POLITECNICO DI MILANO

Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in

Computer Science and Engineering

Autonomous vehicle heading and centerline displacement estimation via computer vision

Advisor: PROF. MATTEO MATTEUCCI

Co-advisor: DR. SIMONE MENTASTI

Master Graduation Thesis by:

PAOLO CUDRANO

Student Id n. 878148

Academic Year 2018-2019

POLITECNICO DI MILANO

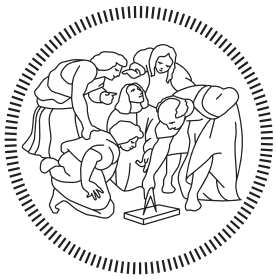
Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Corso di Laurea Magistrale in

Computer Science and Engineering



Autonomous vehicle heading and centerline displacement estimation via computer vision

Relatore: PROF. MATTEO MATTEUCCI

Correlatore: DR. SIMONE MENTASTI

Tesi di Laurea Magistrale di:

PAOLO CUDRANO

Matricola n. 878148

Anno Accademico 2018-2019

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

`http://code.google.com/p/classicthesis/`

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

`http://postcards.miede.de/`

This template has been adapted by Emanuele Mason, Andrea Cominola and Daniela Anghileri: *A template for master thesis at DEIB*, June 2015.

A Mamma e Papà

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Prof. Matteo Matteucci, and my co-advisor Simone Mentasti, for all the help they gave me during the completion of this work.

Thanks also to all my friends for their support. Particular mention goes to Afroja for always being with me.

Last but by no means least, I'm grateful to my beloved mother for supporting me during my studies and for raising me the way she did, regardless of all the difficulties that came into our way.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The ability of autonomous vehicles to maintain an accurate trajectory within their own road lane is crucial for their safe operation. In the context of lane following, a measurement of the position of the road lines is required, together with an estimation of lateral displacement and orientation of vehicle. At times, the shape of the road centerline is also required. While several systems are described in the literature, a sharp cut is often made between perception and control, losing a potentially beneficial link. Moreover their evaluation is rarely performed with data from real scenarios. In this work, we propose the design of a vision-based perception system for lane following, capable of robustly detecting the ego-lane and the parameters needed for controlling the vehicle's trajectory. A CNN is employed for the detection of the line markings, which are then fit in the BEV space to a mathematical model. The selected representation is given in the intrinsic Whewell coordinates to facilitate the estimation of the centerline, for which a specific algorithm is proposed. Heading and lateral displacement of the vehicle are then computed, and an EKF is employed to stabilize the results. The overall performance of the system have been evaluated on real-world data, collected simulating different driving conditions and featuring different road geometries.

SOMMARIO

La capacità di un veicolo a guida autonoma di mantenere una traiettoria accurata all'interno della propria corsia stradale è cruciale per il loro utilizzo in sicurezza. Nel contesto dei sistemi di mantenimento automatico della corsia, la misurazione della posizione della linea stradale è richiesta, insieme alla stima di scostamento laterale e orientamento del veicolo. Alle volte, la forma della linea di centrostrada è ulteriormente richiesta. Mentre diversi sistemi sono descritti in letteratura, è spesso presente un taglio netto tra sistemi di percezione e controllo, il che fa perdere un collegamento potenzialmente benefico. Inoltre, la loro validazione è raramente eseguita utilizzando dati acquisiti in scenari reali. In questo lavoro dunque proponiamo la progettazione di un sistema di percezione basato su visione per il problema del mantenimento automatico della corsia, in grado di rilevare in maniera robusta la propria corsia stradale e i parametri necessari al controllo della traiettoria. Una CNN è utilizzata per la rilevazione delle linee stradali, per le quali viene poi stimato un modello nello spazio della BEV. La rappresentazione scelta è data quindi in coordinate di Whewell in modo da facilitare la stima della centerline, per la quale uno specifico algoritmo viene proposto. Orientamento e scostamento laterale del veicolo sono quindi calcolati, e un EKF è utilizzato per stabilizzare i risultati. Le performance dell'intero sistema sono quindi state valutate su dati reali, raccolti simulando diverse condizioni di guida e geometrie stradali.

1

INTRODUCTION

From science fiction novels to research labs all over the world, self-driving cars have made quite a long way in the last decades, but still considerable effort has to be invested before they can finally make it into our everyday life.

The concept of autonomous driving, or more precisely *driving automation* according to the Society of Automotive Engineers (SAE) [1], can be defined as the process of providing a vehicle with the capabilities and the resources necessary to execute sustained driving tasks with limited or no human intervention.

A substantial amount of research has been dedicated lately to this topic, with effort coming from both the industry and the academia. Several reasons are certainly behind this peak of interest, but the most influential is likely to be the higher level of safety fully autonomous vehicles would offer in comparison with traditional means of transportation. According to the analyses performed by the U.S. National Highway Traffic Safety Administration (NHTSA) in the last 10 years, over 90% of the car crashes registered in the U.S. are believed to be

caused by human errors [2]. Furthermore, they reported that alcohol and drugs abuse, driver distraction and fatigue account for about 40% of the fatal ones [3], while according to their 2012 survey respectively 30% and 14% of the fatal crashes for that year involved speeding and lane departure [3], [4]. Analogous figures are observed in Italy, where the Italian national institute of Statistics (Istituto Nazionale di Statistica, ISTAT) highlights how most of the accidents in the country are due to human misbehavior (93.7%), with distraction, failure of yield right-of-way and speeding accounting overall for 40% of the total amount [5].

Although autonomous vehicles will hardly be impeccable, they would not certainly display most of these disruptive human flaws, and thus determine a potentially large improvement for the safety of drivers, passengers and pedestrians on the road. Not of primary importance for the many, but the reduction of car crashes would also bring economic benefits, given that the annual cost of road accidents in the U.S. is estimated at $277 billions [3].

Other incentives in favor of autonomous vehicles can be found in their inherent capabilities at performing optimizations, given their computational power. This added skill could indeed lead to economical and environmental benefits, with better route planning, traffic management, and optimal fuel consumption [6], as well as a potential increase in the adoption of car sharing alternatives and a consequent reduction of parking needs [7]. Furthermore, in a completely autonomous scenario, they would relief drivers from the stress caused by commuting, especially in large cities, and instead leave them with more time to focus on other activities while taking their everyday rides or even traveling longer distances [6], [8].

With these incentives in mind, a discrete amount of work has been performed in the last decades. With its early conception going as back as at least the 1980s [9], [10], the main forward thrust to the field was certainly given by the DARPA Grand Challenge [11], [9], organized by the Defense Advanced Research Projects Agency (DARPA, of the Department of Defense of the United States) as a way to promote the development of unmanned ground vehicles. After no participant

succeeded in the first edition of the challenge in 2004, the attendance nearly doubled the following year, leading to almost all of the participating vehicles outperforming the results of the previous edition. As the challenge was finally completed by several teams [11], the ground was clear for the organization of the DARPA Urban Challenge in 2007 [12] and the intensification of the research in the area.

