sensors

MDPI

Article

Optimal H_∞ Control for Lateral Dynamics of Autonomous Vehicles

Gianfranco Gagliardi *, Marco Lupia, Gianni Cario and Alessandro Casavola

Dipartimento di Ingegneria Elettronica, Informatica e Sistemistica (DIMES), Universitá della Calabria, 87036 Rende, CS, Italy; mlupia@dimes.unical.it (M.L.); gcario@dimes.unical.it (G.C.); a.casavola@dimes.unical.it (A.C.)

* Correspondence: g.gagliardi@dimes.unical.it

Abstract: This paper presents the design and validation of a model-based H_∞ vehicle lateral controller for autonomous vehicles in a simulation environment. The controller was designed so that the position and orientation tracking errors are minimized and so that the vehicle is able to follow a trajectory computed in real-time by exploiting proper video-processing and lane-detection algorithms. From a computational point of view, the controller is obtained by solving a suitable LMI optimization problem and ensures that the closed-loop system is robust with respect to variations in the vehicle's longitudinal speed. In order to show the effectiveness of the proposed control strategy, simulations have been undertaken by taking advantage of a co-simulation environment jointly developed in Matlab/Simulink © and Carsim 8 ©. The simulation activity shows that the proposed control approach allows for good control performance to be achieved.

Keywords: autonomous vehicles; automotive control; H_∞ control; lateral control; linear matrix inequalities; path tracking; steering angle control

Citation: Gagliardi, G.; Lupia, M.; Cario, G.; Casavola, A. Optimal H_∞ Control for Lateral Dynamics of Autonomous Vehicles. *Sensors* **2021**, *21*, 4072. https://doi.org/10.3390/s21124072

Academic Editor: Maria Jesús López Boada

Received: 10 May 2021
Accepted: 11 June 2021
Published: 13 June 2021

1. Introduction

Vehicle safety is a major human challenge. The World Health Organization (WHO) reports more than one million fatalities in traffic accidents and around 20–50 million injuries each year worldwide. The WHO estimates that approximately one-half of the fatalities involve what are referred to as "vulnerable road users" i.e., pedestrians, cyclists, and motorbikes [1]. A general system approach that accounts for the interactions between humans, vehicles, and the environment and provides the necessary countermeasures in preventing crashes is required to tackle this problem.

With respect to this problem, the automotive industry has made relevant strides, and at the same time, governments around the world have increased safety regulations. Passive and active systems have been developed throughout the years to enhance vehicle safety [2]. Moreover, further research efforts have been devoted to investigating the main causes of road accidents and to designing mathematical models in charge of predicting and reducing the damage caused by road accidents on the basis of statistical analysis of "correlate factors" such as driver gender and age, alcohol use, vehicle type, etc. [3,4]. In recent years, research on Advanced Driving Assistant System (ADAS) and intelligent vehicles have attracted a lot of attention due to technological progress in the fields of sensing, communication, and information processing. This interest pertains not only to the intelligent functions that support a driver during driving but also to automated driving. As road accidents occur frequently and result in property damages, injury, and even death, all of which impose a high cost to societies, the main aim in developing new and more sophisticated functions for ADAS systems is to improve road safety, to mitigate traffic issues, and to improve driving comfort [5,6].

ADAS can be viewed as real-time systems able to react quickly to multiple inputs and to prioritize incoming information to prevent accidents. ADAS can be categorized

and divided into six levels based on the amount of automation. This scale is provided by the Society of Automotive Engineers (SAE) and has been designed to clarify and simplify the SAE standard J3016 of *Levels of Driving Automation*: the standard defines six levels of driving automation, from SAE Level 0 to SAE Level 5 [7,8]. In Level 0 (*No Driving Automation*), ADAS cannot control the car and can only provide information for the drivers to interpret the current situation on their own: parking sensors, traffic signs recognition, lane departure warning systems, etc. belong to this level. Levels 1 and 2 are very similar in that they both let the driver make most of the decisions. The difference is that, at Level 1 (*Driver Assistance*), the ADAS can take over the control of one functionality (e.g., adaptive cruise control, emergency brake assist, etc.). On the other hand, at Level 2 (*Partial Driving Automation*), the ADAS can take over the control of multiple functionalities to aid the driver: autonomous obstacle avoidance, autonomous parking, etc. At Level 3 (*Conditional Driving Automation*), vehicles have "environmental detection" capabilities and can make informed decisions for themselves, such as accelerating past a slow-moving vehicle. However, they still require human override. The driver must remain alert and ready to take control if the system is unable to execute the task. The key difference between Level 3 and Level 4 is that, at Level 4 (*High Driving Automation*), drivers can intervene if things go wrong or there is a system failure. In this sense, these cars do not require human interaction in most circumstances. However, a human still has the option to manually override the system. At level 5 (*Full Driving Automation*), vehicles do not require human attention: the "dynamic driving task" is eliminated.

In this respect, lane-detection (LD) and lane-keeping (LK) systems are important challenges. An autonomous driving car can be viewed as a vehicle that is able to drive in different driving scenarios by replacing human actions at different levels. Then, autonomous cars must be equipped with all of the necessary functionalities to solve three main tasks [9]:

- **Environmental perception**. Due to the fact that the environment in which cars are used must be considered partially unknown, a vision system (cameras and sensors such as radar, lidar, etc.) must be used to detect road boundaries, various objects (e.g., obstacles and pedestrians), and other vehicles. In this way, it is possible to provide a dynamic map of the environment around the autonomous vehicle.
- **Trajectory generation**. This task concerns the generation of a reference trajectory (or reference path) in the navigable environment.
- **Vehicle control**. This task consists of designing control algorithms for longitudinal and lateral control, which use available actuators (accelerator pedal, brakes, steering wheel, etc.) to track the reference trajectory.

In order to achieve these goals, the autonomous vehicle has to use a number of well-placed sensors that detect and continuously observe the location and movement of other vehicles, people, traffic lights, etc.

This paper focuses on the second and third items of the aforementioned steps and relies on the implementation of a lateral controller that automatically acts on the steering wheel to track a reference trajectory. It is important to note that the design of the vehicle's longitudinal controller is beyond the scope of this work. Observe in fact that the later and longitudinal dynamics are practically decoupled so that the longitudinal controller can be designed separately.

Due to the high non-linearity of the system, the uncertainty in the model parameters, and the presence of disturbances, robustness of the controller is an important goal. There is rich literature that reports relevant research efforts to provide suitable lateral guidance in autonomous vehicles. Beside standard (PI-PID) control approaches [10–14], approaches based on Model Predictive Control (MPC) have attracted considerable attention in trajectory following applications. In [15], an MPC algorithm that is in charge of generating smooth and collision-free trajectories for a given predicted velocity profile was presented. In [16], an MPC controller was designed so that the front steering angle is computed in order to follow the trajectory on slippery roads at the highest possible entry speed. The MPC

controller was designed by assuming that the trajectory is known over a finite horizon. A model predictive controller that uses a speed-dependent adaptation of the prediction model and cost function weights to ensure a stable and precise path tracking performance was presented in [17]. An important issue was related to the fact that the performance of a path tracking controller can be affected by system malfunctions due to internal factors (e.g., sensors and electronic device faults) or to the road (e.g., marking quality, etc.), light, and weather conditions. In this respect, in [18,19], the LD system performance and probability of fault were investigated with special focus on the effects of the physical infrastructure related to road characteristics and conditions. In [20], a robust fault-tolerant path tracking control algorithm based on adaptive MPC was proposed. The main drawback in the use of MPC control strategies was in the computation time that, for high-speed driving, becomes too large for real-time operations. Other control strategies have also been used for lateral control purposes: a H_∞ robust lateral controller for differential GPS-based autonomous vehicle was adopted in [21] whilst a H_∞ path-tracking control problem of network-based autonomous vehicles was presented in [22]. Specifically, the controller has been shown to be robust with respect to parameter uncertainties and external disturbances and to allow for good performance in the presence of delays and packet dropout. In [23] and in [9], Linear Quadratic Regulator (LQR) and sliding mode control approaches were presented, respectively.

In this paper, the main goal is to design a control system able to perform the trajectory following task for an autonomous vehicle. The control architecture accounted for is modular and exploits information from onboard sensors and vision systems. The designed control architecture is in charge of the following:

- Determining a reference trajectory in real-time on the basis of the output of a lane-detection procedure that elaborates the environment information acquired by a camera and
- Allowing for path following by controlling the steering angle.

A particularly complex problem arises when the road strips are incomplete or totally missing and, hence, the camera is unable to suitably define the boundaries of the driving lane (e.g., country roads lacking suitable road signage, vehicle/queue management when approaching toll gates or motorway connections, etc.). Furthermore, another important issue is guaranteeing the performance of the overall control system at various vehicle speeds.

In this respect, this paper proposes a control architecture that is capable of verifying if the lane-detection procedure is able to provide a lane estimation and, at the same time, to provide a control action robust with respect to disturbances and variation in the vehicle speed. The lane-detection procedure is designed according to [24], and the control action exploits an optimal H_∞ tracking controller that is computed by solving a convex LMI optimization problem [25].

