

UNIVERSIDADE ESTADUAL DE CAMPINAS  
SISTEMA DE BIBLIOTECAS DA UNICAMP  
REPOSITÓRIO DA PRODUÇÃO CIENTÍFICA E INTELECTUAL DA UNICAMP

**Versão do arquivo anexado / Version of attached file:**

Versão do Editor / Published Version

**Mais informações no site da editora / Further information on publisher's website:**

<https://link.springer.com/article/10.1007/s40534-018-0160-3>

**DOI: 10.1007/s40534-018-0160-3**

**Direitos autorais / Publisher's copyright statement:**

©2018 by Springer. All rights reserved.

DIRETORIA DE TRATAMENTO DA INFORMAÇÃO

Cidade Universitária Zeferino Vaz Barão Geraldo


CEP 13083-970 – Campinas SP

Fone: (19) 3521-6493

<http://www.repositorio.unicamp.br>



# The VILMA intelligent vehicle: an architectural design for cooperative control between driver and automated system

Olmer Garcia<sup>1</sup>  · Giovani Bernardes Vitor<sup>2</sup> · Janito Vaqueiro Ferreira<sup>3</sup> · Pablo Siqueira Meirelles<sup>3</sup> · Arthur de Miranda Neto<sup>4</sup>

Received: 26 September 2017 / Revised: 5 April 2018 / Accepted: 9 April 2018 / Published online: 26 April 2018  
© The Author(s) 2018

**Abstract** Intelligent autonomous vehicles have received a great degree of attention in recent years. Although the technology required for these vehicles is relatively advanced, the challenge is firstly to ensure that drivers can understand the capabilities and limitations of such systems and secondly to design a system that can handle the interaction between the driver and the automated intelligent system. In this study, we describe an approach using different strategies for an autonomous system and a driver to drive a vehicle cooperatively. The proposed strategies are referred to as cooperative planning and control and determine when and how the path projected by the autonomous system can be changed safely by the driver to a path that he wishes to follow. The first phase of the project is described, covering the design and implementation of an autonomous test vehicle. Experiments are carried out with a driver to test the cooperative planning and control concepts proposed here.

**Keywords** Autonomous vehicles · Embedded systems · Cooperative systems · Visual servoing

## 1 Introduction

According to the data published by the United Nations, more than 1.2 million people die on roads around the world every year, and as many as 50 million are injured. Over 90% of these deaths occur in low- and middle-income countries. Brazil is among the states in which the number of such deaths is relatively high. Figure 1 shows historical data for traffic accident deaths in Brazil, the USA, Iran, France, and Germany. There is a significant difference in the figures between developing countries and high-income countries. However, per capita statistics are controversial as the number of people who drive varies between countries, as does the number of kilometers traveled by drivers.

The trend toward the use of automated, semiautonomous, and autonomous systems to assist drivers has received an impetus from significant technological advances and recent studies of accident rates [2]. In parallel, the challenges posed by autonomous and semiautonomous navigation have motivated researchers from different groups to research this area. One of the most important issues when designing an autonomous vehicle [3] or a driver assistance system is vehicle safety and security.

Here, we investigate cooperative control in automated cars, where the driver is sometimes in the control loop, and the vehicle is operating at level 3 automation [4]. The most common strategy for transferring control to the driver, particularly in risk situations, is to use an emergency button. However, in practice, this may have serious drawbacks. For example, Google patent [5], which describes a system in which all the security variables are checked before control, is transferred.

Cooperative control also addresses the problem of driver inattention, as Ref. [6] present. Jain et al. [7] use an autoregressive input–output hidden Markov model to

---

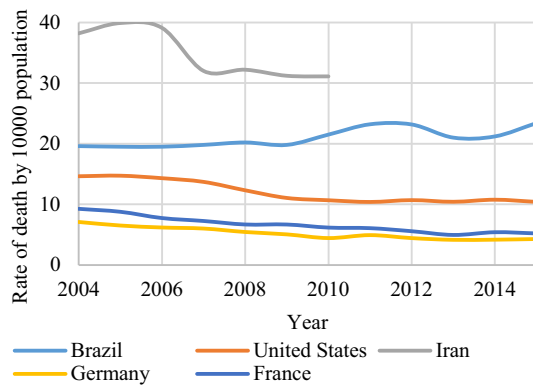
✉ Olmer Garcia  
olmer.garciab@utadeo.edu.co

<sup>1</sup> Universidad de Bogota Jorge Tadeo Lozano, Bogota, Colombia

<sup>2</sup> Robotics, Intelligent and Complex Systems (RobSIC) Lab, Universidade Federal de Itajubá (UNIFEI), Itabira, MG, Brazil

<sup>3</sup> Autonomous Mobility Lab. at the Universidade Estadual de Campinas, (UNICAMP), Campinas, SP 13083-860, Brazil

<sup>4</sup> Terrestrial Mobility Lab at the Universidade Federal de Lavras, (UFLA), Lavras, MG 37200-000, Brazil



**Fig. 1** Traffic accident deaths per 10,000 citizens. Sources: Brazil (DATASUS), United States (NHTSA), Iran [1], Germany (destatis.de), and France ([www.securite-routiere.gov.fr](http://www.securite-routiere.gov.fr))

capture contextual information and driver maneuvers a few seconds before they occur and so prevent accidents. Malik et al. [8] describe an intelligent driver training system that analyzes crash risks for a given driving situation, opening up possibilities for improving and personalizing driver training programs. Liu et al. [9] propose a method for predicting the trajectory of a lane-changing vehicle using a hidden Markov model to estimate and classify the driver's behavior. Amsalu et al. [10] introduce a method for evaluating a driver's intention at each time step using a multi-class support vector machine. Although the approaches described in these studies yield satisfactory results, none of them concretely handle cooperative control between automated intelligent systems and a driver. Merat et al. [11] describe tests in a simulator to investigate driver behavior when the driver is resuming manual control of a vehicle operating at a high level of automation. Their study sought to contribute to an understanding of suitable criteria for the design of human-machine interfaces for use in automated driving and so ensure that messages related to the transfer of control are given in a timely and appropriate manner.

