Adaptive Control System based on Linear Control Theory for the Path-Following Problem of a Car-Like Mobile Robot

J. L. Sanchez-Lopez P. Campoy M. A. Olivarez-Mendez I. Mellado-Bataller D. Galindo-Gallego

Centre for Automation and Robotics - Polytechnic University of Madrid, C/ Jose Gutierrez Abascal, 2. 28006 Madrid Spain (e-mail: {jl.sanchez, pascual.campoy}@upm.es)

Abstract: The objective of this paper is to design a path following control system for a car-like mobile robot using classical linear control techniques, so that it adapts on-line to varying conditions during the trajectory following task. The main advantages of the proposed control structure is that well known linear control theory can be applied in calculating the PID controllers to fulfil control requirements, while at the same time it is flexible to be applied in non-linear changing conditions of the path following task. For this purpose the Frenet frame kinematic model of the robot is linearised at a varying working point that is calculated as a function of the actual velocity, the path curvature and kinematic parameters of the robot, yielding a transfer function that varies during the trajectory. The proposed controller is formed by a combination of an adaptive PID and a feed-forward controller, which varies accordingly with the working conditions and compensates the non-linearity of the system. The good features and flexibility of the proposed control structure have been demonstrated through realistic simulations that include both kinematics and dynamics of the car-like robot.

Keywords: Adaptive control, Autonomous mobile robots, PID control, Linear control systems

1. INTRODUCTION

The motion control of wheeled mobile robots has attracted a remarkable attention during past years [Kolmanovsky and McClamroch (1995)]. As defined in [Morin and Samson (2008)], given a curve on the plane (defined with its curvature c), a non-zero longitudinal velocity for the robot chassis (u_1), and a point of interest attached to this chassis (P), the goal of this problem is to make P move on the curve with velocity u_1. For years, the most studied robot model was the unicycle-type (defined in [Champion and Chung (2008)]) [L.E. Aguilar and Fleury (1998)], [K. Shojaei and Tabibian (2010)], but in the last years, the interest of car-like robots has increased thanks to the direct application to the automotive sector [Mellodge and Kachroo (2007)] and [J. Villagra and de Pedro (2010)]. As we will see, the model that represent the motion of the robot is very non-linear and most commonly non-linear control techniques are used; some based on the model of the robot [L.E. Aguilar and Fleury (1998)], [Morin and Samson (2008)] and others based on a transformation of the model [Mellodge and Kachroo (2007)]. An important aspect of this non-linear controller is its little flexibility, i.e. if the model changes slightly, the controller has to be completely redesigned (a new Lyapunov function must be found). On the other hand, with this kind of controller it is harder to adjust the control objective. To the knowledge of the authors, there are few works about path following for car-like mobile robots based on linear control techniques. The advantage of using such techniques is the simplicity to define a control objective due to the very widely extended knowledge about linear control and PID controllers [Astrom and Hagglund (2006)]. In addition if the model changes, adjusting a new controller is relatively easy. The main drawback of linear techniques is that the designed controller only works well around the selected linearisation point.

In this paper an adaptive control scheme will be designed with linear control techniques to solve the path following problem for a car-like mobile robot. The scheme is formed by and adaptive PID working between a variable point to compensate the known disturbances (like a feed-forward controller). In this way, we benefit of the advantages of the linear control, while we avoid its drawbacks. We will begin the paper selecting the robot model that we will use (section 2). Then, in section 3, we will analyse this model, linearising it and obtaining a linear time variant (LTV) model, that depends on an equilibrium point that we will also calculate (section 3.1). From this point we will be able to get the transfer function (TF) with variable coefficients that represent the problem that we want to control. We will work first of all, in continuous time (section 3.2) and then we will pass to discrete time (section 3.3) because we will assume that the sensors of the robot are visual cameras working at low-frequency (around $30Hz$). Once we get the TF of the system, we will proceed to design the control structure. We will develop an adaptive PID controller working between a variable equilibrium point (section 4). We will use the root locus (RL) to adjust dynamically the PID parameters. After that, in section

6, we will show how this controller works under two kinds of testbed simulations (defined in section 5) to prove its goodness. In section 7, we will discuss some limitations of the original model, and how we can solve these problems with the same design method. Finally, section 8 concludes the paper.

2. PROBLEM STATEMENT

For the whole development of this paper, we selected the widely used Frenet-frame kinematic model of a car-like robot defined in [Morin and Samson (2008)].