The paper is organized as follows. Section 2 describes the steering control architecture and all of its subsystems: the lane-detection and the trajectory-generation modules and the model of the vehicle lateral dynamics. In Section 3, the robust H_∞ lateral controller is presented, whilst in Section 4, the simulation results regarding two driving scenarios are reported. Some conclusions end the paper.

2. Steering Control for Autonomous Vehicles

Steering control for autonomous vehicles is a control architecture that combines various active safety systems and allows for a vehicle to follow a path computed in real-time on the basis of the road conditions (Figure 1):

- **Camera module**, which records the current view of the road. The camera is directed towards the front of the vehicle;
- **Lane-detection module**, which is in charge of detecting lane strips in an image. It makes use of the detected strips in order to compute an estimation of the car position within the lane;

- **Trajectory-generation module**, which is in charge of computing the reference trajectory (ψ_{des}) on the basis of the lane estimation provided by the lane-detection module;
- **Controller**, which provides the necessary control actions to guarantee that the vehicle follows the reference trajectory; and
- **Vehicle lateral dynamics**, which is the module that implements the mathematical model of lateral vehicle motion.

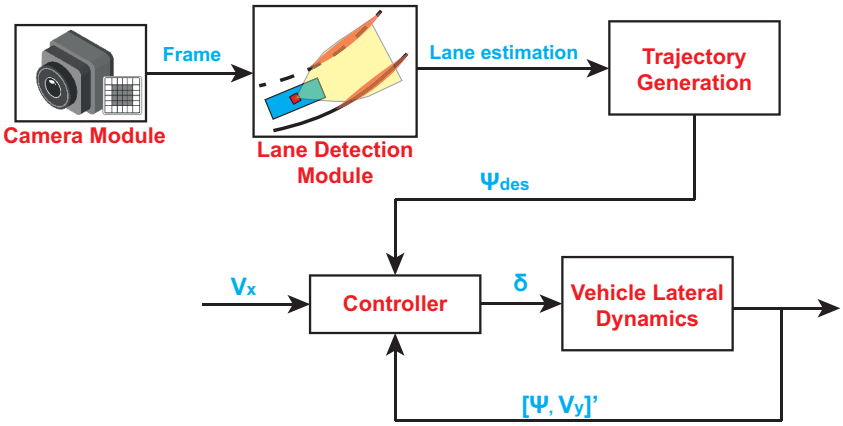


Figure 1. Steering control schematic.

By taking into account the steering control schematic reported in Figure 1, the design procedure was accomplished by considering two main tasks:

1. Designing and testing a video-processing algorithm in charge of providing an estimation of the car position within the lane in a real application context and
2. Designing and validating a steering controller that allows the car to follow a reference trajectory in a co-simulation environment.

In what follows, all of the devices and modules that pertain to the steering control architecture are briefly described.

2.1. Camera Module

An autonomous vehicle can be equipped with various types of sensors (e.g., radar, lidar, camera, etc.) located anywhere outside or inside the vehicle [1].

The monocular camera is a standard vision sensor used in automated driving applications. This type of sensor can be useful in object- and lane-boundary detection and in object-tracking applications. As highlighted in Figure 2a, the camera coordinate system is described by a standard Cartesian representation with the origin located at the cameras' optical center. On the other side, the yaw, pitch, and roll angles are referenced according to the ISO convention (refers to Figure 2b). The camera used in our application is a CMOS FireFly MV camera (Figure 3a) [26,27]. Figure 3b shows the installation of the camera on the car windshield. Note that, in the setup of the camera system, the parameters reported in Table 1 were accounted for. The camera, with a specified rate, acquired the frames that were outputted to the lane-detection module.

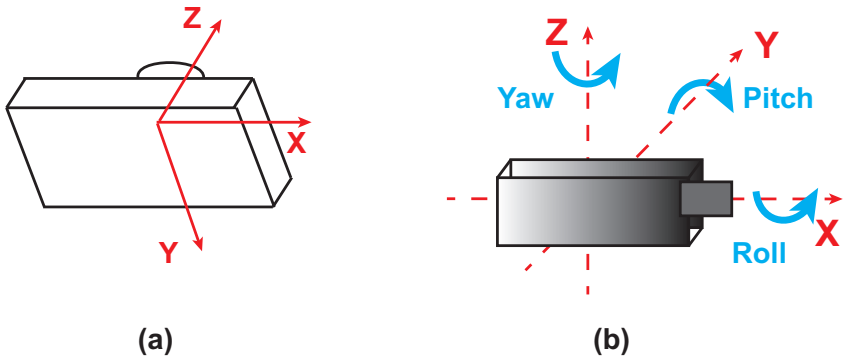


Figure 2. Camera module: reference coordinates (**a**) and ISO convention (**b**).

Table 1. Camera parameters.

Parameter	Description	Value
X(m), Y(m)	*X*-axis and *Y*-axis positions of the camera in the vehicle coordinate system.	[0.83, 3.8]
Height(m)	Height of the camera above the ground.	1.2 m
Focal length (X,Y)	Horizontal and vertical point at which the camera is in focus.	[0.83, 6.2]
Image Height and Width	Horizontal and vertical point camera camera resolution (in pixel).	752 × 480
Principal Point X and Y	Horizontal and vertical image center (in pixel).	[376, 240]
Update Interval	Camera updating frequency.	61 FPS

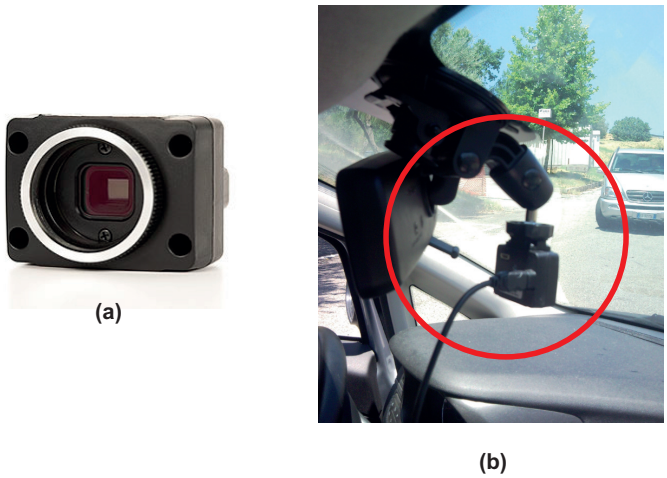


Figure 3. CMOS FireFly MV camera (**a**) and camera vehicle mounting (**b**).

2.2. Lane-Detection and Trajectory-Generation Modules

The lane-detection module is in charge of estimating the strips and road lanes. The typical steps for lane strip identification are reported in Figure 4: the camera acquires an image frame, and a preprocessing module in charge of improving the quality of the frame starts (e.g., improves the image contrast, reduces the image noise, etc.). Then, the processed image is further elaborated to extract the contours and to identify the pixels belonging to the lane strips. A Kalman filter is used to predict and track the lane strips over consecutive frames.

Finally, a fitting process procedure is accomplished to aggregate different groups of pixels potentially belonging to the same strip and to provide the strip coordinates and an estimate of the curvature. The strip-fitting problem could be made more efficient if one knows that certain forms are present in the image and a good mathematical model (e.g.,

linear, linear parabolic, polynomial, clothoid, and spline models) is available to describe them. Here, lane-strip determination is based on a linear–parabolic (LP) fitting [28].

The basic idea is to separate the frame into two fields, near field and far field (Figure 5a), using a fixed threshold x_m and to perform a linear fitting in the first one and a quadratic fitting in the second one. The curve that represents the strips is given by the following:

$$f(x) = \begin{cases} a + b(x - x_m) & x > x_m \\ a + b(x - x_m)x + c(x - x_m)^2 & x \leq x_m \end{cases} \quad (1)$$

In order to accomplish such an LP fitting, the first step is to identify the two areas of interest (Lane Boundary Region of Interest (LBROI)) within the current frame (Figure 5b).

