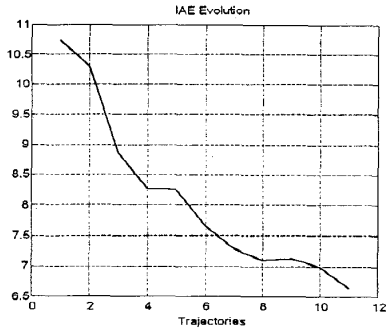

Figure 10. Self-Organising Controller IAE evolution

stable, dynamic behaviour lead to the idea of using some kind of neural-network approach in order to implement a direct inverse control algorithm. The inverse model was identified by the use of a 2 layer feedforward network with 4 inputs, 8 hidden non-linear neurons and a linear output neuron. The network was trained off-line with the Levenberg-Marquardt method [7] and, after 5000 epochs, it was possible to get a "good" inverse model. With the obtained network a direct inverse control scheme was implemented [8]. The results obtained this way are the best ones among the compared control strategies as can be seen in table 1.

4 Conclusions

Table 1 presents the results obtained with each controlling approach. Two main sorts of controllers were used, those capable of learning (SOC, SOC+I and NN) and the remaining ones. From these latter ones it is possible to see that the use of fuzzy controllers does not bring along an immediate benefit. A simple "hand--tuned" PD controller performs better. When an integral component is added to the fuzzy controllers, their performance is improved up to the one of the PD controller. However, the V_{mean} parameter is still superior in the PD approach. Notice that a difference of $0.03ms^{-1}$ yields a difference of 10s after 30m.

Optimally tuning a PD controller is either very difficult (highly time consuming) or even impossible when there is no analytical model of the robot and the tuning has to be done with the real robot. The PID approach is also difficult to tune and, in many cases, the resulting performance may even be worse than for the PD.

The tuning of the fuzzy controllers is easier to achieve since it is embedded in the intuitive rules explicitly given by the programmer (rule-based approach) or in the table built from such rules (table-based approach).

Table 1. Comparing different control strategies to track an unknown line.

Controller	IAE	ISE	Vmean	|Emax|	%IAE/P
P	15.1	0.33	0.30	0.049	1
PD	9.5	0.14	0.32	0.033	37
PD(optimized)	6.6	0.07	0.30	0.023	56
PID	7.8	0.09	0.31	0.025	48
FUZZY	10.8	0.21	0.31	0.051	28
FUZZY+I	9.5	0.14	0.29	0.037	37
FUZZYTABLE	10.7	0.21	0.31	0.053	29
FUZZYTABLE+I	9.5	0.13	0.29	0.034	37
SOC (10 epochs)	6.6	0.07	0.31	0.024	56
SOC+I (15 epochs)	6.2	0.06	0.30	0.021	59
NN (5000 epochs)	5.2	0.04	0.31	0.021	66

Concerning the controllers capable of learning, the SOC presents a good strategy to improve performance with a relatively low computational cost. Besides, its on-line learning capability as well as its speed of learning make it very atractive. The neural-net-based approach is very powerful but very time consuming (off-line training) and requiring large computational resources (floating-point calculations). Thus, it is not well suited to be used with low processing power microcontrollers.

The authors are currently working, together with students, in the construction of a new line-tracking robot (RUA v3.0). Several controllers will be used and experimental data will be collected in order to verify the results presented in this article and obtained with the simulator described in section 2. Particular attention will be given to the use of the SOC controller.

References

[1] Almeida L., 1997. Modelização de Pequenos Robots Autónomos, *Revista do Dep. de Electrónica e Telecomunicações da Universidade de Aveiro*, Aveiro, Portugal, Vol. 2, no. 1.

[2] Sampei M. *et al.*, 1995. Arbitrary Path-Tracking Control of Articulated Vehicles Using Non-Linear Control Theory, *IEEE Transactions on Control Systems Technology*, Vol. 3.

[3] Vieira J. A. *et al*, 1997. Moliceiro - Um Robot que Segue Uma Linha, *Revista do Dep. de Electrónica e Telecomunicações da Universidade de Aveiro*, Aveiro, Portugal, Vol. 1, no. 7.

[4] Rausis K. *et al.*, 1997. Obstacle Avoidance Control of a Mobile Robot, *Proceedings of the IFAC - SICICA '97 Symposium*, Annecy, France.

[5] Jan Jantzen, Fuzzy Control, Lecture notes in On-Line Process Control, Technical University of Denmark, 1991.

[6] Jan Jantzen, Fuzzy Control Course On The Internet, Technical University of Denmark, http://www.iau.dtu.dk/~jj/learn/index.html.

[7] Nøgaard, 1997a. *Neural Network Based System Identification Toolbox*, Tech.Rept. 97-E-851, Dep. Automation, Technical University of Denmark.

[8] Nøgaard, 1997b. *Neural Network Based Control System Design Toolkit*, Tech.Rept. 96-E-830, Dep. Automation, Technical University of Denmark.