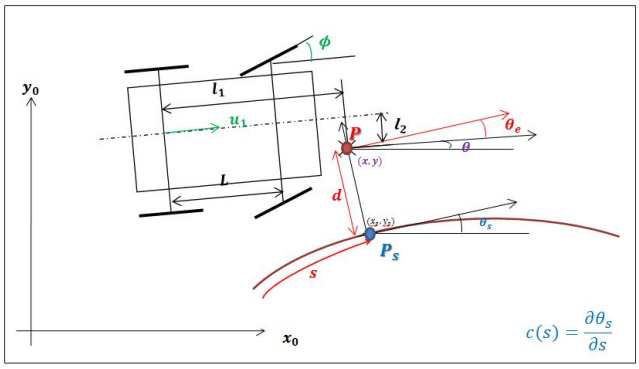

Fig. 1. Model Representation

The robot model used is (see figure 1):

$$\dot{s} = \frac{u_1}{1 - d \cdot c(s)} \cdot \cdot [\cos\theta_e - \frac{\tan\phi}{L} \cdot (l_2 \cdot \cos\theta_e + l_1 \cdot \sin\theta_e)] \quad (1)$$

$$\dot{d} = u_1 \cdot [\sin\theta_e + \frac{\tan\phi}{L} \cdot (l_1 \cdot \cos\theta_e - l_2 \cdot \sin\theta_e)] \quad (2)$$

$$\dot{\theta}_e = \frac{u_1}{L} \cdot \tan\phi - \dot{s} \cdot c(s) \quad (3)$$

Where: P is the point of interest attached to the robot; P_s is the orthogonally projection of P on the circuit (the origin of the Frenet-frame); d is the distance between P and the circuit (P_s); θ_e is the angle between the longitudinal axis of the vehicle and the tangent to the circuit in P_s; s is the progress on the circuit; $c(s)$ is the curvature of the circuit in P_s, u_1 is the linear velocity of the robot; ϕ is the angle of the steering wheels; L is the distance between the robot axis; and l_1 and l_2 is the position of P.

We can simplify the model by combining equations (1) and (3), and for simplicity, we assume $L, l_1 > 0$ and $l_2 = 0$, getting:

$$\dot{d} = u_1 \cdot [\sin\theta_e + \frac{\tan\phi}{L} \cdot l_1 \cdot \cos\theta_e] \quad (4)$$

$$\dot{\theta}_e = \frac{u_1}{L} \cdot \tan\phi - u_1 \cdot \frac{c(s)}{1 - d \cdot c(s)} \cdot [\cos\theta_e - \frac{\tan\phi}{L} \cdot l_1 \cdot \sin\theta_e] \quad (5)$$

Other assumptions that we will use are the following: (1) the robot always moves forward with a known velocity ($u_1 > 0$), (2) this velocity does not change abruptly, (3) occasionally the curvature (c) is known but it is not strictly necessary, (4) this curvature can be discontinuous but it has to be defined in the whole curve, (5) the angle of the robot with the curve has to be small ($\theta_e < \frac{\pi}{2}$) and the distance of the robot to the line will be small ($d \approx 0$).

3. MODELLING AND ANALYSIS

3.1 Model linearisation

We begin linearising the model equations (4), and (5) around an equilibrium point u_{1-lin}, θ_{e-lin}, ϕ_{lin}, c_{lin} and d_{lin}:

$$\Delta\dot{d} = b_1 \cdot \Delta u_1 + b_2 \cdot \Delta\theta_e + b_3 \cdot \Delta\phi \quad (6)$$

$$\Delta\dot{\theta}_e = b_4 \cdot \Delta u_1 + b_5 \cdot \Delta\phi + b_6 \cdot \Delta\theta_e + b_7 \cdot \Delta d + b_8 \cdot \Delta c \quad (7)$$

Where b_i [1] only depends on the equilibrium point and all the defined constants.