In 2014, SAE International issued a Recommended Practice document [13] in order to harmonize the emerging field of autonomous driving with standardized definitions of the actors and features involved. In this document and its successive revisions of 2016 [14] and 2018 [1], they describe a taxonomy for autonomous vehicles composed of 6 levels, ranging from no autonomous features (level 0) to full autonomy (level 5). Following this classification, at the moment only level 1 and 2 features have been successfully developed and are currently implemented by many car manufacturers. Among these are adaptive cruise control, assisted parking and lane keeping technologies. Such systems however are only meant to support the driver during the ride and are not guaranteed to behave optimally and safely all the times, leaving the driver in charge of their supervision and responsible to correct any misbehavior encountered.

To achieve a partial notion of driver disengagement, although valid only under particular, strict conditions, level 3 is needed. Notably however, only one potential level 3 vehicle has ever been released yet. It is the case of the 2019 Audi A8 from the German manufacturer Audi AG, featuring their Traffic Jam Pilot, which is allowed to take full control of the vehicle when in severe traffic. Nevertheless, its full potential is currently unexploited because not yet approved for sale by the regulators of most countries [15].

In this context, it is clear that many advances are still needed to reach a fully functional level 5 technology. On one hand, it is necessary to develop new algorithms in order to achieve full autonomy in increasingly more scenarios, from urban traffic jams to unpaved steep mountain roads. Likewise, the systems currently available have to be significantly improved, in order to provide high reliability and

robustness to the multitude of unpredictable situations they will be required to face.

In doing so, engineers need to focus at the same time on enabling the vehicles to sense their surrounding environment and on designing control techniques capable of governing their operation consistently and in safety. The perceptual aspect is indeed fundamental for the vehicle not only to determine its own position, but also, given a functioning control system, to determine how to reach its destination. This, in turn, has to be done both on a global scale, selecting for instance what roads to follow on a map, but also on a local scale, dynamically updating the designed trajectory in order to remain on such road and avoid collisions with possible obstacle on its way.

With this in mind, it is exactly on the latter task that we base our research, as the meeting point between perception and control. Here indeed a particular problem caught our attention, as it has been long studied, and yet there is margin for several improvements not only in the more technical aspects, but also in the overall approach taken towards it.

In this thesis indeed we focus on the problem of *lane following* for autonomous vehicles, taking a perspective predominantly oriented to the perceptual aspects of the task, but keeping also well in sight the actual requirements and limitations of its control counterpart. We thus begin by studying how line detection systems work and perfecting one ourselves; but we then move one step further. A large amount of literature has been produced in the last decades on line detection systems [16], [17], analyzing what features to extract and how to extract them, what curve models suit best their shape, what projective mappings could be helpful in the process and which filtering techniques achieve better results. However, all of these systems have always been considered as self-standing and thus developed in isolation. Nonetheless, while their outputs could still be compared with ground truth measurements and their results can indeed be quantitatively evaluated, they are actually not targeting the control goal.

What is really needed by lane keeping systems indeed is not a detailed description of every sign on the road (which is likely to be

computationally expensive to obtain, and still potentially inaccurate). Instead, it is an algorithm that can robustly describe only a few, key parameters of the overall scene and can be then reliably used by a control system in order to govern the trajectory of the vehicle. For these reasons, we do not only aim at detecting the line markings delimiting our vehicle's lane, as many other works do. Instead, we additionally refine their representations in order to be optimally employed by the subsequent controller. For this reason, we develop a technique for estimating at any given time the shape and position of the centerline, along with the vehicle's relative heading and lateral displacement, parameters of fundamental importance for lane following control systems.

In the remaining chapters, we start our analysis with an overview of the lane following problem and a presentation of the state of the art in this field. To this extent, the literature on line detection algorithms is first expanded (Chapter 2), followed by a portrayal around the generality of the lateral control problem, with particular emphasis on lane following systems (Chapter 3).

In Chapter 4 our work is described. At first, we present the line detection system we developed, going into the details of each stage of our processing pipeline. Then, our studies on the centerline shape retrieval are discussed, describing at last our approach to the estimation of heading and lateral displacement of the vehicle.

We validate our system in Chapter 5, presenting our experimental setup, the custom dataset we recorded and the results we finally obtained.

At last, Chapter 6 draws our final conclusions and highlights our contributions, closing with an analysis of the future developments that could originate from our work.

2

LINE DETECTION

The first segment of our analysis concerns the problem of line detection. This, in the context of autonomous driving, focuses on locating the position of particular road markings in the proximity of the vehicle. The search could be directed only to lateral lines or include for example stop lines, zebra crossings and possible text indications painted on the pavement. Although dependent on the specific requirements of each system, the final output of this process is in most cases a mathematical representation of shape and relative position of each detected line.

These results are usually achieved using one or more cameras, as they can easily capture colors and all the features used by humans for the same task [18]. In particular, single camera systems are considered in most of the approaches, because of their ease of installation and limited costs [17]. Stereo vision systems are however sometimes adopted. This is mostly due to their capability to estimate distances on the scene [18], task harder to perform in monocular approaches.Nonetheless, other sensors could also be employed. Hata and Wolf [19] use a LIDAR to detect dashed and continuous lines,

together with crosswalks and the position of road boundaries. Specifically, they analyze the reflective intensity of the measured point cloud and isolate all road markings thanks to their higher reflectivity. Similar approaches are taken also by Kibbel et al. [20] and Lindner et al. [21]. While camera systems are easily affordable however, a LIDAR sensor has significantly higher costs, making their usage also an economic choice. On the other hand, LIDAR systems can accurately measure distances and are not affected by changes in illumination, as opposed to traditional cameras. To obtain some advantages from both sensors, Huang et al. [22] propose a detection system for multiple lanes that combines vision and LIDAR. They detect each line marking mainly from their visual data, but integrate then the findings with the laser measurements to filter out obstacles and enforce the position of the road boundaries. In this fashion, they are able to reduce the presence of false positives and thus improve the overall performances.

Another sensing option is represented by infrared (or thermal) cameras. They can be used indeed thanks to their analogous capability to detect different reflectivity values. To this end, Fardi and Wanielik [23] successfully studied their usage in road boundaries detection, but no further research has been devoted towards their employment with road markings. It is likely however, in our opinion, that this negligence can be partly attributed to the high economic cost of the sensors themselves compared to their potential advantages.