The aim of the project described here was to test different cooperative driving strategies where an autonomous system and a driver share the control. The approach, which is intended to help reduce the risk of accidents, is referred to as cooperative planning and cooperative control and determine if and how the path projected by the autonomous system can be changed safely by the driver. The article structure is as follows. Section 2 introduces the concepts and the proposed approach. Section 3 describes the hardware and software architectures implemented in VILMA01 (first intelligent vehicle from the autonomous mobility laboratory). Section 4 discusses the cooperative strategies implemented using visual servoing and road lines. Section 5 discusses the results, and finally, Sect. 6 presents the conclusions and future studies.

## 2 Proposed approach

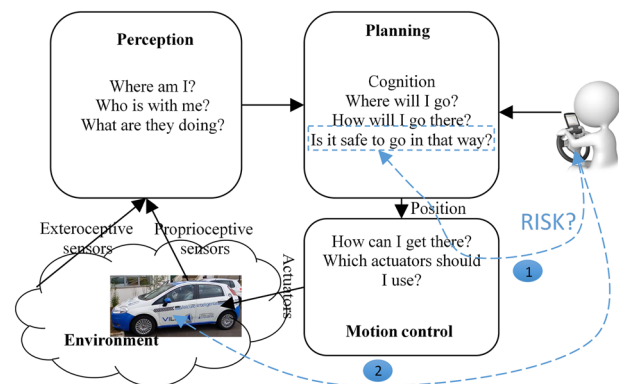
In a typical mobile robotic architecture like that described in Ref. [12], an intelligent, autonomous system consists of three main modules: perception, planning, and motion control. Each module seeks to answer specific questions related to particular tasks performed by the automatic system, as shown in Fig. 2. To handle the interaction between driver and autonomous system, the planning module should include the question “Is it safe to go in that direction?” The need to handle the interaction between both agents (driver and autonomous system) leads to the concepts of cooperative planning and cooperative control in mobile robotics. In the approach proposed here and implemented in VILMA01, a new actuator-based interface between driver and robot (dotted blue lines in Fig. 2) allows managing this cooperative planning and control. The control and planning layers perform a risk analysis for a given path the user wants to take using visual servoing.

In cooperative control, the actuation systems respond to two signals. One is produced by the autonomous system, and the other by the driver. The intelligent system must, therefore, predict the path the driver wants to follow and the risk associated with this path so that it can decide whether to maintain cooperative control or transfer control to the driver. These strategies are detailed in Sect. 4.

The proposal described here assumes that the autonomous system's decisions and perception are reliable, because in emergency conditions always the emergency button is available.

## 3 System architecture

The autonomous robotic vehicle was designed over the concept of layers and functional groups taking account that, this type of architecture facilitates the separation of



**Fig. 2** Layers in the mobile robotics architecture [11]

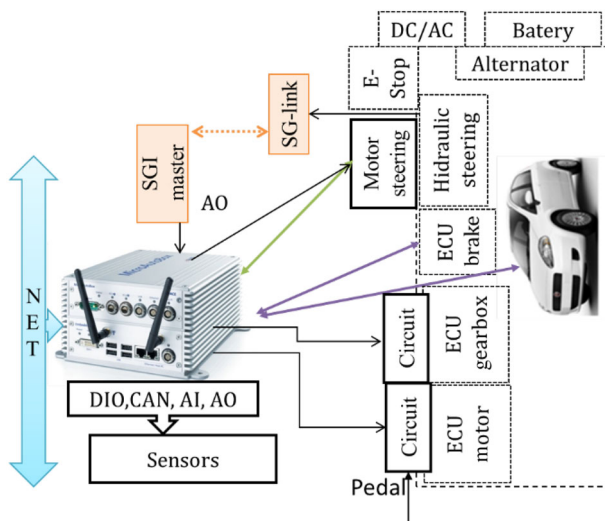
functions and development, allowing for the definition of interfaces between the various subsystems [13]. According to Ref. [14], the six major functional groups are Interface Sensors, Perception, Control, Vehicle Interface, and User Interface. The questions typically asked in each layer of the software were answered by the software architecture.

### 3.1 Automation hardware architecture

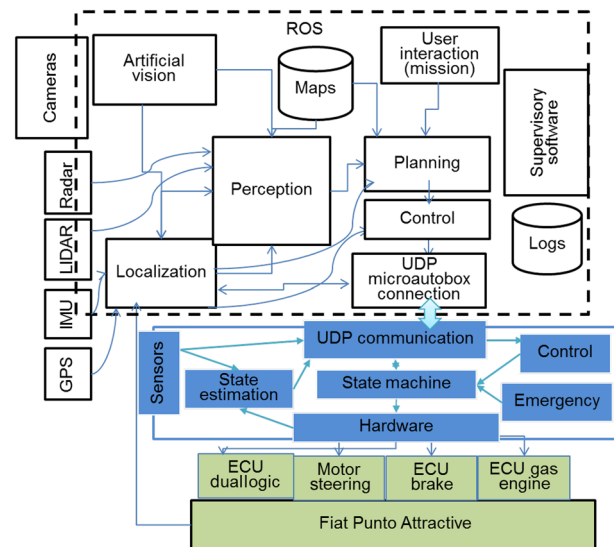
An embedded system was used to control the steering, acceleration, brakes, and gearbox in a Fiat Punto 2011. The steering was controlled by an electric motor coupled to the steering column. No mechanical modifications were made to the steering column to ensure that the vehicle was safe. The three others actuators were controlled through electronic circuits and data networks. Figure 3 shows the functional hardware blocks, where the lines with arrows represent the five different types of interaction between the blocks: the RS232 network (green); AI, AO, and DIO (black) for the actuation commands; the wireless network (orange) for the steering torque sensor; the CAN network (purple), and the Ethernet (Blue) to connect the embedded system to other computer systems. The system hardware includes programming and debugging interfaces.