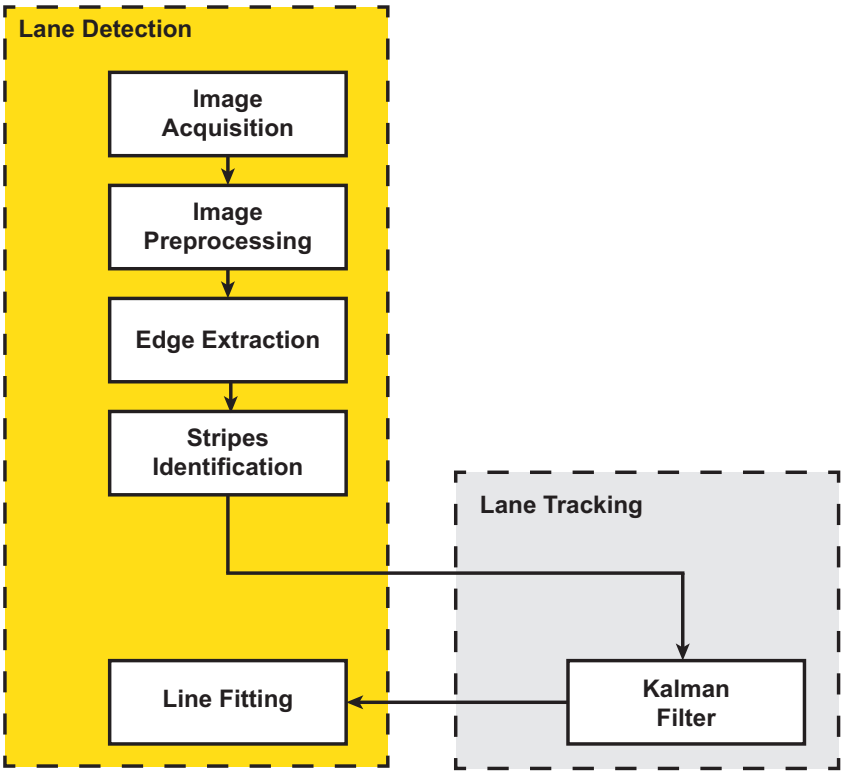
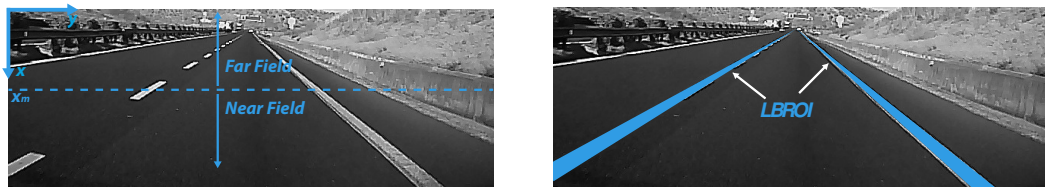


Figure 4. Lane-detection and tracking algorithm.



(**a**) (**b**)

Figure 5. Linear–parabolic fitting: near and far field image separation (**a**) and LBROI (**b**).

The parameters a, b, and c of Equation (1) are determined by a linear fitting minimizing the weighted quadratic error E:

$$E = \sum_{i=1}^{m} M_{n_i}[y_{n_i} - f(x_{n_i})]^2 + \sum_{j=1}^{n} M_{f_j}\left[y_{f_j} - f(x_{f_j})\right]^2 \quad (2)$$

where (x_{n_i}, y_{n_i}), with $i = 1, \ldots, m$, represents the mth pixel coordinates and where M_{n_i} represents the corresponding magnitudes (only for the pixels $\neq 0$). Similarly, (x_{f_j}, y_{f_j}) and

M_{f_j}, with $j = 1, \ldots, n$, represent the same quantities for the n pixels in the far field with magnitudes $\neq 0$.

The E error is minimized analytically by finding a solution to the following linear system of equations:

$$A^T W A C = A^T W B \tag{3}$$

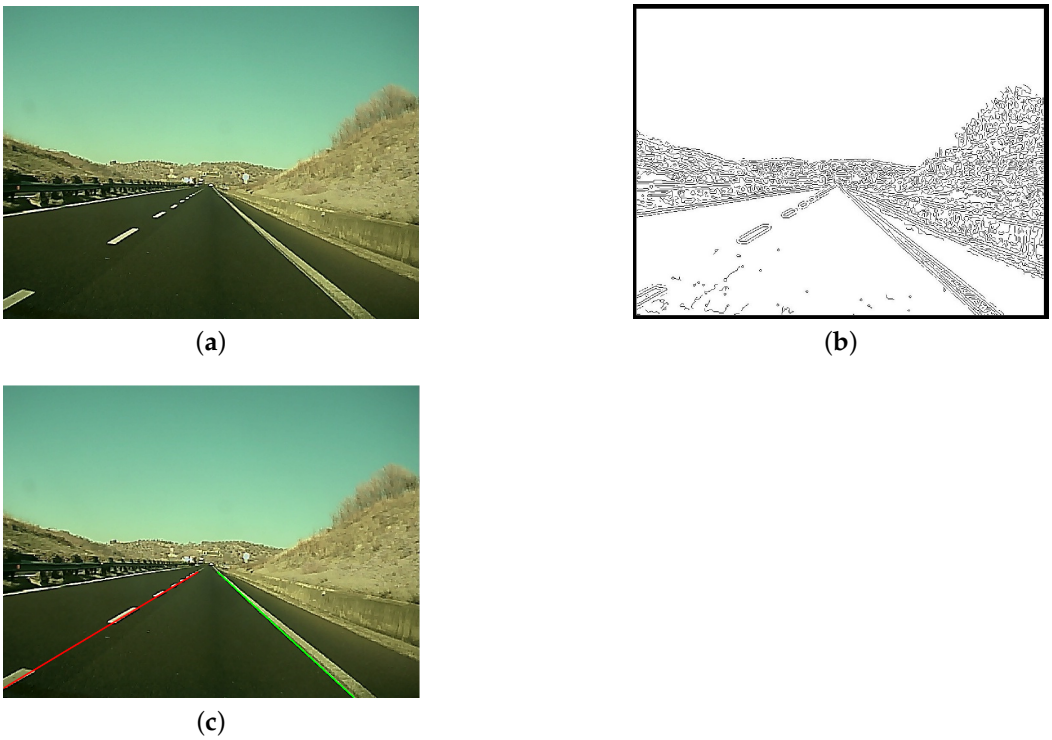
where

$$A = \begin{bmatrix} 1 & x_{n_1} - x_m & 0 \\ \vdots & \vdots & \vdots \\ 1 & x_{n_m} - x_m & 0 \\ 1 & x_{f_1} - x_m & (x_{f_1} - x_m)^2 \\ \vdots & \vdots & \vdots \\ 1 & x_{f_n} - x_m & (x_{f_n} - x_m)^2 \end{bmatrix}, \; W = \begin{bmatrix} M_{n_1} & & & & & \\ & \ddots & & & & \\ & & M_{n_m} & & & \\ & & & M_{f_1} & & \\ & & & & \ddots & \\ & & & & & M_{f_n} \end{bmatrix} \tag{4}$$

$$C = \left[a, b, c^T\right], \; B = \left[y_{n_1}, \ldots, y_{n_m}, y_{f_1}, \ldots, y_{f_n}\right] \tag{5}$$

The procedure is applied to both strips of the lane.

In Figure 6, the various phases of the lane-detection process are shown on a real road frame.



(**a**) (**b**) (**c**)

Figure 6. Lane detection: image acquisition and preprocessing (**a**), edge extraction (**b**), and line identification (**c**).

Details about the implementation of the lane-detection algorithm can be found in [24].

At the end of the lane-detection procedure, the trajectory-generation computation task starts. The trajectory-generation phase consists of finding the trajectory and of computing its curvature on the basis of the information on the lane strips coming from the previous step. In general, the reference trajectory consists of the curvature of the lane centerline (refers to Figure 7) and can be computed as the average value between the left strip of the lane and the right one. Furthermore, since the controller needs to receive the necessary

information to generate the control action, the computed curvature needs to be recast in terms of the desired yaw angle (ψ_{des}). In this respect, the following are highlighted [29]:

- The reference angular velocity of the vehicle can be defined as $\dot{\psi}_{des} = V/R$, where V is the vehicle speed at the center of gravity and R is the radius of curvature.
- The radius of curvature R can be computed as the inverse of the absolute value of the curvature k at a point: $R = 1/|k|$.

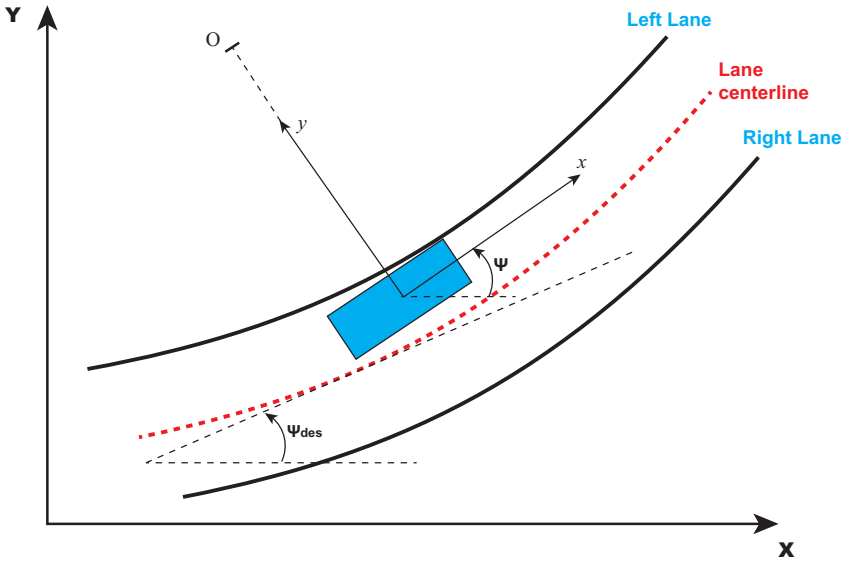


Figure 7. Road representation used in lateral control, highlighting the lane centerline, and the left and right lanes.