Given then the highlighted scenario, and following the tendency of the literature, in the remainder of this work we focus only on monocular camera systems, favoring their competitive costs and ease of operation.

If a camera is indeed chosen, the detection is performed by means of computer vision. While no universal algorithm has been developed for this task, a large literature of techniques is available, each exploring different alternatives or combinations of them. Nevertheless, according to the literature [18], almost all systems are constructed following a similar processing pipeline, which we further analyze in the remainder of this chapter.

Only exception to this view is the case of some learning-based methods. This class of approaches have arisen in the last years thanks to the enormous improvements registered by deep learning methods and convolutional neural networks. With these tools, the traditional techniques have been revolutionized. However, while in most cases learning approaches are integrated with the pipeline, substituting or improving a single stage within it, end-to-end approaches have also been developed. These methods do not rely on any pipeline and are dependent only on a learning technology, thus differing completely from traditional techniques.

As both approaches deserve our attention, in the rest of this chapter we present a detailed description of each stage of the traditional pipeline, along with a brief overview of end-to-end systems and their novel approach.

2.1 PROCESSING PIPELINE FOR TRADITIONAL SYSTEMS

As confirmed by the literature [18], [17], the most common vision-based processing pipeline for line detection systems is composed of four stages: preprocessing, feature extraction, model fitting and line tracking (Figure 2.1). From a system to another however, a few differences could be noted. In some works, for instance, we cannot distinguish between the latter two stages, as they are both coupled and indivisible [24]. Nonetheless, a general description of the main architecture should provide enough details to appreciate also its variations.

2.1.1 PREPROCESSING

For each frame acquired, the processing begins with some preliminary operations. Their aim is to reduce the clutter in the recorded images as well as to highlight the relevant information they contain.

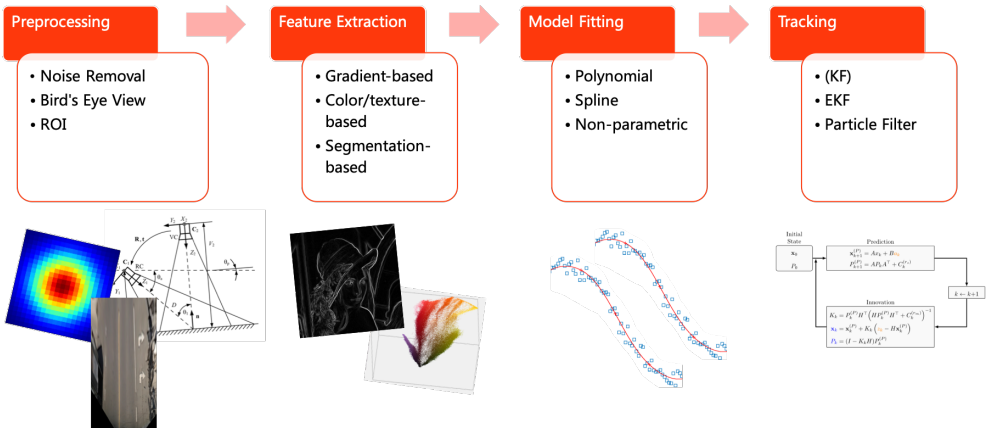


Figure 2.1: Common processing pipeline for line detection systems.

For this purpose, one common operation has to do with the reduction of acquisition noise. Ding et al. [25], for instance, design a specific two-dimensional Gaussian kernel to reduce the impact of noise and brighten the color of line markings. For the same purpose, Gaussian smoothing is used by Kim and Lee [26]. In addition, depending on the camera used, other sources of acquisition noise could become relevant, leading for example to vignetting or wrong exposure [18]. Yet, several modern cameras can correct these artifacts internally with real-time adaptive techniques [27], or at least provide complete access to their internal configurations to allow a third-party software to perform the changes [28]. For this reason, their treatment is outside the scope of this work.

The problem worsens when we consider also the effect of different weather conditions and night-time illumination, as they severely distort how colors are captured [29, Chapter 2 and 3]. It is therefore important for the system to cope with a large range of disturbances, from the lens flare due to direct sunlight to the soft light of rainy days, and from the reduced view in presence of fog to the artificial illumination during night-time [18]. Moreover, irregular shadows are often present on the road, coming from trees or other vehicles, and abrupt illumination changes can additionally be registered when entering and exiting a tunnel or passing under a bridge [18], [16]. To mitigate this conditions, Huang et al. [22] utilize date, time and Earth location to estimate the

position of the sun and correct their detections in areas under direct sunlight. In an analogous context, Rasmussen [30] describes a similar approach to predict the position of shadows and reduce the presence of false positives. Other solutions often involve the transformation of the image into different color spaces. In this context, Sun et al. [31] propose the HSI space to enhance the color of each road marking and mitigate the illumination changes. Son et al. [32] instead claim that most line features are easily retrieved in the YCbCr space, and construct their algorithm based on this consideration.

Shifting our attention from color to shape, it is known that every shape, when captured on an image, is geometrically distorted. This effect is due to the perspective projection of the 3-dimensional world object to the 2-dimensional space of the camera [33]. For this reason, for example, lines that are parallel in the world appear to intersect in the image. Although natural, this effect additionally hinders the detection of lines, as it prevents any algorithm from reasoning on their real-world shape, enforcing their parallelism or comparing them with local maps. To mitigate this issue however, an additional preprocessing operation can be applied. This consists in the rectification of the image by means of perspective warping, to obtain the so-called Bird's Eye View (BEV) [34], [35]. Its computation requires the camera to be calibrated, operation that could be performed once, beforehand, or repeatedly at run-time with more sophisticated algorithms [36]. Many are the systems relying on this procedure , with some using it to enhance edges and similar features [37], [24], [38], while others to rectify the road lines and represent them with an accurate world model [39], [40].

Besides the camera calibration parameters, in some cases also the vanishing points are estimated directly from the scene [42]. Worth mentioning is, among all, the CHEVP (Canny/Hough estimation of vanishing points) algorithm, designed by Wang et al. [43] in their work on line detection. Often, the vanishing points are used as a support for different tasks, as the definition of a Region of Interest (ROI) [44], [32] or the adaptive calibration of the main camera [38]. Sometimes instead, their computation is even exploited within the line fitting stage [45].