### 3.2 Framework architecture

Figure 4 shows the architectural framework used. This is divided into three areas (green, blue, and white) according to the programming approach used:



**Fig. 3** Hardware architecture of the automation. Communication: Ethernet (blue), RS232 (green), RF (orange), electric signals (black), and CAN (purple)



**Fig. 4** Framework architecture of VILMA. Linux software (white), firmware (green), and embedded software (blue)

- *Firmware (green)* Software developed directly in the electronic equipment. The only firmware implemented directly in this architecture was the steering motor firmware. The other firmware was third-party code that only needed configuration with the parameters required to work inside the proposed architecture.
- *Embedded system (blue)*. The software implemented in the embedded system was designed for real-time tasks and developed in Simulink. Examples include state machines to interact with the hardware, an emergency routine, routines in the control layer which have real-time requirements, interactions with sensors and localization algorithms to process the sensor outputs. The ControlDesk program, which runs on Linux, allows communication with the embedded system.
- *Linux software (white)*. Software implemented in PC-like architectures distributed on different computers. The Robotic Operating System (ROS) [15] acts as a meta-operating system and a toolbox. It provides communication interfaces between programs (defined as nodes) such as publisher-subscriber interfaces, client-server interfaces, data queues, and many other tools for programming, including clocks and a cyclical task manager. It also provides GUIs with tools to aid software development. These can be used at runtime or with stored data for further processing. ROS is free with the Apache license and can work with programming tools for real-time applications such as OROCOS [16].

Finally, the software architecture included a vehicle simulator integrated into the embedded system (based on a

dynamic model of the vehicle) [17] and in the Linux software (based on a simulator in a 3D environment) [18]. For this, simulator was tested on Linux: Gazebo [19], V-Rep [20], and MORSE [21]. The last of these was selected so that work could in future be carried out in partnership with the research group that developed the CARINA vehicle [22] at the University of São Paulo—Brazil.

Figure 5 shows the nodes used in VILMA01 for the experiments described here. The `vilma_ma_ros` and `vilma_perception` packages developed specifically for the vehicle, while the other packages are available free in ROS. The `vilma_ma_ros` package implements the nodes required to connect specific hardware to ROS, such as the embedded system, which uses the UDP protocol to establish low-latency, loss-tolerating connections. The next section describes the development of the node used to perform cooperative planning based on visual servoing.

#### 4 Cooperative trajectory control based on visual servoing

In this section, the cooperative trajectory control used to follow road lines is described. A ROS node, which implements the control layer, is summarized in the block diagram in Fig. 6. Note that path planning is simulated through a joystick to test cooperative approach.

The various blocks are described below under the following headings: perception, motion control, planning, and cooperative control.

##### 4.1 Perception

In the perception layer, a low-cost IP camera with a line detection algorithm detects the road line. The algorithm can be divided into three steps. First one, a line segment

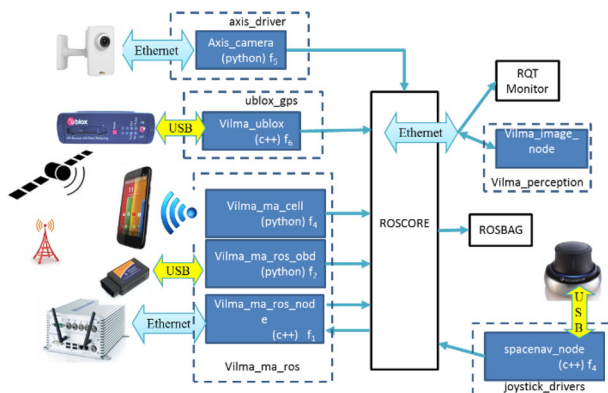


Fig. 5 Nodes executed in ROS for VILMA01

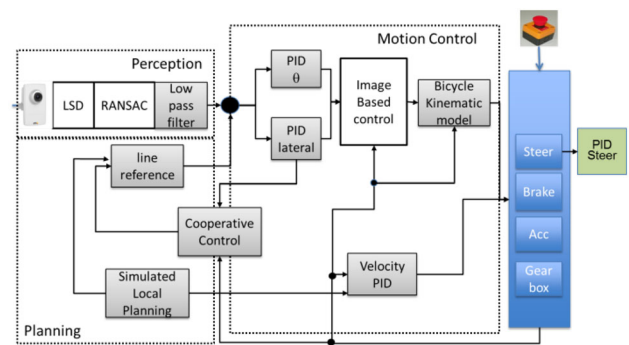


Fig. 6 Block diagram of the cooperative trajectory control based on visual servoing

detector (LSD) [23] is used to generate several lines in the image. Two sets of the lines are obtained according to their slopes. Finally, one of these two sets is used to estimate the desired line using the RANSAC (RANDOM Sample Consensus) algorithm [12, 24] and the PCL (Point Cloud Library) [25]. The estimated line is described by Eq. (1), where  $m_k^{\text{image}}$  is the slope of the line, which determines the orientation error, and  $b_k^{\text{image}}$  is the  $x$  coordinate which cross zero in the image frame and helps to find the lateral error. This point in the image is a heuristic parameter and allows a compromise to be chosen between control stability and smoothness in the response [26].

$$x_k^{\text{image}} = m_k^{\text{image}} y_k^{\text{ref}} + b_k^{\text{image}}. \quad (1)$$

The line parameters are filtered with a discrete low-pass time filter given by

$$\begin{aligned} m_k^{\text{image}} &= m_{k-1}^{\text{image}} (1 - \alpha_m) + \alpha_m m^{\text{RANSAC}}, \\ b_k^{\text{image}} &= b_{k-1}^{\text{image}} (1 - \alpha_b) + \alpha_b b^{\text{RANSAC}}, \end{aligned} \quad (2)$$

where  $\alpha_m = T_i/T_{lpm}$ ,  $\alpha_b = T_i/T_{lpb}$ , and  $T_i$  are the time required to acquire and process the image. Note that  $T_{lpb}$  and  $T_{lpm}$  are heuristics parameters; RANSAC line parameters are the data generated by this algorithm (Fig. 7).