With reference to Equation (1) and Figure 8, in order to compute the curvature k, it is necessary to consider the *osculating circle*, that is, the circle with radius R centered at the curvature center. This circle allows one to locally approximate the curve up to the second order. Then, the curvature k can be expressed in terms of the first- and second-order derivatives of the curve f as [30]:

$$k = \frac{|\ddot{f}|}{[1 + \dot{f}^2]^{3/2}} \tag{6}$$

Then, the desired yaw angle can be computed as follows:

$$\psi_{des} = \int_0^t V|k|dt \tag{7}$$

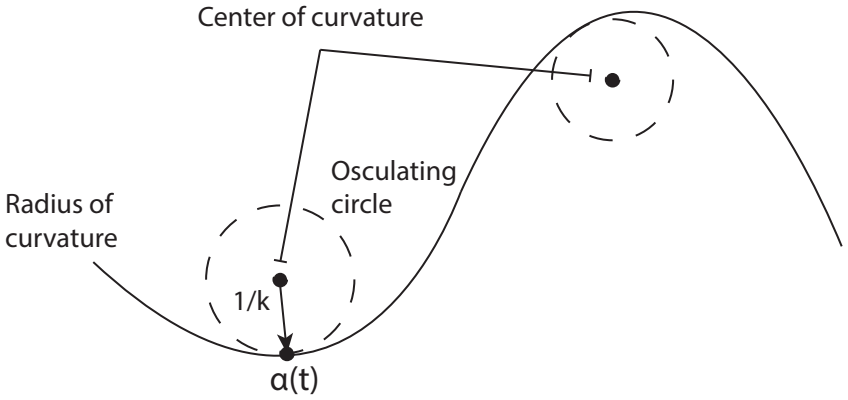


Figure 8. Osculating circle and radius of curvature.

It is important to note that, due to the fact that it is preferable that the vehicle always stays at the lane centerline, this task plays a key role in the overall steering control process. Since the lane detection algorithm (Algorithm 1) can fail and the detection algorithm cannot perform correct lane-strip identification, four cases must be accounted for in the reference trajectory computation:

1. Both the left and right strips are detected.
2. Only the left strip is detected.
3. Only the right strip is detected.
4. The left and right strips are not found: in this case, a safe driving condition (*limp home* driving mode) must be activated in order to avoid dangerous events from taking place.

Then, the following logic is included in the computation of the curvature k for the trajectory-generation procedure where r, k_l, and k_r are a constant indicating the half-lane dimension (r = 1.8 [m]) and the curvature of the left and right strips, respectively.

Algorithm 1 Curvature computation

procedure CURVATURE(k_l, k_r)
  if k_l & k_r **then** ▷ Both the left and right strips are detected.
    $k = (k_l + k_r)/2$
  else if k_l & $!k_r$ **then** ▷ Only the left strip is detected.
    $k = k_l/(1 - k_l * r)$
  else if k_r & $!k_l$ **then** ▷ Only the right strip is detected.
    $k = k_r/(1 + k_r * r)$
  else ▷ The left and right strips are not found.
    disable autonomous guidance and activate "limp home" strategy
  end if
end procedure

The lane-detection and trajectory-generation algorithms were implemented via the Image Processing and Computer Vision Toolbox available in Matlab/Simulink. In order to build up a prototypal setup, the Matlab/Simulink code of the video-processing algorithm was coded in the *C language*, suitable for the Board EVM TMS320DM642. This evaluation module (Figure 9) is a low-cost standalone development platform that enables users to evaluate and develop applications for the TI C64xx Digital Signal Processor (DSP) family [31].

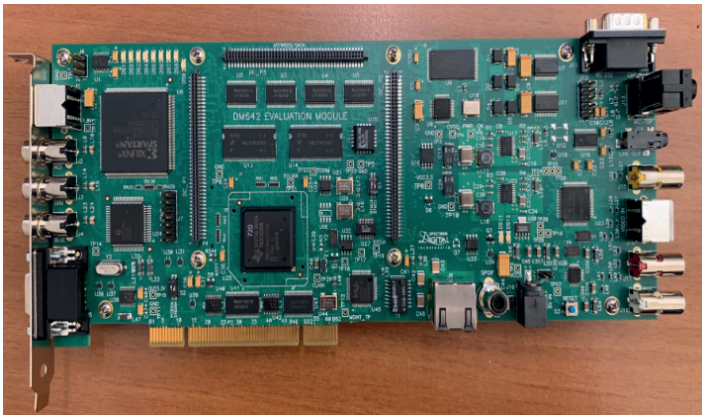


Figure 9. Board EVM TMS320DM642.

2.3. Vehicle Lateral Dynamics

In the design of a model-based lateral controller for autonomous cars, knowledge of the car model plays a key role in the design of the path planning and path-tracking modules. Under certain assumptions (e.g., for low longitudinal speed of the vehicle), a kinematic model can be considered for vehicle lateral motion. When this type of modeling approach is adopted, it is possible to provide a mathematical description of the vehicle motion without considering the forces that influence the motion. The motion equations are simply based on geometric relationships.

In order to compute the motion equations for the kinematic model, the bicycle model reported in Figure 10 is usually used [32]. In this figure, two coordinate systems are highlighted: the *world coordinate* system (X, Y) and the body-fixed coordinate system (x, y). The major assumption in the correct use of the kinematic model is that the velocities at the points **A** and **B** are always oriented in the direction of the wheels' orientation, i.e., this corresponds to assuming that the wheel slip angles are zero, a condition that almost holds true at low speeds. The total lateral force needed to ride a circular road of radius **R** is given by the following:

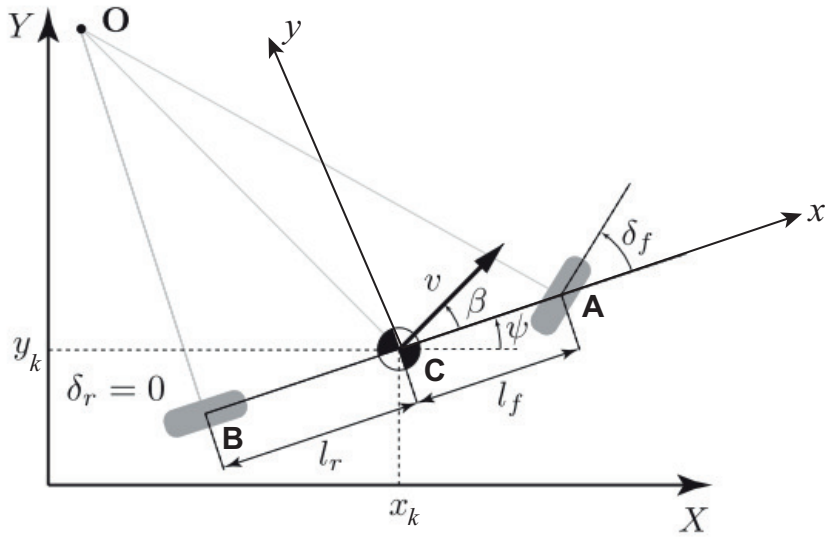


Figure 10. Vehicle bicycle model.

$$F_y = \frac{mv^2}{R} \tag{8}$$

where m and v are the vehicle mass and the vehicle speed, respectively. The motion equations for the kinematic model are given by the following [29]:

$$\begin{cases} \dot{x}_k &= v cos(\psi + \beta) \\ \dot{y}_k &= v sin(\psi + \beta) \\ \dot{\phi}_z &= \frac{v cos(\beta)}{l_f + l_r}(tan(\delta_f) - tan(\delta_r)) \end{cases} \tag{9}$$

where

- x_k and y_k are the trajectories;
- ϕ_z is the yaw angle;
- ψ is the vehicle orientation;
- β is the slip angle;
- l_f and l_r are the distances of points **A** and **B** from the center of gravity (**C**), respectively; and

- δ_f and δ_r are the front and rear wheels steering angles, respectively. Note that the steering angle of rear wheel is assumed to be $\delta_r = 0$.

It is important to note that the kinematic model (9) is not accurate at higher longitudinal vehicle speeds due to the fact that the previous assumption is no longer true. In this case, a dynamic model of the lateral vehicle model should be accounted for.

The dynamic model can be derived by taking into account the two-degrees-of-freedom bicycle model depicted in Figure 10, where the degrees of freedom are represented by the vehicle lateral position (y) and the vehicle yaw angle (ψ). Then, starting from Newton's second law for motion along the y-axis and by computing the moment balance around the vertical z-axis, the following model can be derived for the vehicle lateral dynamics:

$$\begin{cases} ma_y &= F_{yf} + F_{yr} \\ I_z\ddot{\psi} &= l_f F_{yf} - l_r F_{yr} \end{cases} \tag{10}$$

in which

- a_y is the inertial acceleration of the vehicle at the center of gravity in the y-axis direction;
- F_{yf} and F_{yr} are, respectively, the lateral tire forces of the front and rear wheels; and
- I_z is the inertia of the vehicle around the z-axis.