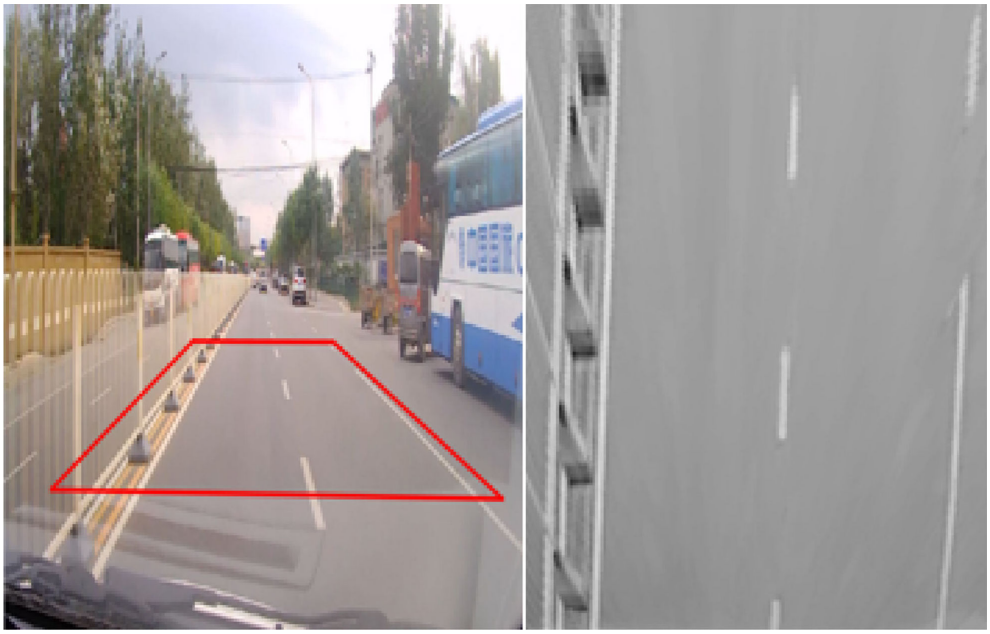


Figure 2.2: Example of a fixed ROI highlighted on an image (left) and its BEV projection (right). Adapted from [41, Figure 1].

As we mentioned above, it is not uncommon to see algorithms focusing only on a particular region of the scene, named the Region of Interest (ROI). This allows them to save computational resources, while also focusing their attention on what they assume to be the most important portion of the scene. While its boundaries are usually fixed and predefined [18], some variations are possible. To further reduce the computational needs of their system, Wu et al. [46] redefine their ROI according to its content, in order to minimize its dimension. For the same purpose, Hsiao et al. [47] use the position of the vanishing points and estimate where the lines should be found. Finally, Mammeri et al. [48] even exploit a segmentation algorithm (Maximally Stable Extremal Regions, or MSER) to shrink the ROI for the subsequent stages.

At last, we notice from the literature that, at times, the vehicle is assumed to feature also other components. Particularly common is in this case the presence of an object detection system, essential to identify vehicles and pedestrians on the road and thus avoid collisions. When this information is processed externally and is thus provided

with no costs, it can be certainly exploited also by a line detection system. In this fashion, Huang et al. [22] evaluate in their algorithm the presence of occluding objects around the vehicle, in order to discard the portions of their images where these are found. Although they rely on a LIDAR for the measurements, their technique can be traced back to the well-known GOLD system of Bertozzi and Broggi [49]. In this pioneering work however, the authors not only consider the obstacle detection as an integral part of the line detection system, but also they completely rely on vision for both tasks.

2.1.2 FEATURE EXTRACTION

Once the image has been prepared, it is important to extract some of the features characterizing the road markings [18]. To this end, this stage is ultimately required to determine which parts or pixels of an image are likely to belong to a road marking. Although numerous algorithms and techniques have been developed throughout the years, they can be classified in four groups, each based respectively on gradient, color, segmentation and machine learning.

Gradient-based methods are designed noticing that the color difference between each road marking and the pavement gives rise to a strong edge. They thus propose to locate the road markings by looking for relevant edges in the image, which usually requires evaluating their gradients. Most of these methods can be applied, in principle, to both the front-view image recorded or its projection into BEV. Among the simplest methods, traditional edge detectors such as the Sobel operator and the Canny algorithm [29, Chapter 5] have been widely adopted [44], [45], especially for their fast execution, although they require a fine-grain tuning of their parameters and are very sensitive to noise and illumination changes. To attenuate this problem, steerable filters are at times used [50], while not uncommonly custom techniques are designed to better improve their response. Jiang et al. [24] combine a customized distance transform (DT) with a simple edge detector to eliminate false edge points. To speed up the computation instead, Li

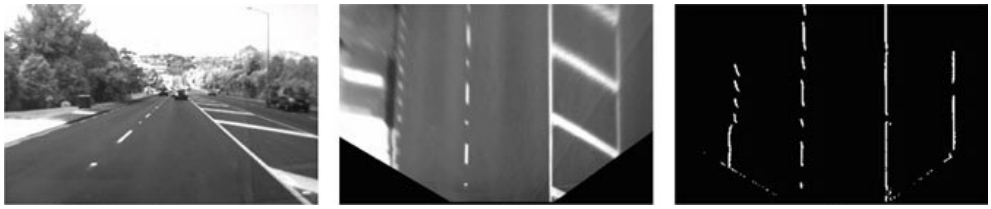


Figure 2.3: Gradient-based features highlighting strong vertical edges in BEV. Adapted from [24, Figure 2].

and Nashashibi [51] proposes a modified version of the Canny detector with some observation on the expected shape of the line markings.

Instead of edges, some systems rely on color features to highlight the position of each marking. They thus search the image for white or yellow regions strongly standing out against the pavement. To this end, they often require the image to be projected in different color spaces. As a clear added benefit, these techniques can further classify lines according to their color, enabling subsequent modules to analyze their semantic meaning according to the traffic laws. In this context, after studying the properties of line markings in the YCbCr, Son et al. [32] propose a threshold-based mechanism to separately isolate white and yellow road markings, arguing that their technique is robust against changes in the illumination conditions.

Less common, segmentation algorithms can be used to isolate the road from the rest of the scene, or even to isolate the lane markings from the pavement., as proposed by González and Özgüner [52] in their system based on histogram-based segmentation.

Last but not least, the large improvements registered in recent times by machine learning algorithms gave rise to a new class of learning-based techniques for the detection of lines. The early work of Kim [53] in this sense explored the usage feed-forward neural networks, support-vector machines (SVM) and naive Bayes classifiers to determine from an image patch if a pixel belongs to a line marking. In his experiments moreover, all these methods outperformed a traditional baseline. Nevertheless, these learning techniques remained mostly unnoticed in the subsequent years, until the real revolution arrived with convolutional neural networks (CNN). Already in 2014,

Kim and Lee [26] introduced a CNN as a fall-back mechanism for their traditional gradient-based algorithm. Shortly after, Huval et al [54] present their positive analysis on the application of CNNs to the overall line detection task, based on the attempts of several research groups during the previous years. Following the same path, He et al. [55] develop their Dual-View CNN (DVCNN), a network relying on both front-view and BEV, and capable of detecting line markings and other road signs painted on the pavement. More standard is instead the architecture proposed by Chen et at. [56], where a computationally optimized encoder-decoder CNN is in charge of detecting road lines and highlighting its position on an image mask.