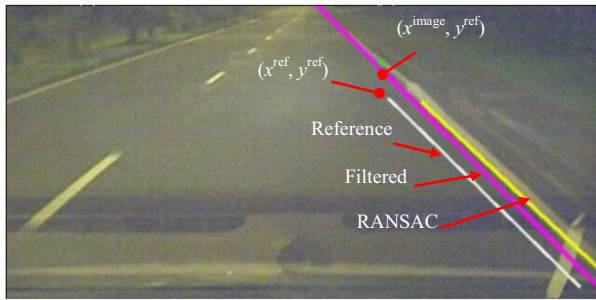
##### 4.2 Planning

In the path planning, the reference is described by a straight line given by

$$x_k^{\text{ref}} = m_k^{\text{ref}} y_k^{\text{ref}} + b_k^{\text{ref}}. \quad (3)$$

The planning layer depends only on the road line and does not take into account extrinsic and intrinsic risks such as large bends or obstacles. To simulate changes in this kind in the path planned, the route changes according to commands from a 3D mouse from 3Dconnexion. The proposed speed can be adjusted using one of the buttons on the mouse to modify the target speed based on a predefined longitudinal acceleration and deceleration curve.





**Fig. 7** Lines used in the image-based control

In the cooperative planning experiments, two other buttons on the 3D mouse are used to simulate the steering maneuver (left or right) performed by the driver. If a lateral change is permitted (i.e., if both buttons are pressed), the torque applied by the driver to the steering will change the lateral distance to the road line. This change is given by

$$b_k^{\text{ref}} = b_{k-1}^{\text{ref}} + \beta_1 \tau^{\text{user}} + \beta_2, \quad (4)$$

where  $\beta_1$  and  $\beta_2$  are heuristic parameters in pixels/Nm and pixels, respectively. When the driver stops steering, the lateral reference returns linearly to its original value ( $b_k^{\text{ref}} = b_0^{\text{ref}}$ ) to smooth the change in the path.

### 4.3 Motion control

Figure 7 shows the straight lines defined for the path control. The yellow line is the line detected by the image processing; the purple one is the line filtered in time in the perception layer and the white one the reference line generated in the planning layer. The ultimate objectives of the path control were that the white and purple lines should have the same orientation and that the lateral error at the reference point  $y^{\text{ref}}$  should be zero.

To achieve two objectives, a controller is implemented in the image frame as described in Ref. [27]. This calculates the yaw speed required by a non-holonomic robot using the generalized camera model. The desired yaw speed is converted to a steering angle using the kinematic model of the vehicle simplified according to the bicycle model [26] and the vehicle speed from the speed control system. To ensure that the lateral error ( $e^l$ ), orientation error ( $e^\theta$ ), and speed error ( $e^{v_x}$ ) converge to zero, three PID controllers were implemented. The output signals from the first two PID controllers are added using Eq. (5) to determine the steering angle, which is controlled by a PID controller in the steering motor firmware. The signal from

the third PID controller determines the percentage braking or acceleration required.

The PID controllers are represented mathematically by

$$u_k^{\{\cdot\}} = k_p e_k^{\{\cdot\}} + \frac{k_d}{T_i} (e_k^{\{\cdot\}} - e_{k-1}^{\{\cdot\}}) + k_p T_i \sum e^{\{\cdot\}}, \quad (5)$$

where  $e^{\{\cdot\}}$  is the error:  $e^\theta = \text{atan}(m^{\text{ref}}) - \text{atan}(m_k^{\text{image}})$ ,  $e^l = x_k^{\text{ref}} - x_k^{\text{image}}$ , or  $e^{v_x} = v_x^{\text{ref}} - v_x$ , and  $u^{\{\cdot\}}$  is the control signal for each error. The controllers have several nonlinearities: firstly, in all the controllers, the integrators are saturated at a particular value, and secondly, the speed controller has a dead zone to ensure smooth braking.

### 4.4 Cooperative control

The first cooperative strategy implemented was the strategy for the speed controller. For safety reasons, when the driver pushes the brake pedal the speed controller is deactivated, but the path controller continues working until the driver presses the emergency button when all the automated control is turned off.

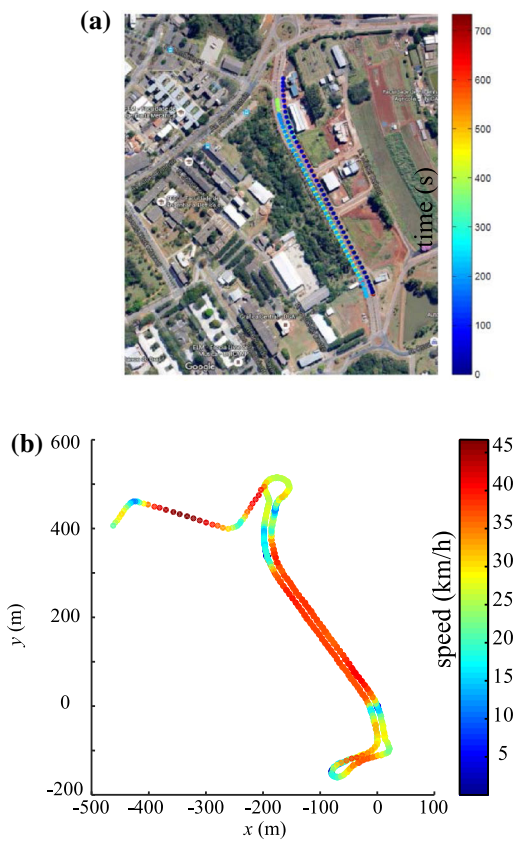
The second strategy implemented was in the path controller. This was a cooperative strategy because the driver can help to achieve zero lateral error by applying torque to the steering wheel. One problem in achieving this objective is that, as identified in simulations [28] and tests in the vehicle with the visual servoing control, any driver torque is perceived as a perturbation by the steering PID controller. The strategy adopted was therefore to change the reference signal  $b_k^{\text{ref}}$  at a rate dependent on driver torque if this was in the same direction as the lateral error and both were outside a dead band.