By hypothesis, we define three (of five) components of the linearisation point: first, $d_{lin} = 0$ because we want to work at $d_{ref} = 0$; second, $u_{1-lin} = u_{1-lin}(t)$ is a known value; and third, $c_{lin} = c_{lin}(t)$, is a variable value that depends on the circuit and, in the best case, we know it. To get the expression of the other component of the linearisation point, we use the non-linear equations (4) and (5) evaluated in the equilibrium point:

$$\dot{d}_{lin} = u_{1-lin} \cdot [\sin\theta_{e-lin} + \frac{\tan\phi_{lin}}{L} \cdot l_1 \cdot \cos\theta_{e-lin}] \quad (8)$$

$$\dot{\theta}_{e-lin} = \frac{u_{1-lin}}{L} \cdot \tan\phi_{lin} - u_{1-lin} \cdot \frac{c_{lin}}{1 - d_{lin} \cdot c_{lin}} \cdot \cdot [\cos\theta_{e-lin} - \frac{\tan\phi_{lin}}{L} \cdot l_1 \cdot \sin\theta_{e-lin}] \quad (9)$$

Another hypothesis that is assumed is $\dot{d}_{lin} = 0$. This means that the distance to the curve does not change (it is our control goal). It implies that considering the functional restrictions $u_{1-lin} \neq 0$ and $|\theta_e| < \frac{\pi}{2}$ equation (8) becomes:

$$\phi_{lin} = \arctan(-\frac{L}{l} \cdot \tan\theta_{e-lin}) \quad (10)$$

If we only care of the final value of the equilibrium points, we use the hypothesis $\dot{\theta}_e = 0$, and then we get the final value of the last component of the equilibrium point:

$$\theta_{e-lin} = -\arcsin(c_{lin} \cdot l_1) \quad (11)$$

To obtain the real value of the linearising point, we have to solve the equation (9) according to the rest of the equilibrium point, and using the final values calculated above. Using the MATLAB®symbolic engine for equation integrations, we obtain:

$$\theta_{e-lin}(t) = \arcsin(e^{-\frac{u_{1-lin}}{l_1} \cdot t} \cdot \cdot (c_{lin-final} - c_{lin-init}) \cdot l_1 - c_{lin-final} \cdot l_1) \quad (12)$$

Finally we can simplify the value of the b_i coefficients, adding the linearisation point conditions (10):

$$\Delta\dot{d} = b_2 \cdot \Delta\theta_e + b_3 \cdot \Delta\phi \quad (13)$$

$$\Delta\dot{\theta}_e = b_4 \cdot \Delta u_1 + b_5 \cdot \Delta\phi + b_7 \cdot \Delta d + b_8 \cdot \Delta c \quad (14)$$

[1] More info about all coefficients can be found on www.vision4uav.es

3.2 Continuous transfer function

The continuous transfer function (CTF) that represents the model is $G(s) = \frac{\Delta d}{\Delta \phi}$. We operate with equations (13) and (14) in the Laplace domain getting the following CTF:

$$\Delta d = G(s) \cdot \Delta\phi + P_{u_1}(s) \cdot \Delta u_1 + P_c(s) \cdot \Delta c \quad (15)$$

Where the main CTF that represents the system is

$$G(s) = \frac{\Delta d}{\Delta \phi} = A_1 \cdot \frac{s + A_2}{s^2 + A_3} \quad (16)$$

And the CTFs that represent the disturbances are:

$$P_{u_1}(s) = \frac{\Delta d}{\Delta u_1} = \frac{A_4}{s^2 + A_3} \quad (17)$$

$$P_c(s) = \frac{\Delta d}{\Delta c} = \frac{A_5}{s^2 + A_3} \quad (18)$$

Where coefficients A_i depend on b_i.

Now, the system is modelled as shown in figure 2.

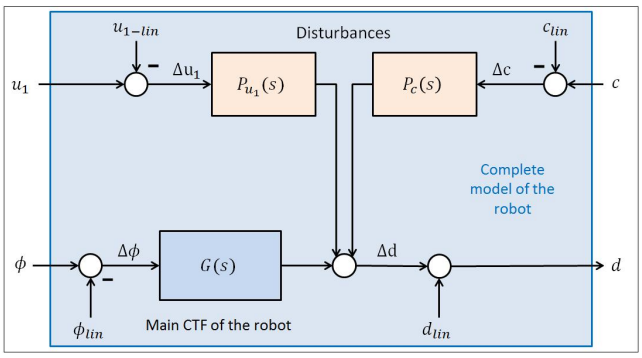

Fig. 2. System Representation

We can work with the system in three different ways:

(1) We can use a constant linearisation point $u_{1-lin} = cte$, $c_{lin} = cte$ and $\theta_{e-lin} = cte \Rightarrow \phi_{lin} = cte$. In this case, the CTF will be linear time invariant (LTI) but if we are not in the conditions of the linearisation point, $G(s)$ will not represent the system well and the modelled disturbances (c and u_1) will affect the robot model.
(2) The second option is to choose a variable linearisation point in case it is known $u_{1-lin} = u_1$, $c_{lin} = c$ and $\theta_{e-lin} \approx \theta_{e-lin}(t) \Rightarrow \phi_{lin} \approx \phi_{lin}(t)$. Now, the system may be simplified to $\Delta d = G(s) \cdot \Delta\phi$, but $G(s)$ has variable coefficients. The system is know LTV and there are no modeled disturbances.
(3) The third option is a mixed case in which we know the value of u_1 but not the value of c. The system is LTV and there are modeled disturbances. This is the most usual case.

Usually in classic control, the chosen option is the first one but for this extremely non-linear problem, this option does not work well. Whenever it is possible, we will work in the conditions of the second option, but it can be common to work in the conditions of the third possibility.

To conclude this section, we are going to analyse the CTF $G(s)$. Our intention is to understand it better in order to design the best control structure in the next section. The system has two poles given by:

$$p_{1,2} = \pm\sqrt{-A_3} = \pm|c_{lin} \cdot \frac{u_{1-lin}}{\cos\theta_{e-lin}}| \cdot j \quad (19)$$

And one zero in

$$z_1 = -A_2 = -\frac{u_{1-lin}}{l_1 \cdot \cos^2\theta_{e-lin}} \cdot (1 + c_{lin} \cdot l_1 \cdot \sin\theta_{e-lin}) \quad (20)$$

Under the conditions of operation ($u_{1-lin} > 0$ and $|\theta_{e-lin}| < \frac{\pi}{2}$), the poles (see equation (19)) are on the imaginary axis and they will move towards the point $(0, 0)$ according to the growth of $|c_{lin}|$, $|u_1|$ and $|\theta_{e-lin}|$. The zero (see equation (20)) is on the real negative axis and it also moves (see figure 3).

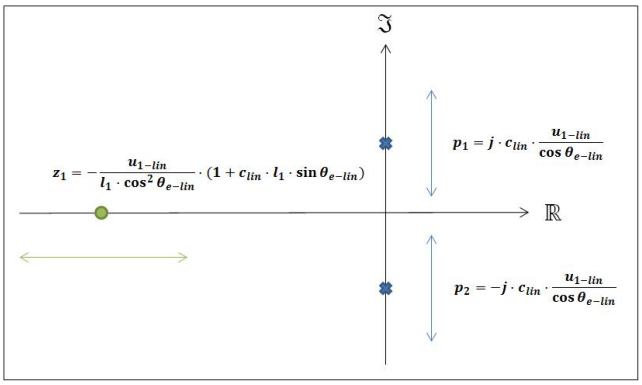

Fig. 3. Position of zero and poles if the system is continuous

3.3 Discrete transfer function

As we said in section 1, we work with discrete sensors like digital cameras. Therefore we have to work in discrete time. We assume a sampling period T. We discretise the continuous model $G(s)$ of section 3.2 with the exact residuals method (Aracil and Jimenez (1993)), getting:

$$B_0 G(z) = \frac{\Delta d_k}{\Delta \phi_k} = B_1 \cdot \frac{z + B_2}{z^2 + B_3 \cdot z + B_4} \quad (21)$$

where B_i depends on A_i and T.

The discrete system has the same number of zeros and poles than the continuous one (saw in section 3.2) and they move analogously. But now instead above the imaginary axis, the poles do it above the unit circle, and the zero instead of moving above the negative half of the complex plane, it do it above the real axis inside unit circle.

4. CONTROL SCHEME

To design the control system, the model of section 3 will be used. We assume that the velocity of the robot (u_1) is externally given and known by us, and we can only act on the steering of the robot. We will close the loop with the value of the distance to the curve (d), obtained with a sensor (in some case it can be discrete). We occasionally know the value of the curvature of the curve (c). The error to the line is given by $\varepsilon = d_{ref} - d$ (and for our problem, we assume $d_{ref} = 0$).