To conclude this section, we must mention that most of the presented methods can also be combined into an hybrid form, as demonstrated by the work of Southall and Taylor [57].

2.1.3 MODEL FITTING

Knowing where all line markings are, traditional systems usually aim at describing the lines with a mathematical representation. This benefits them as it is usually more compact to store and easier to manipulate for the estimation of derived quantities, such as the relative orientation and the curvature of each line.

As usually more than one road line is present in the scene, a common issue when coming to this stage is the association of each feature points to its respective line. This effect is often worsen by the fact that several feature detectors do not merge together each segment of a dashed line, consequently splitting the line in several pieces. Although no affirmed solution can be found in the literature, several works rely on probabilistic methods. RANSAC, in particular, is used by Kim [53] to robustly select the most appropriate control points for its fitting procedure. Its variation of the algorithm is combined with a segment-grouping mechanism to fit left and right lines at once and evaluate their adequacy. Borkar et al. [58] instead combine for the same purpose RANSAC with a low-resolution Hough transform. More complex algorithms have also been designed. In their work,

Hur et al. [59] introduce the concept of supermarkings and propose an algorithm to link them with remarkable results. They combine a low-level association method with an energy minimization algorithm based on conditional random field (CRF) graphs, finally demonstrating how their system is capable to recognize and correctly group even intersecting dashed lines. Reusing the concept of supermarkings, Chen et al. [56] design a similar grouping algorithm to postprocess the output of their CNN. At first, they aggregate nearby feature points with a clustering algorithm, and then they rely on the connected component labeling (CCL) technique to link them together when corresponding to the same line. Furthermore, following the rising trend highlighted in Section 2.1.2, recent works rely more consistently on learning methods also for these aspects. In this context, Neven et al. [40] design their deep learning approach, LaneNet, to not only recognize each line markings, but also to separate them into each different line. To do this, they introduce a custom CNN decoder, developed to generate specific features for each image point. The values they assume are then trained so that, when processed by a clustering algorithm, they lead each line into a different cluster.

Once the feature points are ready, a fitting algorithm can finally find a compact representation of the underlying line. Before discussing the algorithms however, a model has to be chosen. For road lines, a large variety of models is available, and according to the literature, we can distinguish between parametric and non-parametric ones.

Even before analyzing the most common choices however, another clarification has to be made. While some authors fit their models directly within the image space [44], this is seldom useful, as such representation is meaningless in the world space. On the contrary, most works assume that the image has already been projected into BEV, either before or after extracting the features. For this reason, we will mostly focus on the latter case, although most considerations are be valid also for the former.

To begin with, parametric models are characterized, as expected, by a finite number of parameters, which entirely determine their shape and position within the plane. They thus serve as a very compact

representation of a curve, which allows them to be stored very efficiently. However, a compact representation translates also in a reduced expressive power for the model, able to represent real curves only under certain constraints. What is then important to evaluate are the limitations imposed by such constraints.

The simplest class of parametric model can be found in polynomial curves. In particular, straight lines are often chosen when interested in a region of the road particularly close to the vehicle. Under this condition indeed, any line covers only a limited distance and does not have enough space to strongly deviate from a linear approximation [25]. At times, straight lines are also adopted for systems confined to highway scenarios [60]. This significantly simplifies the fitting problem and allows the usage of robust fitting methods, such as Hough transform, maintaining low computational costs [60].

Higher order polynomials are adopted instead for curved roads. In particular, second [61] and third [62] degree models are usually selected, the latter being more versatile, while the former less expressive and thus more robust to overfitting.

Thanks to their high flexibility, splines are also widely adopted. A spline is non other than a constrained piecewise polynomial, and it possesses a potentially large expressive power thanks to its capability to adapt parameters (and order) along its domain. This comes at the price of less robustness, issue to consider when tuning the number of break points and smoothness conditions. In this context, Huang et al. [22], and Kim [53] adopt cubic splines as their models. The former explain moreover how their choice allows them to follow, at times, even lines with the less common shapes. Wang et al. [63] instead exploits the versatility of B-splines to design an active contour model. Catmull-Rom splines are instead introduced by Wang et al. [43] and reconsidered later, with the work of Zhao et al. [36], for their smooth results.

Studying the shape of most highways, Gehrig et al. [64] discuss the possibility to introduce clothoids as a model choice. They argue indeed that large roads are designed enforcing slow changing curvatures, distinctive trait of clothoid functions. With this in mind, they

then propose a mathematical system to relate a clothoid line in natural parametrization to its projection on the image. Their derivation assumes the knowledge of the lane width and the full calibration of the camera. Expanding this model, McCall and Trivedi [65] integrate it with a tracking technique and adapt it for their line detection system VioLET.

Selected an appropriate parametric model, particular algorithms are usually considered for fitting it to the retrieved feature points. Among all, very frequent is the choice of least squares methods and variations. In this regard, González and Özgüner [52] use first the ordinary least square algorithm to fit the image with straight lines and obtain an overall estimate of the road orientation. Then, once all line markings have been detected, a weighted least square fitting optimizes the parameter of their polynomial model.

As these implementations are strongly sensitive to outliers, Labayrade et al. [66] adopt an M-estimator. This is nothing but a variation of the traditional weighted least squares formulation, where a lower weight is given to points very far from the model. In this way, outliers are left with a marginal influence on the estimate [29, Chapter 10].

When only straight lines are expected, the Hough transform [29, Chapter 10] is widely adopted, as in [67] and [42]. Kuk et al. [68] moreover develop a faster variation of the algorithm adding a few customized constraints. In Springrobot, their autonomous vehicle, Li et al. [69] apply instead the adaptive randomized Hough transform (ARTH), a probabilistic version of the algorithm they designed.

In the rest of the cases, solutions based on RANSAC [29, Chapter 10] are very common. To support this claim, Kim [53], Kim and Lee [26], Borkar et al. [58], and Huang et al. [22] all adopt it at some stage of their algorithm. Lipski et al. [70] instead employs RANSAC to determine width and orientation of the road at fixed distances from vehicle, estimates that are then used to model the shape of the road.

At last, it is important to mention also non-parametric frameworks. In this case, the representation of the line does not depend on only a finite set of parameters, but is instead largely less constrained and is depends only on the measured data. Given the openness of the

situation, no general representation or fitting algorithms is shared in this context. We can mention however the work of Ruyi et al. [24], who opt for a weak lane model in order to achieve complete adaptability to each road shape. In their system, the line model is indirectly represented by the state of a particle filter, updated at each step with the features just extracted.