Finally, the cooperative path control changes the reference line for the planning layer in an attempt to reduce the real lateral error, making it necessary to increase the steering angle calculated by the path controller. Algorithm 1 shows the cooperative control and planning strategy used.

## 5 Experimental results

Figure 8a shows the areas on the map where the visual servoing control was activated, and Fig. 8b shows the longitudinal speed of the car on the test route, transforming the GPS coordinates to meters using the WGS84 global reference system.

Algorithm 1 Cooperative planning and control algorithm.



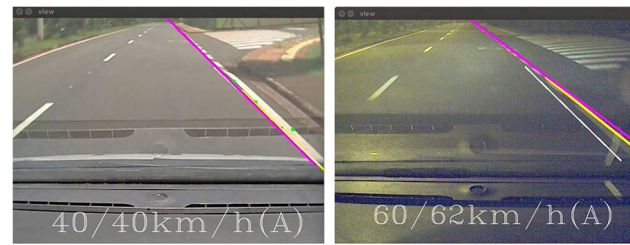
**Fig. 8** Route used in the experimental test. **a** Route used to test cooperative control. The color means the time (s) when the point was reached. **b** Speed of the vehicle in km/h (color) in the test

if *left button* or  $\text{sgn}(\tau^{\text{user}}) = \text{sgn}(e^1)$  and  $|\tau^{\text{user}}| > \tau_{\min}$   
 $b_k^{\text{ref}} \leftarrow b_{k-1}^{\text{ref}} + \beta_1 \text{sgn}(\tau^{\text{user}}) + \beta_2$   
 else if *right button* or  $\text{sgn}(\tau^{\text{user}}) = \text{sgn}(e^1)$   
     and  $|\tau^{\text{user}}| > \tau_{\min}$   
 $b_k^{\text{ref}} \leftarrow b_{k-1}^{\text{ref}} - \beta_1 \text{sgn}(\tau^{\text{user}}) + \beta_2$   
 else  
   if  $|b_k^{\text{ref}} - b_0^{\text{ref}}| > \Delta_b$   
      $b_k^{\text{ref}} \leftarrow b_{k-1}^{\text{ref}} - \beta_1 \text{sgn}(\tau^{\text{user}}) \Delta_b$   
   else  
      $b_k^{\text{ref}} \leftarrow b_0^{\text{ref}}$

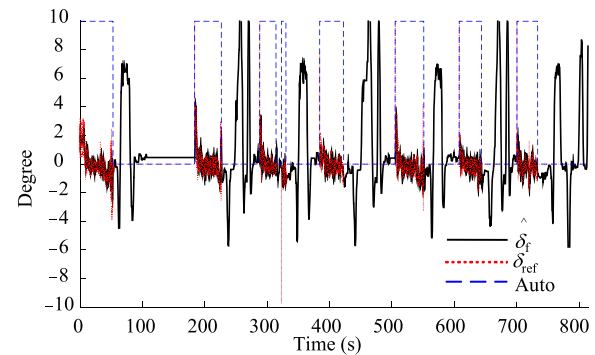
Algorithm 1 Cooperative planning and control algorithm.

The results for line detection by the perception layer in different light levels and at different speeds are given in Fig. 9. These show that the proposed system works satisfactorily under different environmental conditions. One limitation of the system is that the closed loop period in image control is between 20 and 30 Hz limited by the processing of the camera information.

Figure 10 shows the angle of the wheels, which correspond to the control signal, was estimated from the position



**Fig. 9** Perception tests under different light conditions

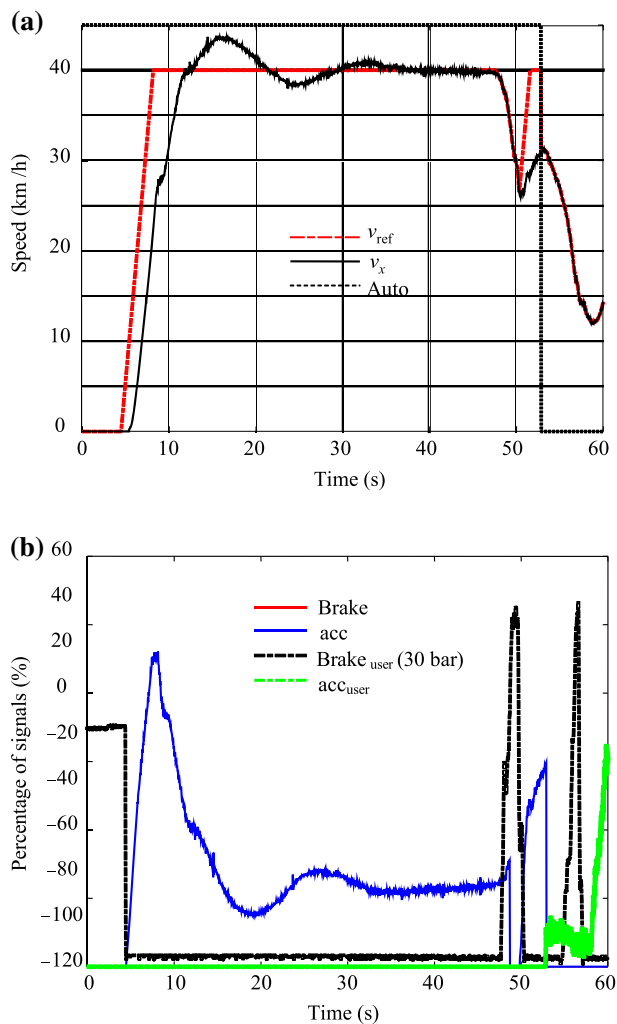


**Fig. 10** Estimated angle of the wheels in the test and reference angle calculated by the path controller. Auto means the moments where autonomous system is activated

of the steering motor based on the vehicle geometry model described in Ref. [26]. The graph shows that the PID controller for the steering servo works correctly in the position control mode. One of the problems found with the control layer was that the motor is configured to operate at the low angular speeds typically requested by the path controller and used by the driver during typical driving situations. However, the angular rates required by the controller are higher by the noise in the reference signal, indicating that the noise filtering by the path control layer needs to be improved. Another problem is that the zero position of the motor is not known precisely, and the dynamics of the steering system is not taken into account when the angle of the wheels is estimated.