The designed control scheme (figure 4) is an adaptive PID controller ($R(s) = \frac{\Delta\phi}{\Delta\varepsilon}$), working between a theoretical and variable linearisation point. This point is calculated in section 3 and the objective of making it variable is to compensate with a theoretical equation, the known disturbances as the feed-forward controller would do. The adaptive PID has the objective of compensate the error that is not considered in the equilibrium point. The input of the PID controller is given by $\Delta\varepsilon = \varepsilon_{lin} - \varepsilon = \Delta d = d$, assuming $d_{ref-lin} = d_{lin} = 0$. The output is $\Delta\phi$, and we have to transform it to the input of the robot making $\phi = \Delta\phi + \phi_{lin}$.

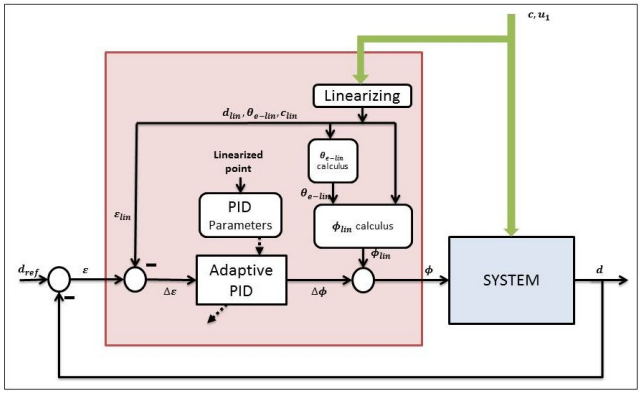

Fig. 4. Control Structure: adaptive PID working between a variable point to compensate the disturbances

4.1 Continuous controller

First of all, we are going to design a continuous controller to control the LTV system modelled in (16). We can design the PID controller with the criterion that we want (RL, Bode diagram, ..). In this paper, we will use the RL to design it. As the system is modelled as a LTV, the RL will change continuously. Therefore the parameters of the PID will change with it.

From the observation of the RL, we can extract that the system can be stably controlled with a P controller, but it will have position and tracing error due to the model disturbances (because of modelling errors or simply because we do not know the value of the curvature). To avoid this source of errors, we design a PI controller:

$$R(s) = \frac{\Delta\phi}{\Delta\varepsilon} = K_C \cdot \frac{(s+a)}{s} \tag{22}$$

With this method we can put the zeros and the poles of the system where we want (see figure 5). We select the following method for designing the PID (suggested in [Matia et al. (2003)]):

First of all, we put the zero of the controller $(-a)$ to a third of the place where the dominant poles are if we use a P controller and we want them to be pure real and equal:

$$a = \frac{1}{3} \cdot (A_2 + \sqrt{A_2^2 + A_3}) \tag{23}$$

Then, the controller gain (K_C) is calculated to put the poles of the system in the pure real zone. The characteristic polynomial of the close loop system is $p(s) = s \cdot (s^2 + A_3) + K_C \cdot A_1 \cdot (s+a) \cdot (s+A_2)$. The points of the complex

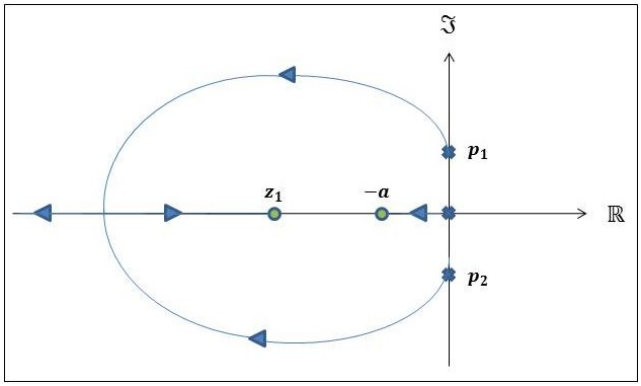

Fig. 5. Continuous Root Locus with a PI controller

plane belonging to the RL, satisfy $p(s) = 0$, and we get computationally the point where the RL converges $(-d_1)$. Then, we get the value of the gain of the PI controller:

$$|K_C| = \frac{1}{|A_1|} \cdot \frac{d_1 \cdot (d_1^2 + A_3)}{(d_1 - a) \cdot (d_1 - A_2)} \tag{24}$$

The system is stable because we have designed that way with the RL method, but we can also prove it with the Routh-Hurwitz criterion [Matia et al. (2003)]. In the transitions of the linearising point the system might not be stable. As we assume that the linearisation point will change slowly, we assume that there will not be stability problems in the transitions. In section 6 we will see that despite the linearisation point change, the controller works well.