In Equation (10), two terms contribute to the lateral acceleration (a_y): the acceleration due to the motion along the y-axis ($\ddot{y}$) and the centripetal acceleration ($v_x\dot{\psi}$). Furthermore, the lateral tire forces are proportional to the slip angle that is defined as the angle between the orientation of the tire and the orientation of the velocity vector of the wheel (see Figure 11).

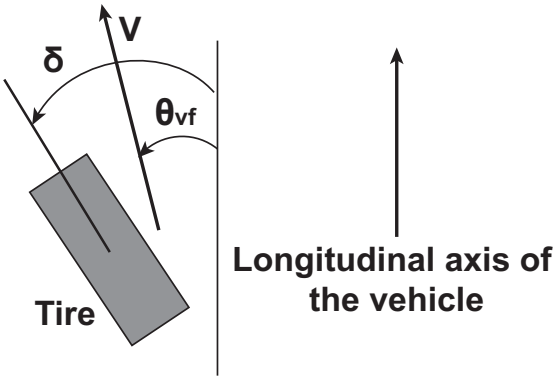


Figure 11. Slip angle.

Then, Equation (10) can be written as follows:

$$\begin{cases} m(\ddot{y} + \dot{\psi}v_x) &= 2C_{\alpha f}(\delta - \theta_{vf}) + 2C_{\alpha f}(-\theta_{vr}) \\ I_z\ddot{\psi} &= l_f 2C_{\alpha f}(\delta - \theta_{vf}) - l_r 2C_{\alpha f}(-\theta_{vr}) \end{cases} \tag{11}$$

Furthermore, by assuming small-angle approximation [29],

$$\theta_{vf} \approx \frac{\dot{y} - l_f\dot{\psi}}{v_x}, \quad \theta_{vr} \approx \frac{\dot{y} - l_r\dot{\psi}}{v_x}$$

the following state-space model can be derived.

$$\frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -2\frac{C_{\alpha f}+C_{\alpha r}}{mv_x} & 0 & -v_x - 2\frac{C_{\alpha f}l_f - C_{\alpha r}l_r}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & -2\frac{C_{\alpha f}l_f + C_{\alpha r}l_r}{I_z v_x} & 0 & -2\frac{C_{\alpha f}l_f^2 - C_{\alpha r}l_r^2}{I_z v_x} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2C_{\alpha f}l_f}{I_z} \end{bmatrix} \delta \quad (12)$$

Since the objective is to design a steering control for automatic lane keeping in autonomous vehicles, it is useful to consider a state-space model where the state variables are the position error (e_1, i.e., the distance of the center of gravity of the vehicle from the centerline of the lane) and the orientation error with respect to the road (e_2). Then, by introducing the following error variables,

$$\dot{e}_1 = \dot{y} + \int v_x \dot{e}_2 dt, \quad e_2 = \psi - \psi_{des} \quad (13)$$

and by substituting them in Equation (11), the state-space model expressed in terms of the traking error variables can be obtained:

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -2\frac{C_{\alpha f}+C_{\alpha r}}{mv_x} & 2\frac{C_{\alpha f}+C_{\alpha r}}{m} & -2\frac{C_{\alpha f}l_f - C_{\alpha r}l_r}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & -2\frac{C_{\alpha f}l_f + C_{\alpha r}l_r}{I_z v_x} & 2\frac{C_{\alpha f}l_f + C_{\alpha r}l_r}{I_z} & -2\frac{C_{\alpha f}l_f^2 - C_{\alpha r}l_r^2}{I_z v_x} \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2C_{\alpha f}l_f}{I_z} \end{bmatrix} \delta +$$

$$+ \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2C_{\alpha f}l_f}{I_z} \end{bmatrix} d + \begin{bmatrix} 0 \\ -\frac{2C_{\alpha f}l_f - 2C_{\alpha r}l_r}{mv_x} - v_x \\ 0 \\ \frac{2C_{\alpha f}l_f^2 - 2C_{\alpha r}l_r^2}{I_z v_x} \end{bmatrix} \dot{\psi}_{des} \quad (14)$$

where $\delta = \delta_f$ is the front steering angle and d is a disturbance signal. Furthermore, by assuming the following, it is possible to define the rate of change in the reference orientation of the vehicles as $\dot{\psi}_{des} = \frac{v_x}{R}$:

- The vehicle rides at longitudinal velocity v_x on a curved road of radius R.
- R is sufficiently large so that the small angle assumption holds true.

Then, by integrating the latter expressions, it is possible to compute the reference yaw angle ψ_{des}.

Remark 1. *The position of the vehicle in the global (world) coordinates (X, Y) can be computed from the body-fixed coordinates $(\dot{x}, \dot{y})$ as follows:*

$$\begin{cases} X &= \int_0^t (v_x cos(\psi) - \dot{y} sin(\psi)) dt \\ Y &= \int_0^t (v_x sin(\psi) + \dot{y} cos(\psi)) dt \end{cases} \quad (15)$$

The lateral dynamics model given by Equation (14) is a function of the longitudinal vehicle speed v_x, which can be assumed to be a known varying parameter. The value of v_x can be obtained through a standard feedback control law (e.g., PI control) in charge of providing the control action to track a desired speed v_x^d.

3. Problem Statement and Controller Design

In general, a lane-following system is a control system that allows the vehicle to ride along a lane while maintaining a reference-set velocity and a safe distance from the preceding vehicle. This type of system combines longitudinal and lateral control for the following:

- Maintaining a driver-set velocity and to maintain a safe distance from the preceding car by adjusting the acceleration of the vehicle (*longitudinal controller*), and
- Ensuring the vehicle travels along the centerline of the lane by regulating the steering angle (*lateral controller*).

Within the above-described context, the control objective is to make the autonomous vehicle track a desired reference trajectory via regulation of the steering angle δ. To proceed to the controller design, the following compact state-space model of the vehicle lateral dynamics is introduced. We denote the following:

- $x = [e_1, \dot{e}_1, e_2, \dot{e}_2]$, the plant state;
- $u = [\delta]$, the manipulable input;
- $\xi = [d, \dot{\psi}_{des}]$, the plant disturbance;
- $y = x$, the system output; and
- $z = [e_1, e_2,]$, the performance output signals.

Thus, the dynamic model (12) can be rewritten in the following parameter-dependent state-space representation:

$$\Sigma : \begin{cases} \dot{x}(t) = A(v_x(t)) + Bu(t) + D_d(v_x(t))\xi(t) \\ y(t) = Cx(t) \\ z(t) = C_z x(t) \end{cases} \tag{16}$$

where the system matrices depends on the varying parameter $v_x(t)$ that is assumed to be bounded as

$$\underline{v}_x \leq v_x \leq \overline{v}_x \tag{17}$$

The goal of the proposed controller is to reduce the position and the orientation error by controlling the steering angle. Another important issue in the controller design is related to minimization of the effects of the disturbance term $\dot{\psi}_{des}$ that causes the tracking errors to not converge to zero when the vehicle rides along a curve. Then, by assuming that the longitudinal velocity is bounded as described by Equation (17), it is possible to consider static state-feedback control laws as possible controller candidates,

$$u(t) = Kx(t), \tag{18}$$

to be computed by solving a H_∞ optimal control synthesis problem that can be formulated as a convex LMI optimization problem in the unknown matrices $X = X^T$ and Y [25]:

$$\begin{cases} \begin{matrix} min_{X,Y,\gamma} & \gamma \\ s.t. & \end{matrix} \\ \begin{bmatrix} A_i X + BY + XA_i^T + Y_i^T B_i^T & D_{d_i} & XC_z^T \\ D_{d_i}^T & -\gamma I & 0 \\ C_z X & 0 & -\gamma I \end{bmatrix} < 0, \; i = 1, 2 \\ X = X^T > 0 \end{cases} \tag{19}$$

where each vertex $i = 1, 2$ corresponds to the system matrices computed for $v_x = \underline{v}_x$ and $v_x = \overline{v}_x$. If solvable, the optimal H_∞ control gain is given by $K = YX^{-1}$.