Finally, we conclude this section with a warning. The above discussion was intentionally developed focusing only on single-lane systems. In this category, only the so-called ego-lane (i.e. the line where the vehicle is found) is modeled. We must point out however that it is clearly possible to extend this study in order to include several lines. Nevertheless, they will not be treated here, as they fall outside of the objectives of this work.

2.1.4 TRACKING

At the end of the pipeline, it is important to enforce the temporal consistency of the results obtained. In this fashion, the estimates on the position and shape of each line remain robust to occasional noise as well as missed measurements.

When parametric models are used, the most common associated techniques revolve around the concept of Kalman filtering. In particular, Wu et al. [71], Lim et al. [44], and Labayrade et al. [66], all rely on a standard Kalman filter (KF) for the tracking of their line. Moreover, we register that such basic filter is also adopted for lines expressed in polar coordinates [58] and it is successfully applied for the tracking of derived parameters such as line heading and curvature or vehicle lateral displacement from the centerline [65].

With nonlinear dynamics or more complex scenarios, the extended Kalman filter (EKF) is instead usually applied, as can be noticed in the approach of Tian et al. [72]. In the work of Zhao et al. [36] instead, their spline model is tracked thanks to an EKF. To this end, they consider a state vector composed of all the control points of their spline.

When the models are non-parametric instead, the most associated tracking method is the particle filter (PF), as used by Kim [53], Zhou et al. [73], and Ruyi et al. [24].

2.2 END-TO-END LEARNING-BASED SYSTEMS

As introduced at the beginning of this chapter, learning techniques experienced a tremendous rise in the last decade. As a direct consequence, what was before not possible in many fields, might be worth studying now.

This exact same fate applies to a specific branch of line detection systems: end-to-end deep-learning systems.

Although no system is yet ready for this challenge, we find it useful to mention some promising results.

To this end, we start from the work of Neven et al. [40]. They present a system that acquires an image and outputs a polynomial representation of all the road lines in the scene. For reference, its architecture is depicted in Figure 2.4. They begin by feeding each new frame into two separate CNNs. The smaller one, H-Net, is responsible for estimating the calibration parameters of the camera, in order to ultimately project the results on the world frame. At the same time, the second and bigger network, LaneNet, combines two tasks. On one hand, it detects which image pixels belong to a road marking, thus producing a prediction mask. On the other hand instead, it associates to each pixel some specific features, trained to allow for an easy separation, via clustering, of each different line on the road. In this way, each line is ultimately represented by a different binary mask. What they miss however to integrate in their end-to-end system, and have to cover with additional components, is the fitting of a line model. In their overall system, they do so by projecting each mask on the world reference frame and fitting its points with a third degree polynomial curve.

In like manner, John et al. [74] devise a complex learning system to simultaneously perform road and line detection. As they combine

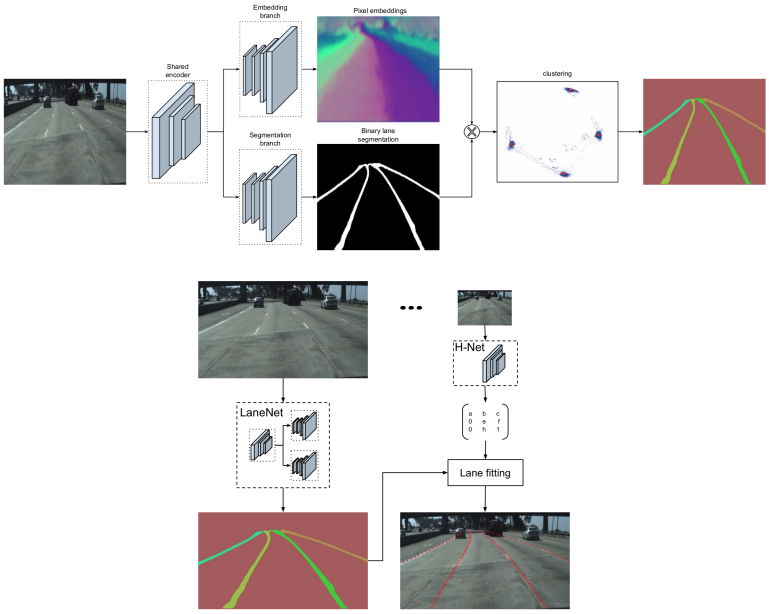


Figure 2.4: LaneNet (top) and the overall system in which it is presented (bottom). Adapted from [40].

different models, their system is not technically end-to-end. However, they show how the road and line representations could, in principle, be completely learned. They adopt a CNN to detect the road surface, and then pass the same extracted features to an extra trees regression model, trained to predict the location of each road line. With the same features moreover, they are able to also indicate a semantic interpretation for each of these line.

In conclusion, as an end-to-end system has yet to be developed, we must mention that some concerns have already arisen regarding the feasibility of its actual adoption in the field of autonomous driving. In fact, the biggest drawback of end-to-end approaches lies in their limited explainability. As a consequence, if their internal functioning cannot be easily interpreted, it becomes very hard to guarantee all safety standards are respected [75].

3

LANE FOLLOWING

The problem of lane following (also known as lane centering or lane keeping) is a particular instance of the vast field of lateral control. This consists, as the name suggests, in the control of the overall lateral behavior of a vehicle, operation that involves modulating its steering angle but also requires to properly understand its dynamics to guarantee its full stability and safe operation.

In this context, lane following consists in performing lateral control with a planned trajectory that maintains the vehicle within the boundaries of its ego-lane on the road.

With first attempts in the early 1990s, this task has a long history [76]. At that time of course, rudimentary sensors and restricted computational capabilities constrained their algorithms and limited their effectiveness. As years passed however, new hardware was developed and new algorithms proposed until, with time, the approaches adopted were consolidated. Thereafter, systems were expected to first detect the lines on the pavement, then plan a compatible trajectory, and finally act on the vehicle accordingly [77].

As our work focuses on these type of systems, in this chapter we present in details their operation and analyze their strengths.

In contrast with these approach however, recent new achievements led to a large increase of attention towards end-to-end solutions. In fact, while their inception is not new and dates back to the late 1980s [10], modern hardware and software technologies finally brought them to a competitive level. For this reason, although they remain out of the focus of our work and are still not mature enough, we briefly highlight their potential and discuss their drawbacks at end of this chapter.

3.1 TRADITIONAL LANE FOLLOWING

While the first road-following systems for automotive vehicles began to be researched in the late 1980s and exploited road boundaries as a guide [78], [79], [80], the research focus widened at the beginning of the next decade to also include lines. One of the first rudimentary lane following systems is described by Manigel and Leonhard [81]. In the meantime, vehicle steering control was studied on a theoretical level [82], [83].