4.2 Discrete controller

Once we have designed the continuous controller, we can design the discrete one. We select a similar structure with a discrete PI, working between the linearisation point. Now the linearisation point will be sampled. The design of the discrete adaptive PI can be done in two ways. The first one is to use the discrete RL in the same way that in the section above. This is quite difficult. We prefer to discretise the continuous PI controller by doing:

$$R(z) = \frac{\Delta\phi_k}{\Delta\varepsilon_k} = R(s = \frac{z-1}{T \cdot z}) = K_{Cd} \cdot \frac{z - a_d}{z - 1} \tag{25}$$

where:

$$K_{Cd} = K_C \cdot (a \cdot T + 1) \tag{26}$$

$$a_d = \frac{1}{a \cdot T + 1} \tag{27}$$

In that chase, we may have stability problems. We use the Jury criterion [Aracil and Jimenez (1993)] to prove the stability of the feedback system.

5. TESTBED

We will use two testbeds to test the controller. The first one is a C program that simulates the non-linear kinematic model defined by equations (1), (2), and (3). It will be sampled at $f = 29Hz$ because we assume that the robot has a face down camera to get the distance to the circuit. The circuit was a line painted on the floor. To make it more

realistic, white noise is added. The dimensions of the robot are the same of a Citron C3 Pluriel ($L = 2.46m$). The sensor will be placed on the front of the car and centred ($l_1 = 3.41m$ and $l_2 = 0$) The simulated circuit is shown in figure 6. The robot (car) will move with a variable and

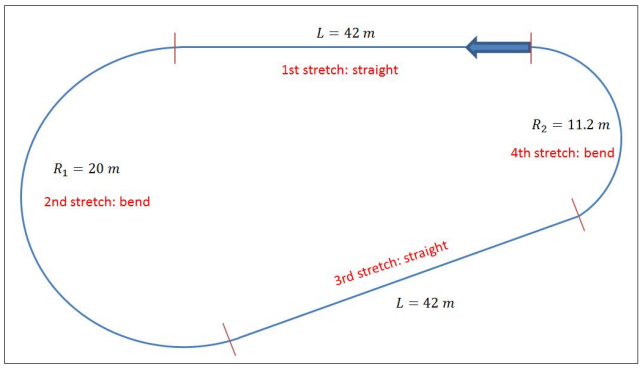

Fig. 6. Circuit of the first testbed. The robot moves in the direction of the arrow and starts from it

known velocity u_1 shown in figure 7. We may know the value of the curvature c according to the kind of test that we want to do. This first simulation has the objective to prove that the designed controller works well against its model with adverse conditions (noise, variable velocity).

The second testbed is a modification of the autonomous vehicle included in the examples of Webots. This example is a realistic model of a BMW X3 that includes not only kinematics but also dynamics. A face down camera was put in the front center of the vehicle to get the distance to the circuit, which is a painted line on the floor. For simplicity, the car moves at $15Km/h$ (constant). In this case, we want to test the controller in very adverse conditions. Therefore, we will not know the value of the curvature c and we will add a deliberately strong noise to the camera acquisition (especially in the curve) and an error in the measurements of L and l_1. With this simulation we will test the controller in very adverse conditions with all kinds of errors in the model and noise.

6. RESULTS

In this section we will evaluate the controller designed in section 4 with the simulation results. In the first section (section 6.1) we will use the first testbed defined in section 5, and in the section 6.2 we will use Webots as testbed.

6.1 Model simulation

We will show two kinds of simulations. In the first one, we assume that we know the value of the curvature c, and

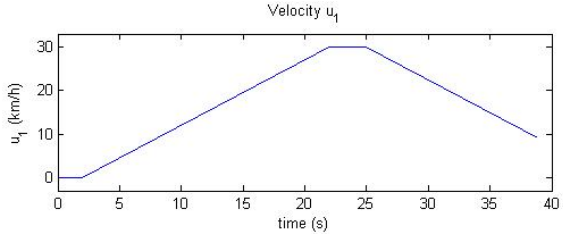

Fig. 7. Velocity u_1 in circuit number one

we calculate the controller using this information. In the second, we assume that we ignore this value. We establish that, during all the simulation, we are in a $c = 0$ curve zone, and we allow the integer action of the controller solve this error.