4. Simulations

In order to verify the effectiveness of the proposed approach, some simulations have been undertaken under different guidance scenarios. In this respect, in order to consider realistic driving scenarios, a co-simulation environment based on the joint use of the MATLAB/Simulink © and Carsim 8 © packages was set up. The Carsim 8 software allows one to simulate and test autonomous driving systems by providing the possibility to add onboard vision systems as well as sensor fusion algorithms, path planning routines, and vehicle controllers. Visualization features also include the bird's-eye-view plot, sensors (e.g., camera, radar, lidar, etc.) simulation and scope for sensor coverage, detections, and tracks (Figures 12 and 13). The co-simulation environment model is depicted in Figure 14. Essentially, five modules have been included in this model:

1. The **Vehicle Model**, which models the longitudinal and lateral dynamics of the car. The inputs of the model are the longitudinal acceleration and the steering angle; the outputs of the model are the lateral and longitudinal velocities, the XY positions and velocities, the yaw angle, and the yaw rate of the vehicle;
2. The **Carsim Module**, which allows us to include all features related to the simulation of a driving scenario in the simulation environment. This module enables us to configure the vehicle parameters (Figure 12) and the camera point of view (Figure 13a). All of the outputs of the vehicle model are inputs of this module; the outputs of the module is a video containing the current scene (Figure 13b);
3. The **Longitudinal Controller**, which implements the control of longitudinal vehicle speed (v_x) through a PI controller. It computes the acceleration and deceleration commands on the basis of the current reference longitudinal speed. In particular, the controller implements the Stanley method, for which the details can be found in [14].
4. The **Lane Detection and Trajectory Generation Module**, which implements the algorithm reported in Section 2.2 and provides an estimation of the reference yaw angle;
5. The **Lateral Controller**, which implements the H_∞ controller described in Section 3 and provides the steering angle command to the vehicle model. The control algorithm was designed in Matlab/Simulink, and the optimization problem (19) was solved using the SeDuMi Matlab toolbox.

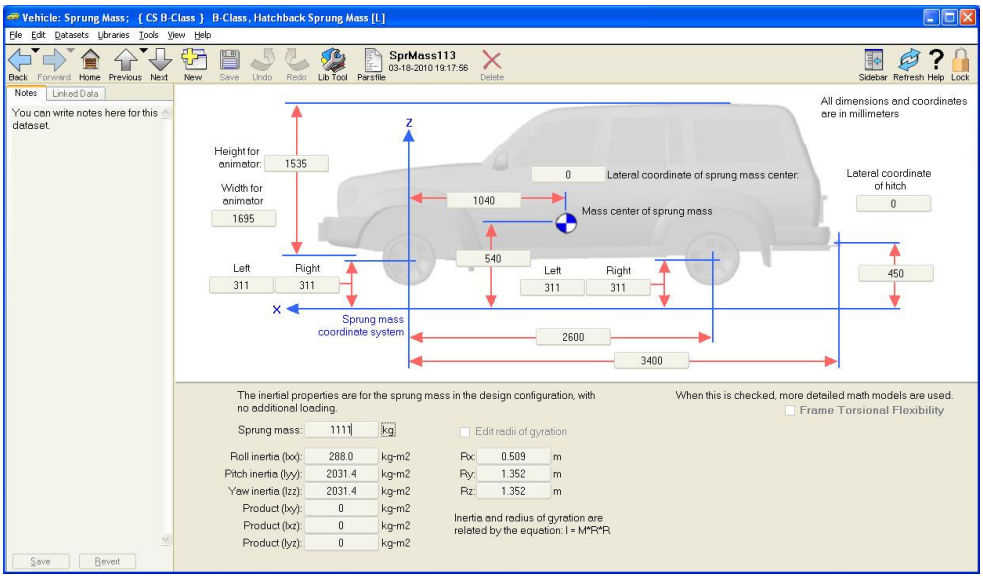


Figure 12. Carsim 8: vehicle parameter configuration.

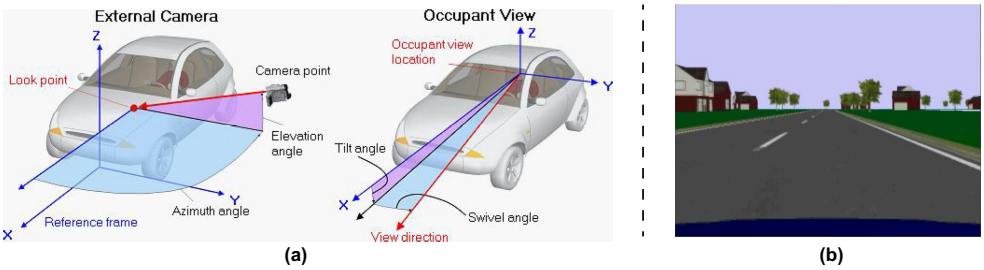


Figure 13. Carsim 8: camera point of view configuration (**a**) and camera front view (**b**).

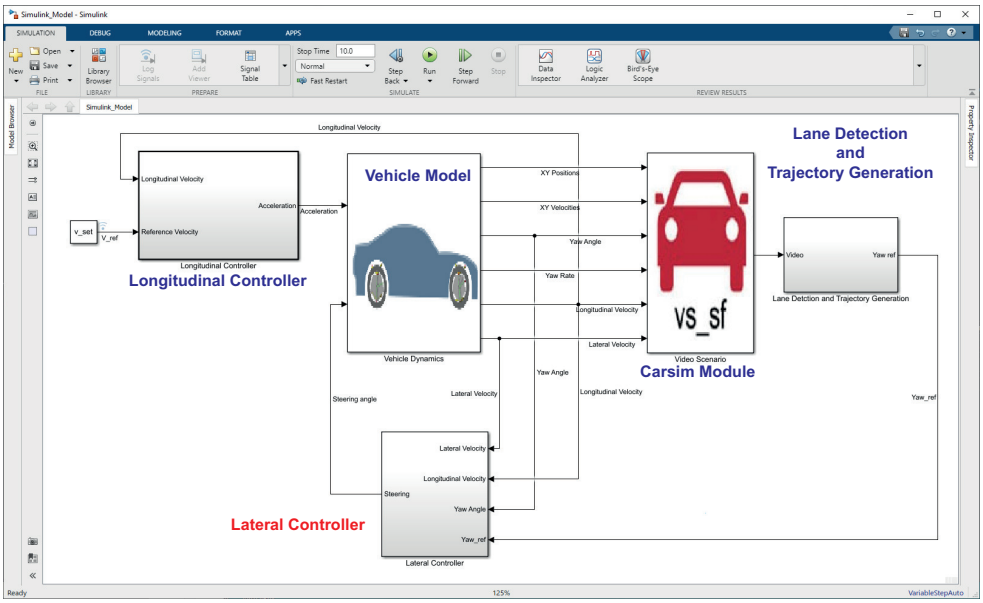


Figure 14. Co-simulation environment.

The value reported in Table 2 were used as vehicle parameters.

Table 2. Vehicle parameters.

Parameter	Value	Description
M	1.575 Kg	Vehicle mass
I_z	2.875[1.2]	Inertia
l_f	1.2 m	Distance of the front tire from the vehicle center of gravity
l_r	1.2 m	Distance of the rear tire from the vehicle center of gravity
$C_{\alpha f}$	19.000 N/rad	Cornering stiffness of front tire
$C_{\alpha r}$	33.000 N/rad	Cornering stiffness of rear tire

Furthermore, in all scenarios accounted for (Figure 15), we considered longitudinal speed always in the range $5 \leq v_x \leq 30$ m/sec.

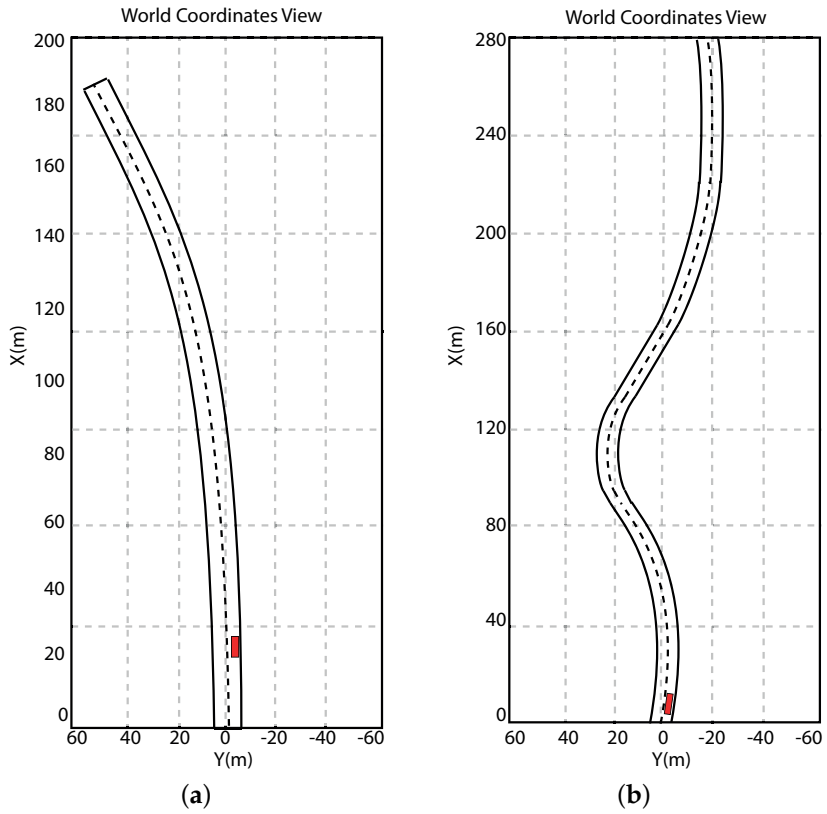


Figure 15. Simulation scenarios: scenario 1 (**a**) and scenario 2 (**b**).