Soon, the first complete work arrived, when in 1996 Pomerleau and Jochem [84] successfully developed their Rapidly Adapting Lateral Position Handler, RALPH. Through a series of detailed manipulations and engineering considerations, they were able to process the images acquired by a camera and produce the correct steering commands for their vehicle. Innovative was also their testing method, which is probably what gained them popularity. With their Navlab 5 test bed vehicle, they completed a U.S. coast-to-coast highway journey from Washington, D.C. to San Diego, in which Ralph had control of the vehicle for 98.1 percent of the distance traveled (i.e. 2,850 miles).

In the subsequent years, several new works were proposed to modify and improve the lateral control systems for this task [85].

In this way however, only half of the problem was always considered, and this lead to a progressive decoupling of the control models from the respective perception task. Line detection systems began to be

intensively studied, as reported in Chapter 3, but at the time they were only considered as mere self-standing tasks, used at the most for delivering lane departure warnings. Waiting for them to be perfected, new control systems were being tested using computer simulations [86] or in real but implausible scenarios [87].

As a consequence, when complete systems were again proposed a decade later, their architecture was naturally constituted of two separate components: first, a line detector refined to retrieve the position of the ego-lane and its centerline, and then, a control system able to maintain the required trajectory. This is the case of the vehicles presented by Liu et al. [61] and Hammarstrand et al. [88]. With the latter moreover, testing on real-world scenarios was finally introduced.

In these circumstances however, perception and control cannot benefit from their mutual cooperation and their development is instead constrained withintheir respective boundaries. We notice indeed that the lane detection is performed without knowledge of what measurements are important for the control and that this, in turn, does not provide additional information to the vision system.The result is thus a potential loss of performance on the overall lane following task and is an important aspect to consider for any further development.

Respecting the current state of the art however, in the next sections we will separately explore both components and their structure, treating first the perception task and then the planners and control methods employed. In between, a discussion on the lane models commonly adopted acts as a link between the two topics.

We conclude this section with a glance to some known commercial systems.

3.1.1 THE PERCEPTION PROBLEM

To sense the environment and perceive the road lines, different sensors can be used. Analogously to what happens for line detection systems (see Chapter 2)), cameras are the most popular, for their ability to detect the same features humans seek, together with their ease of usage.

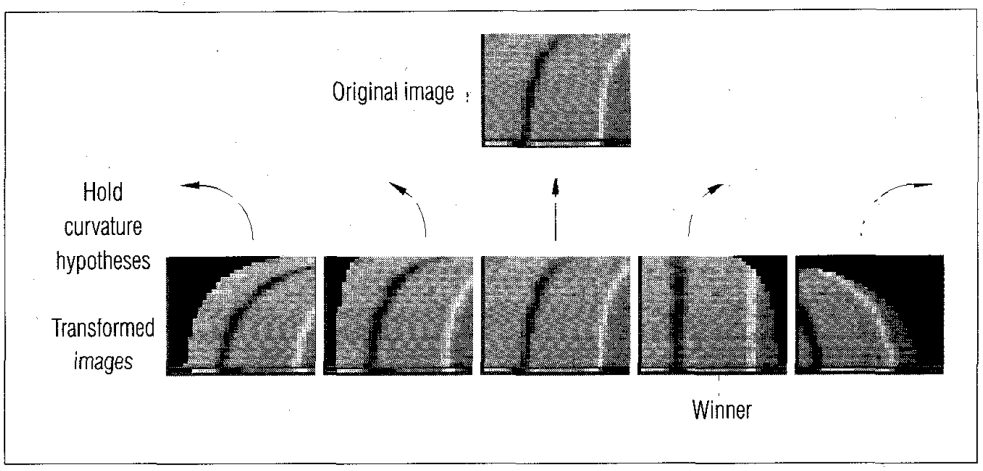


Figure 3.1: The nonlinear transformation applied in Ralph to determine the best steering action to perform. Adapted from [84, Figure 3].

Due mostly to the bad quality of their cameras, early methods relied predominantly on low level features of the recorded images and did not spend effort in modeling the observed lines. Instead, they exploited colors and geometric considerations in order to find the most appropriate steering command to execute.

In this regard, Pomerleau and Jochem [84] in their system Ralph (Rapidly adapting lateral position handler) begin by devising a nonlinear transformation to produce the same effects on the image of applying a particular steering command for a fraction of time (Figure 3.1). They then observes that, if the command applied is the optimal one, this operation should completely straighten the lines. To find the optimal command then, they just need to try for several possible steering actions, and select the one that better linearizes the detected lines.

With this clever technique, they could spare themselves the inconvenience of modeling each road configuration and estimating a trajectory from it. Besides, as they report about their coast-to-coast experience, the lack of strict dependence on the lines themselves allowed Ralph to continue along its path even when it encountered worn or missing markings. At those times indeed, they found it would still manage to rely on road boundaries and other similar features. In rainy weather

for example, only guidance for its operation were the tracks left on the wet pavement by the tires of preceding vehicles. Nevertheless, this extreme simplification has also its disadvantages. They don't expand on it, but scenarios such as line splits or wore out pavement could produce numerous false positives and lead to destructive consequences if no modeling of the actual road markings is performed. Moreover, the task becomes a lot harder when we leave the highways and approach urban scenarios.

These drawbacks, together with the advancements in sensing technologies, exposing more detectable features, led in the subsequent years to a departure from these techniques towards more stable algorithms based on the mathematical modeling of road markings [61].While in the meantime the studies on control models kept advancing, these techniques rapidly converged into the vast world of line detection methods (see Chapter 2), which is where we find them today.

Only in later times, additional sensors have been experimented to support single cameras. This is the case of radar, introduced in the work of Hammarstrand et al. [88] to measure the heading of other vehicles and sense stationary objects, such as guardrails, in order to obtain clues on the geometry of the road. With this consideration, they are able to correctly estimate the shape of several highway roads, including tracts with a strong curvature. Thanks to these measurements moreover, when they find themselves on a road bend, they can determine with better precision whether a vehicle perceived in front of them is actually on their trajectory or if, instead, it is just traveling on a parallel lane.

3.1.2 LINE MODELS

Assuming the lines are correctly detected, attention should be devoted to the representation used to describe them and ultimately communicate them to the control system. In this section, we analyze the implications of each different representation. Before doing so however, we introduce the topic with a theoretical review of the mathematical properties of plane curves.