## 5.1 Speed controller

Figure 11a shows the longitudinal speed of the vehicle. The reference speed is 40 km/h, which is generated using a constant acceleration rate to achieve a smooth curve. The reference signal is only generated when the driver's brake pedal is released (the black line in Fig. 11b). Around 6 s, there is a delay between the reference signal and the longitudinal speed. This problem may be due to various factors: the speed sensor, which only starts to generate values after a speed of approximately 2 km/h. The model did not



**Fig. 11** Results of the longitudinal speed control test. **a** Longitudinal speed of the car. **b** Acceleration and brake signals under driver control ( $acc_{user}$  and  $Brake_{user}$ ) and automated control ( $acc$  and  $Brake$ )

consider the longitudinal dynamics of the vehicle, which means that the linear speed is measured using the angular velocity and radius of the wheel. The response time of the car engine, which was also not considered. Finally, a dead zone in the accelerator pedal signal. Note that the problem is not about closed loop period, which is in 50 Hz. with a low jitter over all the experiments.

The safety strategy implemented can be observed at around 50 s, when the brake pedal is pressed by the driver and the longitudinal speed control is switched off (the  $acc$  signal drops to zero) until the brake pedal is released. At this point, a new reference curve is generated for the vehicle to return to the planned reference speed. In future studies, feedforward should be used so that the longitudinal speed control can compensate for the force of gravity generated by the pitch angle of the vehicle, as the desired acceleration could not be achieved with the same controller parameters on uphill and downhill slopes.

## 5.2 Cooperative path control

Figure 12a shows the behavior of the lateral error at the start of the route. After 6 s, the vehicle starts to move, and the lateral error converges to zero. It then oscillates around zero as detection of the road line is subject to errors, mainly because of radial distortion introduced by the camera. Between 15 and 20 s, the driver applies torque to the steering wheel ( $\tau_{user}$ ) (Fig. 12c). It is in the opposite direction to the lateral error, so the steering motor increases its force to eliminate the torque applied by the driver as what the steering motor requires is the steering motor controller (path controller) requires is different from the angle the driver wants. After that, a torque is applied in the same direction as the lateral error, so cooperative control is activated. The lateral error is then modified (blue line in Fig. 12a) by changing  $b_k^{ref}$  so the required yaw speed of the vehicle increases, producing a change in the steering angle. Note in Fig. 12b that the orientation angle is not noticeably affected when cooperative control is activated. Finally, Fig. 12d shows the steering control system response; after cooperative control has been activated, there is an evident reduction in the amount of noise in the reference signal.

## 5.3 Cooperative path planning

Figure 13a shows the lateral error in the cooperative path planning mode when the driver wants to change the lateral distance to the road line. In the scenario shown, the driver wants to go in a direction not permitted by the cooperative control, but the path planning layer has determined that it is safe to go in this direction (the button is pressed on the 3D mouse).

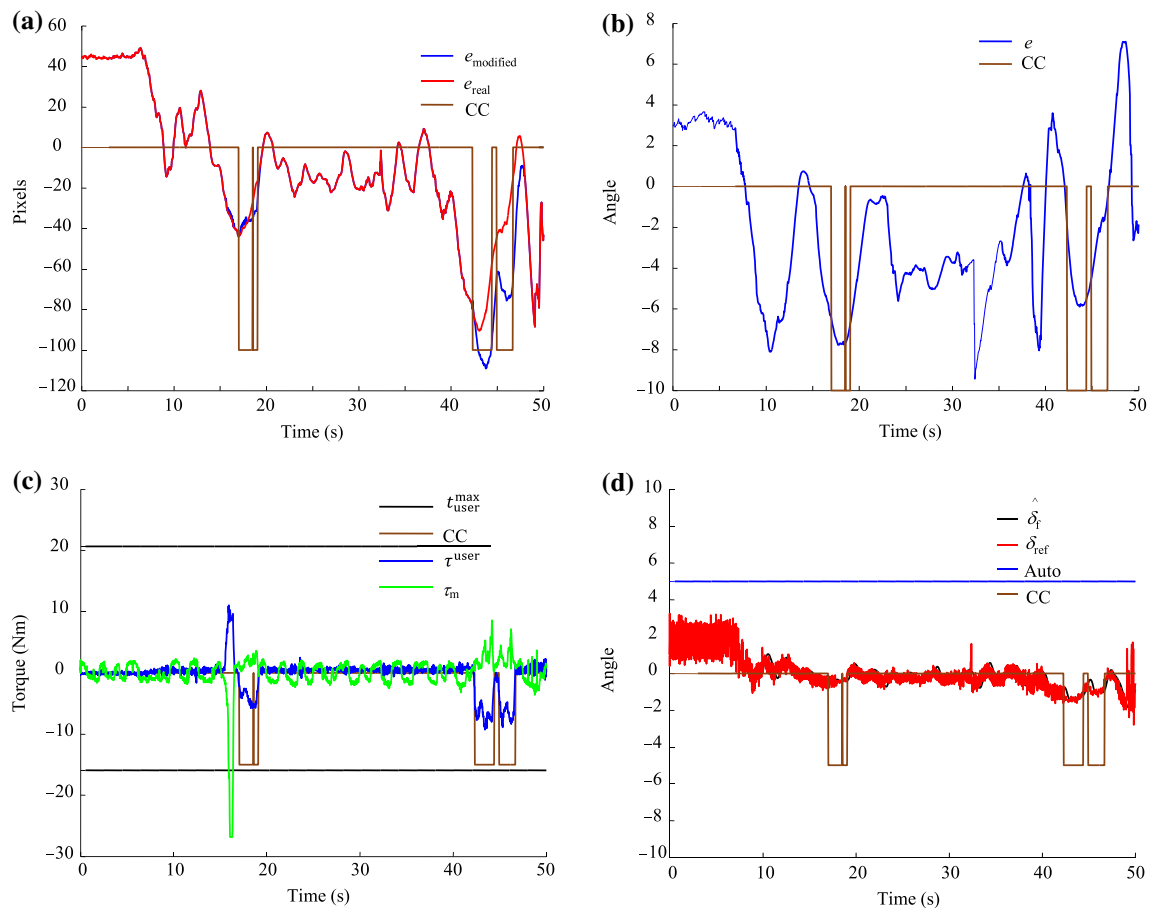
In this strategy, as shown in Fig. 12c, a low steering motor torque could not be achieved during the cooperative stage. There are two main reasons for this. The first is that the path chosen by the driver means that the reference signal is not being changed correctly. Secondly, with the control strategy implemented, whatever way the lateral error is being modified the correction is always in the opposite direction to the driver's torque signal.