4.1. Scenario 1

The first simulation scenario refers to a driving condition where the autonomous vehicle must turn left on a road. Details about this scenario are reported in Figure 15a. In this condition, the autonomous vehicle must be able to perform the necessary actions allowing the car to ride on the road without taking risks. Then, the onboard electronic devices and the electronic control unit must accomplish the following tasks:

- Estimate the road lane;
- Compute the reference trajectory; and
- Perform the control actions thanks to which the vehicle can follow the computed trajectory.

The results related to this scenario are reported in Figures 16 and 17. In particular, Figure 16 shows a comparison between the reference and the vehicle trajectories in the world coordinates. From this figure, it is evident that the proposed algorithm allows for good performance to be achieved in terms of trajectory followed. This result is also evident in Figure 17a,b, where a comparison between the reference and the measured yaw angle (a) and the position and orientation errors (b) is reported. A more quantitative comparison is reported in Table 3, where the error variables (Equation (13)) averaged along all the simulation time steps are reported.

Table 3. Scenario 1: relative errors.

	e_1 (%)	e_2 (%)
Scenario 1	4.46	5.79

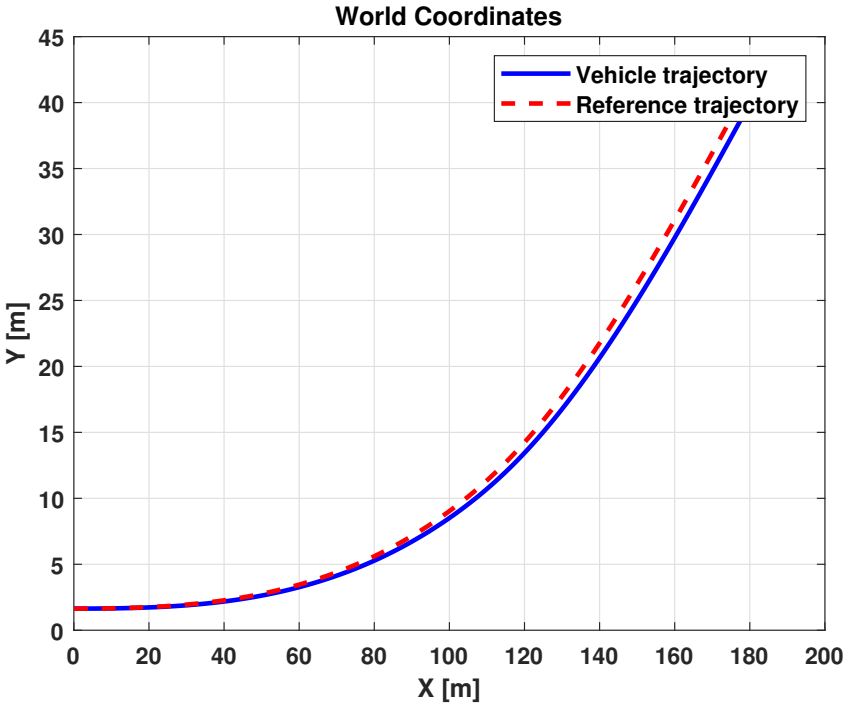


Figure 16. Simulation scenario 1: vehicle and reference trajectories.

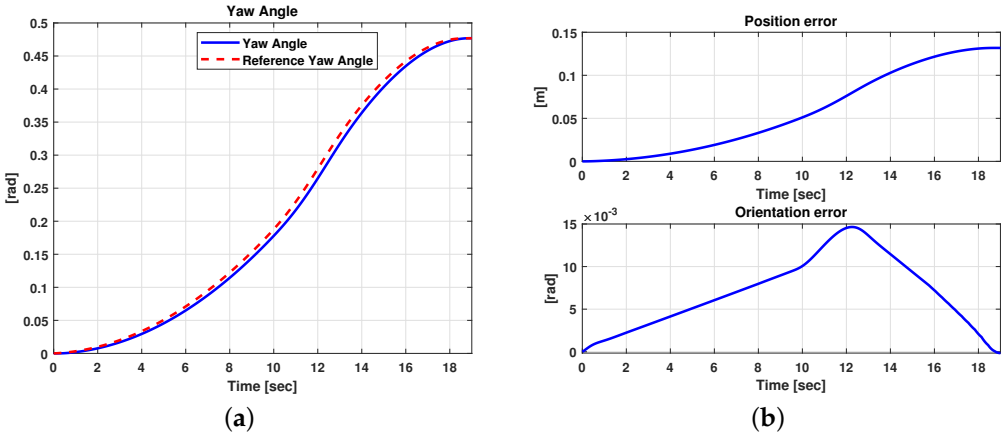


Figure 17. Simulation scenario 1: comparison between reference and measured yaw angles (**a**) and position and orientation errors (**b**).

4.2. Scenario 2

In order to test the proposed method in a more complex driving condition, the scenario depicted in Figure 15b was taken into account. This scenario describes a driving condition where three consecutive road curves must be faced. The results related to this second scenario are reported in Figures 18 and 19. It is evident that, in this case, the controller allows for good performance to be achieved in terms of the trajectory followed. Furthermore, as expected, Figure 19 shows small position and orientation errors, which tend toward zero when the curve ends and a straight road begins. As for the previous scenario, a quantitative performance evaluation was performed In this respect, Table 4 reports the error variables (Equation (13)) averaged along all the simulation time steps.

Table 4. Scenario 2: relative errors.

	e_1[%]	e_2[%]
Scenario 2	5.06	6.19

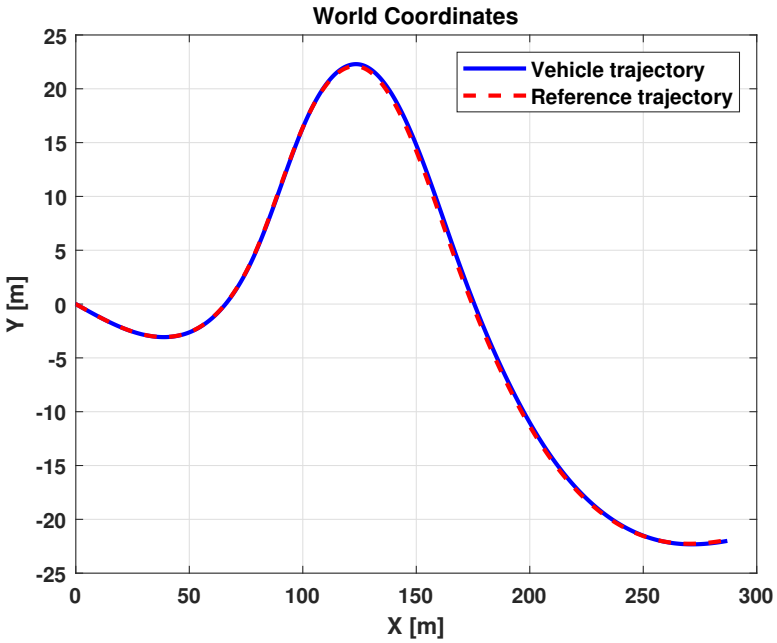


Figure 18. Simulation scenario 2: vehicle and reference trajectories.

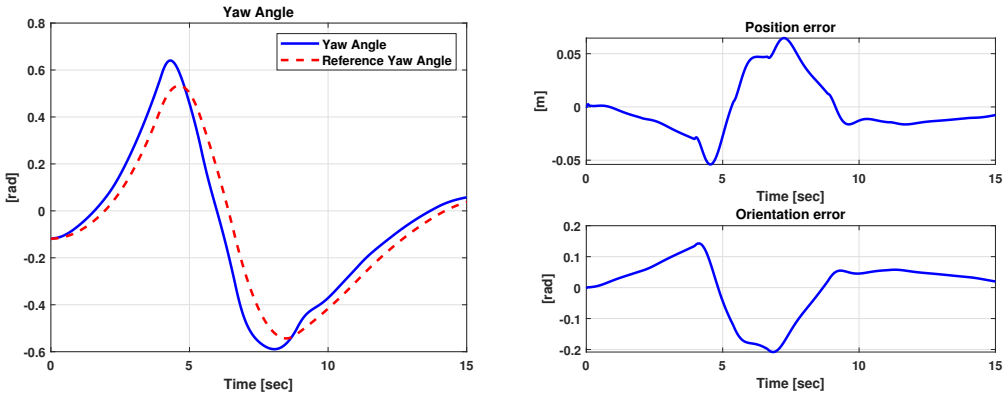


Figure 19. Simulation scenario 2: comparison between reference and measured yaw angles (**a**) and position and orientation errors (**b**).