In figure 8 we can show the steering action done for the whole controller, and the obtained error (distance to the curve) when the curvature is known.

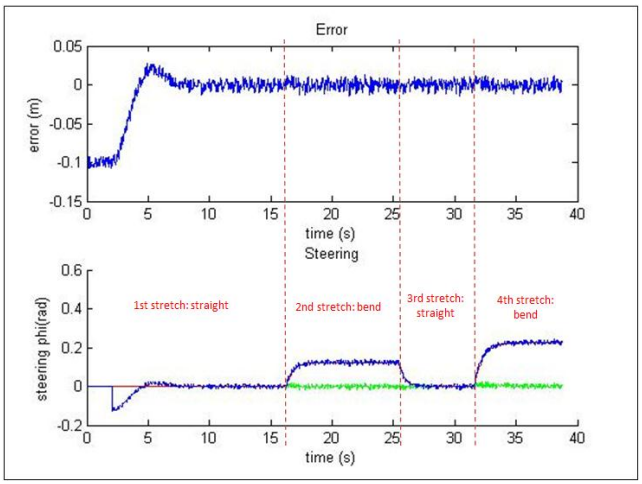

Fig. 8. Simulation knowing c: In the upper plot, the evolution of the error is depicted. In the lower one the steering is shown: ϕ (blue), ϕ_{lin} (red) and $\Delta\phi$ (green)

In figure 9, the same variables are plotted but now ignoring the value of the curvature of the curve.

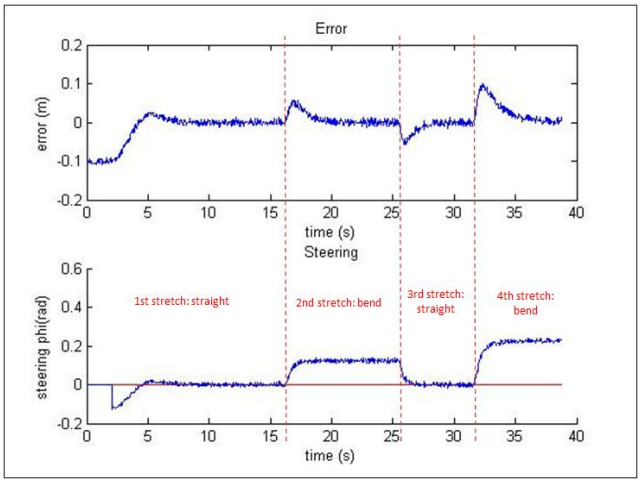

Fig. 9. Simulation ignoring c: In the upper plot, the evolution of the error is depicted. In the lower one the steering is shown: ϕ (blue), ϕ_{lin} (red) and $\Delta\phi$ (green)

We can see that the maximum error obtained is lower than $0.03m$ in the case of knowing the curvature and lower than $0.1m$ if we do not know it. The error is very small (compared to the size of the robot) and is stabilized on zero even when the velocity u_1 change continuously to $30Km/h$. The simulation results show that the controller

extensively meet the requirements for the proposed application, even in very closed curves (the smallest one has a radius of $11.2m$). The performance is also very good without knowledge of the curvature, that means $\phi_{lin} = 0$ and the PID controller has to do all the control action $\phi = \Delta\phi$. However it produces a peak in the error when we ignore c, but the controller compensates it. The noisy measurements do not seem to affect the performance of the controller either.

6.2 Webots simulation

A graphic with the error and the steering given by the simulation is shown in figure 10.

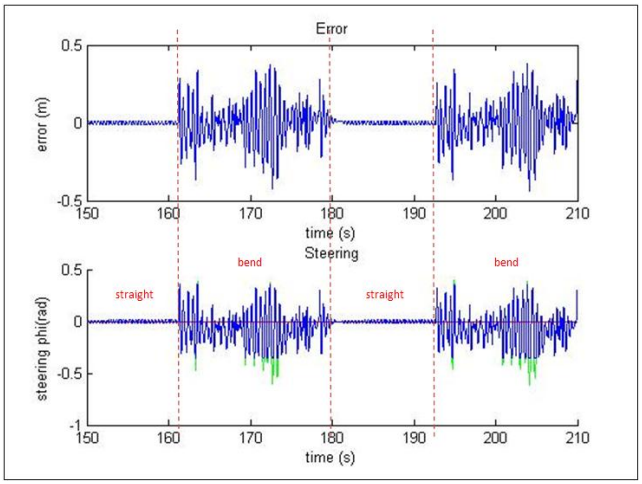

Fig. 10. Simulation with Webots: In the upper figure, the error. In the lower the steering: ϕ (blue), ϕ_{lin} (red) and $\Delta\phi$ (green)