Figure 13a shows how the lateral error changed during the test, and Fig. 13b shows that the orientation error is poorly controlled because when the lateral error changes, the desired orientation angle should change too.

## 6 Conclusion

Automated, intelligent vehicle control systems have gained importance in recent years, as they have the potential to increase road safety by removing human involvement in





**Fig. 12** Cooperative path control test. CC means that driver has been detected. **a** Lateral error in cooperative path control. **b** Orientation error in cooperative path control. **c** Steering torque in cooperative path control.  $\tau_m$  is the steer motor torque. **d** Estimated angle in cooperative path control

driving. However, intelligent vehicles that can ensure crash-free driving at all times are not yet a reality. One field of interest to researchers investigating intelligent vehicle control systems is the interaction between a human driver and an intelligent system that can handle many driving environments and conditions. This paper has described a hardware and software architecture used to embed solutions for intelligent systems and has discussed preliminary tests of a new cooperative planning and cooperative control strategies for automated handling of the interaction between a driver and an autonomous system in a low-level implementation project.

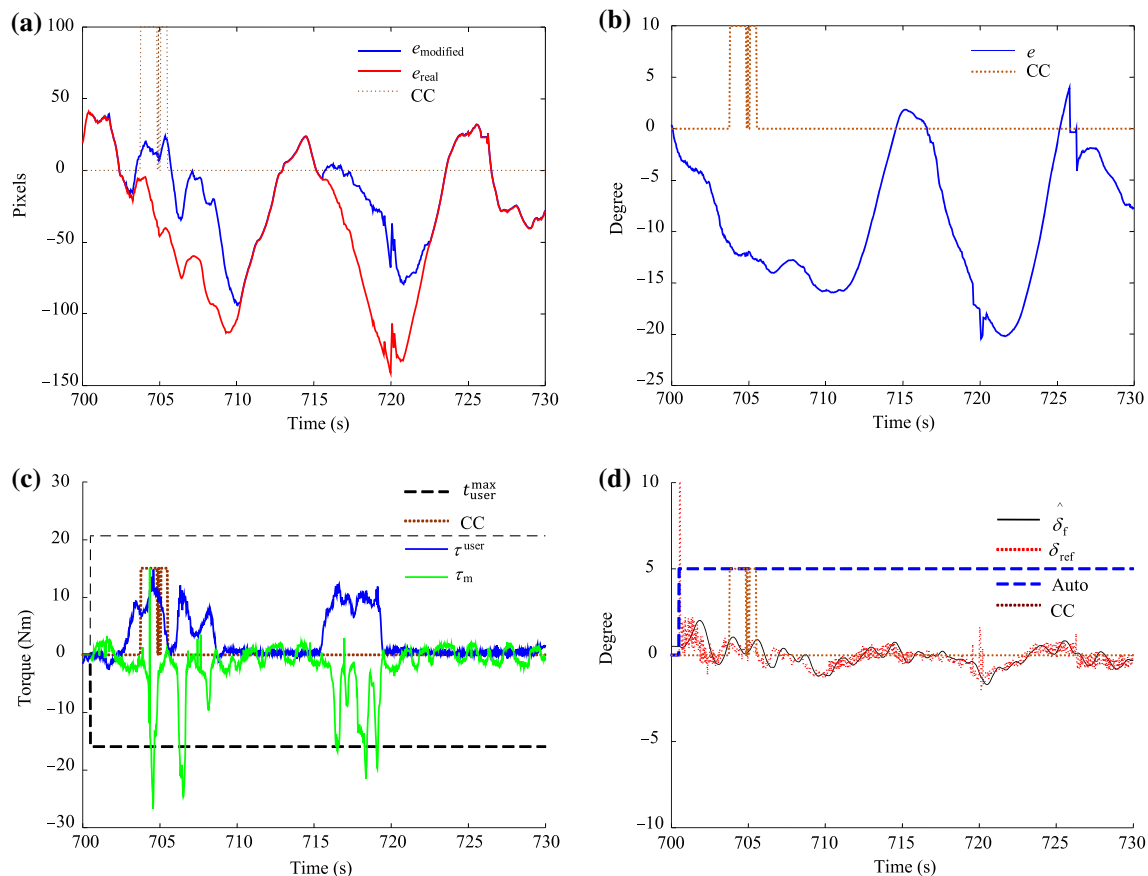
As part of this study, a commercial vehicle was successfully converted to autonomous operation (VILMA01, First Intelligent Vehicle from the Autonomous Mobility Laboratory). The methodology used was based on the concepts of mechatronics. During the conversion, the individual elements were tested separately and together. The hardware and software architecture used allowed the reliability of an embedded automotive system (MicroAutobox dSpace) to be combined with the flexibility and

computational capacity of a distributed PC system based on Linux and ROS. The architecture also allowed code to be developed quickly, reused, and easily debugged.

Although the perception part needs to be improved to increase reliability, a visual servoing control was implemented so that road lines could be followed using only a low-cost camera. Experiments in the autonomous mode were carried out in different environmental conditions, including day and night, at speeds of up to 60 km/h.

The results show that the cooperative control algorithm lets the user feel that he was driving the vehicle because it added a smaller torque to the torque applied by the driver, while the visual servoing system ensured that the road line was followed.