Remark 2. *Video clips related to validation of the lane-detection algorithm in a real scenario and to the assessment of the steering control algorithm can be found, respectively, at the following links:*

- *Lane detection algorithm in a real application scenario: https://youtu.be/4mcSdDFoivU (accessed on 5 June 2021).*
- *Steering control test in the co-simulation environment: https://youtu.be/CNrHAG6a4QU (accessed on 5 June 2021).*

5. Conclusions

In this paper, a procedure for lateral control of an autonomous vehicle was developed. The proposed approach consists of the design of a controller that is robust with respect to disturbance and variations in longitudinal vehicle speed. The control law was designed

by solving a convex optimization problem and allows for a good trajectory-following performance to be achieved with low online computations. In fact, the optimization problem used for the synthesis is solved offline and only the computed controller gain enters the state-feedback control architecture. The simulations undertaken considered different driving conditions, and the results demonstrate the robustness of the designed control law. Despite these promising results, further efforts must be dedicated to improving the design of both longitudinal and lateral controllers in order to provide an integrated solution. In fact, this work focused on the design of a lateral control based on the assumption of an a priori knowledge of the longitudinal speed. Moreover, the approach only addresses how to eliminate orientation and position errors according to the desired path but does not take into account coupling with the longitudinal dynamics. In this respect, future work will address the design of a coupled lateral and longitudinal controller taking into account the linear parameter varying (LPV) framework as a possible solution [33]. The choice of the LPV framework is convenient in that it allows one use a single model to globally describe the dynamics of a nonlinear system subject to transitions between different working points and conditions, related to changes of some fundamental system parameters [34] and amenable for direct use for control synthesis purposes.

Author Contributions: Conceptualization, G.G.; data curation, G.G.; investigation, M.L.; project administration, A.C. and G.G.; resources, M.L.; software, G.G. and G.C.; supervision, A.C.; validation, G.C.; writing—original draft, G.G., G.C., and M.L.; writing—review and editing, G.G., G.C., M.L., and A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Eskandarian, A. *Handbook of Intelligent Vehicles*; Springer: London, UK, 2012,
2. Winner, H.; Hakuli, S.; Lotz, F.; Singer, C. *Handbook of Driver Assistance Systems, Basic Information, Components and Systems for Active Safety and Comfort*; Springer: Cham, Switzerland, 2016.
3. Chen, M.-M.; Chen, M.-C. Modeling Road Accident Severity with Comparisons of Logistic Regression, Decision Tree and Random Forest. *Information* **2020**, *11*, 270. [CrossRef]
4. Deng, Q.; Soeffker, D. A Review of the current HMM-based Approaches of Driving Behaviors Recognition and Prediction. *IEEE Trans. Intell. Veh.* **2021**. [CrossRef]
5. Virgilio, G.V.R.; Sossa, H.; Zamora, E. Vision-Based Blind Spot Warning System by Deep Neural Networks. In *Pattern Recognition*; Figueroa Mora, K., Anzurez Marín, J., Cerda, J., Carrasco-Ochoa, J., Martínez-Trinidad, J., Olvera-López, J., Eds.; MCPR 2020; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2020; Volume 12088, pp. 185–194. [CrossRef]
6. Arce, F.; Zamora, E.; Fócil-Arias, C.; Sossa, H. Dendrite ellipsoidal neurons based on k-means optimization. *Evol. Syst.* **2019**, *10*, 381–396. [CrossRef]
7. Galvani, M. History and future of driver assistance. *IEEE Instrum. Meas. Mag.* **2019**, *22*, 11–16. [CrossRef]
8. Sowmya Shree, B.V.; Karthikeyan, A. Computer Vision based Advanced Driver Assistance System Algorithms with Optimization Techniques-A Review. In Proceedings of the 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 29–31 March 2018; pp. 821–829. [CrossRef]
9. Tagne, G.; Talj, R.; Charara, A. Higher-order sliding mode control for lateral dynamics of autonomous vehicles, with experimental validation. In Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, QLD, Australia, 23–26 June 2013. [CrossRef]
10. Zhao, P.; Chen, J.; Mei, T.; Liang, H. Dynamic motion planning for autonomous vehicle in unknown environments. In Proceedings of the Int. IEEE Conference on Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 5–9 June 2011.
11. Marino, R.; Scalzi, S.; Netto, M. Nested PID steering control for lane keeping in autonomous vehicles. *Control. Eng. Pract.* **2011**, *19*, 1459–1467. [CrossRef]
12. Zhang, H.; Wang, J. Vehicle lateral dynamics control through afs/dycand robust gain-scheduling approach. *IEEE Trans. Veh. Technol.* **2016**, *65*, 489–494. [CrossRef]

13. Han, G.; Fu, W.; Wang, W.; Wu, Z. The lateral tracking control for the intelligent vehicle based on adaptive pid neural network. *Sensors* **2017**, *17*, 1244. [CrossRef] [PubMed]
14. HHoffmann, G.M.; Tomlin, C.J.; Montemerlo, M.; Thrun, S. Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. In Proceedings of the American Control Conference, New York, NY, USA, 9–13 July 2007; pp. 2296–2301. [CrossRef]
15. Gutjahr, B.; Groll, L.; Werling, M. Lateral vehicle trajectory optimization using constrained linear time-varying MPC. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1586–1595. [CrossRef]
16. Falcone, P.; Borrelli, F.; Asgari, J.; Tseng, H.; Hrovat, D. Predictive active steering control for autonomous vehicle systems. *IEEE Trans. Control Syst. Technol.* **2007**, *15*, 566–580. [CrossRef]
17. Gallep, J.; Govender, V.; Müller, S. *Model Predictive Lateral Vehicle Guidance Using a Position Controlled EPS System*; IFAC-PapersOnLine; Elsevier: Amsterdam, The Netherlands, 2017; Volume 50, pp. 265–270.
18. Cafiso, S.; Pappalardo, G. Safety effectiveness and performance of lane support systems for driving assistance and automation-Experimental test and logistic regression for rare events. *Accid. Anal. Prev.* **2020**, *148*, 105791. [CrossRef] [PubMed]
19. Pappalardo, G.; Cafiso, S.; Di Graziano, A.; Severino, A. Decision Tree Method to Analyze the Performance of Lane Support Systems. *Sustainability* **2021**, *13*, 846. [CrossRef]
20. Geng, K.; Liu, S. Robust Path Tracking Control for Autonomous Vehicle Based on a Novel Fault Tolerant Adaptive Model Predictive Control Algorithm. *Appl. Sci.* **2020**, *10*, 6249. [CrossRef]
21. Park, H.G.; Ahn, K.K.; Park, M.K.; Lee, S.H. Study on Robust Lateral Controller for Differential GPSBased Autonomous Vehicles. *Int. J. Precis. Eng. Manuf.* **2018**, *19*, 367–376. [CrossRef]
22. Chen, C.; Shu, M.; Wang, Y.; Liu, R. Robust H_∞ Control for Path Tracking of Network-Based Autonomous Vehicles. *Math. Probl. Eng.* **2020**, *2020*, 2537086. [CrossRef]
23. Wang, R.; Sun, Y.; Lin, M.; Zhang, H. Research on bus rollstability control based on LQR. In Proceedings of International Conference on Intelligent Transportation, Big Data and Smart City, Halong Bay, Vietnam, 19–20 December 2015; pp. 622–625.
24. Cario, G.; Casavola, A.; Lupia, M. Lane Detection and Tracking Problems in Lane Departure Warning Systems. In *Computer Vision and Imaging in Intelligent Transportation Systems*; Loce, R.P., Bala, R., Trivedi, M., Eds.; John Wiley & Sons: Pondicherry, India, 2017.
25. Chilali, M.; Gahinet, P.M. H_∞ design with Pole Placement Constraints: An LMI Approach. *IEEE Trans. Aut. Contr.* **1996**, *41*, 358–367. [CrossRef]
26. Available online: https://www.avsupply.com/ITM/12323/FIREFLY$%$20MV.html (accessed on 5 June 2021).
27. Available online: https://www.flir.com/iis/machine-vision/ (accessed on 5 June 2021).
28. Jung, C.R.; Kelber, C.R. A lane departure warning system based on a linear-parabolic lane model. In Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processin, Parma, Italy, 14–17 June 2005.
29. Rajamani, R. *Vehicle Dynamics and Control*; Springer: New York, NY, USA, 2006.
30. Rutter, J.W. *Geometry of Curves*; Chapman & Hall/CRC. Routledge: London, UK, 2000.
31. Available online: https://www.ti.com/lit/an/spra920/spra920.pdf (accessed on 5 June 2021).
32. Kong, J.; Pfeiffer, M.; Schildbach, G.; Borrelli, F. Kinematic and dynamic vehicle models for autonomous driving control design. In Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea, 28 June–1 July 2015. [CrossRef]
33. Alcalá, E.; Puig, V.; Quevedo, J. *LPV-MPC Control for Autonomous Vehicles*; IFAC-PapersOnLine; Elsevier: Amsterdam, The Netherlands, 2019; Volume 52, pp. 106–113, ISSN 2405-8963. [CrossRef]
34. Gagliardi, G.; Tedesco, G.; Casavola, A. A LPV modeling of turbocharged spark-ignition automotive engine oriented to fault detection and isolation purposes. *J. Frankl. Inst.* **2018**, *355*, 6710–6745. [CrossRef]