Now the error is bigger than before, because we included a lot of intentional errors to put the controller to the test and a very big noise (bigger in bend stretches). However, the controller stabilizes the error in zero with a maximum peak of $0.5m$ (it remains small compared with the size of the robot). Therefore, the simulation demonstrates that the control scheme designed along the paper works very well even with a realistically simulated model with big noise, dynamics and intentional errors.

7. EXTENSION

Throughout the paper, we have done some hypotheses that must be clarified before conclusion. First of all, we assume that the model of the robot is the one described in equations (1), (2) and (3). This might not be real. The system could have, for instance, steering dynamics. This is not a problem. With the same design method, we can identify a CTF (either LTV or LTI) that we can connect in cascade with the robot model. The new RL, may have new poles or zeros, and the new controller may be different, but the design method is the same. Secondly, we assume that $l_2 = 0$ and the objective of the control is $d_{ref} = 0 \Rightarrow d_{lin} = 0$. We can remove this hypothesis at the expense of complicating the calculation. Also, if we want to move the robot backwards ($u_1 < 0$), we have to stabilize the system with other method (for example state feedback), because moving backwards puts a zero in the positive real axis that with the classical control theory cannot be stabilized.

8. CONCLUSION

We designed a control scheme based on linear techniques with all their advantages (flexibility and wide knowledge), but avoiding its disadvantages by using an adaptive structure. This adaptive structure is composed of a PID with variable parameters calculated on-line; and a linearisation point with a variable theoretical equation that complements the PID like a feed-forward control would do. This controller is based on a linearisation of the kinematic Frenet frame model of a car-like mobile robot. The adequate performance of this controller was demonstrated by two different kinds of simulations with intentional bad conditions.

ACKNOWLEDGEMENTS

The authors want to thank the Spanish "Ministerio de Educacin y Ciencia" MEC for granting a Research Scholarship to the main author and also for the research project DPI2007-66156. The authors would also like to thank Siemens Espaa S.A. that has established the industrial requirements of some parts of this project. We would finally like to thank ISA-Spain for awarding a distinction to the main author for his Master's Degree Final Thesis.

REFERENCES

Aracil, R. and Jimenez, A. (eds.) (1993). *Sistemas discretos de control*. Seccion de publicaciones de la ETSII-UPM.

Astrom, K.J. and Hagglund, T. (eds.) (2006). *Advanced PID control*. ISA.

Champion, G. and Chung (2008). Wheeled robots. In B. Siciliano and O. Khatib (eds.), *Handbook of Robotics*, volume 1, 391–410. Springer, Germany, 1st edition.

J. Villagra, V. Milanes, J.P. and de Pedro, T. (2010). Control basado en pid inteligentes: aplicacion al control de crucero de un vehiculo a bajas velocidades. *Revista Iberoamericana de Automatica e Informatica Industrial*, 7, 44–52.

K. Shojaei, A.M.S. and Tabibian, B. (2010). Adaptive-robust feedback linearizing control of a nonholonomic wheeled mobile robot. *IEEE/ASME International Conference on Advance Intelligent Mechatronics*, 497–502.

Kolmanovsky, I. and McClamroch, H. (1995). Developents in nonholonomic control problems. *Proceedings of the IEEE Control Systems*, 15, 20–36.

L.E. Aguilar, P.Soueres, M.C. and Fleury, S. (1998). Robust path following control with exponential stability for mobile robots. *International conference on Robotics and Automation*, 3279–3284.

Matia, F. et al. (eds.) (2003). *Teora de sistemas*. Seccion de publicaciones de la ETSII-UPM.

Mellodge, P. and Kachroo, P. (2007). Open-loop vehicle control using an abstraction of its model. *American Control Conference*, 3023–3028.

Morin, P. and Samson, C. (2008). Motion control of wheeled mobile robots. In B. Siciliano and O. Khatib (eds.), *Handbook of Robotics*, volume 1, 799–826. Springer, Germany, 1st edition.