The cooperative planning allowed the user to select the desired path after the autonomous system had emulated the path and analyzed the risk, making the vehicle safer. However, for the algorithm to allow the driver to enjoy a genuine feeling of cooperative driving, i.e., for the strategy to be successful regarding cooperative torques, the system must predict the path the user is planning to take. As a



**Fig. 13** Cooperative path planning. **a** Lateral error in cooperative path planning. **b** Orientation error in cooperative path planning. **c** Steering torque in cooperative path planning. **d** Estimated angle in cooperative path planning

follow-up to this research, we are working to overcome the problems identified on the perception area and implement a cooperative control and planning approach using a predictive control strategy [28] and a local path planning algorithm based on a curvilinear [29].

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

1. Bahadorimonfared A, Soori H, Mehrabi Y, Delpisheh A, Esmaili A, Salehi M, Bakhtiyari M (2013) Trends of fatal road traffic injuries in Iran (2004–2011). *PLoS ONE* 8:e65198
2. Organization WH (2015) Global status report on road safety 2015. WHO, Geneva
3. Park J-U, Bae B-H, Lee J-W, Kim J-H (2010) Design of failsafe architecture for unmanned ground vehicle. In: 2010 international conference on control automation and systems (ICCAS)
4. NHTSA (2013) US department of transportation releases policy on automated vehicle development
5. Cullinane B, Nemec P, Clement MC, Mariet RCE, Jonsson LIM (2014) Engaging and disengaging for autonomous driving. Google Patents
6. Kaplan S, Guvensan MA, Yavuz AG, Karalurt Y (2015) Driver behavior analysis for safe driving: a survey. *IEEE Trans Intell Transp Syst* 16:3017–3032
7. Jain A, Koppula HS, Raghavan B, Soh S, Saxena A (2015) Car that knows before you do: anticipating maneuvers via learning temporal driving models. In: 2015 IEEE international conference on computer vision (ICCV), 2015
8. Malik H, Larue GS, Rakotonirainy A, Maire F (2015) Fuzzy logic to evaluate driving maneuvers: an integrated approach to improve training. *IEEE Trans Intell Transp Syst* 16:1728–1735
9. Liu P, Kurt A, Özgüner U (2014) Trajectory prediction of a lane changing vehicle based on driver behavior estimation and classification. In: 17th international IEEE conference on intelligent transportation systems (ITSC), 2014
10. Amsalu SB, Homaifar A, Afghah F, Ramyar S, Kurt A (2015) Driver behavior modeling near intersections using support vector machines based on statistical feature extraction. In: 2015 IEEE intelligent vehicles symposium (IV), 2015
11. Merat N, Jamson AH, Lai FCH, Daly M, Carsten OMJ (2014) Transition to manual: driver behaviour when resuming control from a highly automated vehicle. *Transportation Research Part F: Traffic Psychology and Behaviour* 27(Part B):274–282

12. Siegwart R, Nourbakhsh IR, Scaramuzza D (2011) Introduction to autonomous mobile robots, 2nd edn. MIT Press, Cambridge
13. Chen Y-L, Sundareswaran V, Anderson C, Broggi A, Grisleri P, Porta PP, Zani P, Beck J (2008) TerraMax: team Oshkosh urban robot. *J Field Robot* 25:841–860
14. Thrun S, Montemerlo M, Dahlkamp H, Stavens D, Aron A, Diebel J, Fong P, Gale J, Halpenny M, Hoffmann G et al (2006) Stanley: the robot that won the DARPA grand challenge. *J Field Robot* 23:661–692
15. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) ROS: an open-source robot operating system. In: ICRA workshop on open source software, 2009
16. Bruyninckx H (2001) Open robot control software: the OROCOS project. In: Proceedings of the IEEE international conference on robotics and automation, 2001 ICRA
17. Garcia O, Ferreira JV, de Miranda Neto A (2013) Dynamic model of a commercial vehicle for steering control and state estimation. In: XI Simposio Brasileiro de Automação Inteligente (SBAI)
18. Ferreira T, Garcia O, Vaqueiro J (2015) Software architecture for an autonomous car simulation using ROS. In: XII Simpósio brasileiro de automação Inteligente
19. Koenig N, Howard A (2004) Design and use paradigms for gazebo, an open-source multi-robot simulator. In: Proceedings of the 2004 IEEE/RSJ international conference on intelligent robots and systems, 2004 (IROS 2004)
20. Freese M, Singh S, Ozaki F, Matsuhira N (2010) Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In: Simulation, modeling, and programming for autonomous robots, Springer, 2010, pp 51–62
21. Echeverria G, Lassabe N, Degroote A, Lemaignan S (2011) Modular open robots simulation engine: morse. In: 2011 IEEE international conference on robotics and automation (ICRA)
22. Fernandes LC, Souza JR, Pessin G, Shinzato PY, Sales D, Mendes C, Prado M, Klaser R, Magalhães AC, Hata A et al (2014) CaRINA intelligent robotic car: architectural design and applications. *J Syst Architect* 60:372–392
23. Grompone von Gioi R, Jakubowicz J, Morel J-M, Randall G (2012) LSD: a line segment detector. *Image Processing On Line* 2:35–55
24. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 24:381–395
25. Rusu RB, Cousins S (2011) 3D is here: point cloud library (PCL). In: IEEE international conference on robotics and automation (ICRA), Shanghai
26. Snider JM (2009) Automatic steering methods for autonomous automobile path tracking. Robotics Institute, Pittsburgh
27. Cherubini A, Chaumette F, Oriolo G (2011) Visual servoing for path reaching with nonholonomic robots. *Robotica* 29:1037–1048
28. Garcia O, Ferreira JV, Neto AM (2014) Design and simulation for path tracking control of a commercial vehicle using MPC. In: 2014 joint conference on robotics: SBR-LARS robotics symposium and robocontrol (SBR LARS robocontrol)
29. Lemos R, Garcia O, Ferreira JV (2016) Local and global path generation for autonomous vehicles using splines. *Ingeniería* 21:188–200