3.1.2.1 *Background on plane curves*

Considering the Euclidean space $\mathbb{R}^2$, and given an interval $I \in \mathbb{R}$, a plane curve γ can be represented as a continuous vector function

$$\gamma : \quad \mathbf{r} : I \to \mathbb{R}^2 \tag{3.1}$$

which corresponds to

$$\gamma : (x, y) = (f_x(t), f_y(t)), \quad t \in I \tag{3.2}$$

This generic formulation is known as *parametric representation*, as it relies on the parameter t. If we consider this to be a description of time, Equation 3.1 represents the trajectory of a particle moving along the curve.

The support of the curve (Γ) is then defined as the image of $\mathbf{r}$, i.e. the set of points $(x, y) \in \mathbb{R}$ crossed by the curve.

It is obvious from its definitions that infinite functions $\mathbf{r}$ share the same support Γ. We say that each of those functions is a different *parametrization* of the same curve and that a map $\varphi : I \to \tilde{I}$ such that

$$\tilde{\mathbf{r}}(\varphi(t)) = \mathbf{r}(t) \ \forall t \in I \tag{3.3}$$

realizes a *re-parametrization* of γ in $\tilde{\gamma}$.

For our purposes, only the support Γ of a curve is important, regardless of its specific parametrization. For this reason, several alternative representation can be exploited.

At first, we can consider the so-called *implicit representation*,

$$F(x, y) = 0 \tag{3.4}$$

However, this type of formulas are very hard to manipulate, as x and y are potentially non-separable. For this reasons, the *explicit representation* is instead often used:

$$y = f(x) \tag{3.5}$$

A model in this form is very easy to employ thanks to the immediate correspondence between the x and y variables. Moreover, their separation simplifies the optimization process for interpolation and

fitting tasks. Nevertheless, not any line can be represented in this way. Indeed, the model is itself a function, and as such no two y's can be mapped to the same x. This implies that the line cannot wrap on itself and must instead always move forward in the x-direction.

Conversely, Going back to the parametric framework, a particular parametrization, said *natural parametrization* (or also arc-length parametrization), describes exactly the support of a curve with no dependence from the arbitrariness of t. To develop its derivation, we need to first define the length (or arc-length) of a curve ℓ, which is

$$\ell = \int_I \|\mathbf{r}'(\tau)\| d\tau \tag{3.6}$$

where $\mathbf{r}'$ represents the first derivative of the trajectory $\mathbf{r}$. If we then define also the cumulative length of a curve

$$s(t) = \int_{t_0}^{t} \|\mathbf{r}'(\tau)\| d\tau \,, \qquad t_0 = \inf(I) \tag{3.7}$$

and this integral exists, we can then use $s(t)$ as a map to re-parametrize our original curve, obtaining

$$\begin{aligned} \gamma_s : \quad & \mathbf{r_s} : \mathbb{R} \to \mathbb{R}^2 \\ & \mathbf{r_s}(s) = \mathbf{r}(s(t)) \,, \; t \in I \end{aligned} \tag{3.8}$$

The cumulative arc-length s is also known as the *natural parameter*.

All these representations are strictly dependent on the position and orientation of the coordinate frame used to define them. It is desirable at times to model the curve in such a way to be independent of any coordinate transformation. Since s is completely unrelated to the reference frame and is only dependent on a property of the curve itself, it is possible to exploit it to introduce yet another couple of useful representations. The *Whewell representation* (Figure 3.2) indeed relates the arc-length s of a curve to its tangential angle ϑ:

$$\vartheta = f(s) \tag{3.9}$$

while the *Cesàro representation* relates the arc-length s to its curvature κ:

$$\kappa = f(s) \tag{3.10}$$

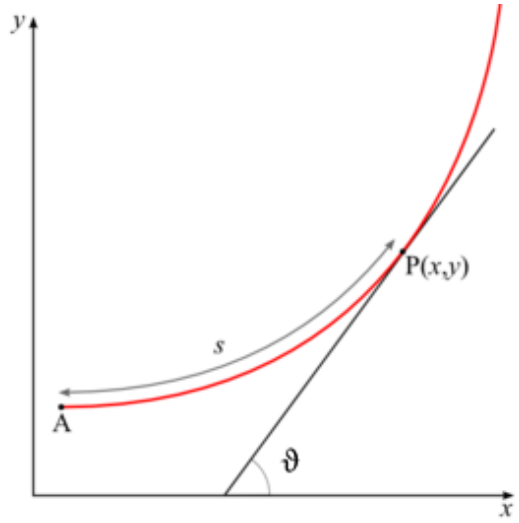


Figure 3.2: Reference frame for the Whewell representation. Adapted from [89].

Not mentioned before, the curvature can be shown to be

$$\kappa(s) = \|\mathbf{r}''(s)\| = \frac{d\vartheta}{ds} \tag{3.11}$$

Because of their dependence only on properties of the curve, these representations are said to be given as *intrinsic equations*. The correspondence of these latter frameworks with explicit $x - y$ representation can be achieved integrating according to the well-known Fresnel integrals:

$$\begin{aligned} x &= \int \cos(\vartheta) ds \\ y &= \int \sin(\vartheta) ds \end{aligned} \tag{3.12}$$

3.1.2.2 *Representations for road line modeling*

Looking first at the representations proposed in works on perception, explicit models cover almost the totality of the cases. This is due to their strong dependence on line detection systems (see Section 3.1.1), which are almost completely represented in this fashion (see Section 2.1.3). As already reported, common models here are low-order polynomials

$$y = P^n(x) = a_n x^n + \cdots + a_1 x + a_0, \quad n \in \{1, 2, 3\} \tag{3.13}$$

and cubic splines

$$y = s_i(x) \qquad \text{with } x_i \leqslant x < x_{i+1}$$

s.t.

$$\begin{aligned} & s_i(x_{i+1}) = s_{i+1}(x_{i+1}) \\ & s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}) \\ & (*) \end{aligned} \tag{3.14}$$

with

$s_0, \ldots, s_k$ cubic polynomials,

$x_0, \ldots, x_{k+1}$ knots of the spline

$(*)$: *additional smoothness constraints*

For what concerns the control task instead, explicit models are seldom adopted. As Hammarstrand et al. [88] argue indeed, these models are not appropriates for sensor fusion and road modeling. On the contrary, they point out how parametric models and especially intrinsic Whewell representations are more suitable for this task. While most other works do not focus directly on line detection, they generally assume that either the trajectory to be followed is represented with Whewell or Cesàro equations [90], [88], or they are provided with heading, curvature and other parameters easily retrievable from such representations [91].

To this regard, common intrinsic models are usually constructed on polynomials, as they are easy to manipulate and at the same time have often enough expressive power. In fact, in a Whewell frame, low order polynomials respectively represent the following curves:

ORDER 0: $\vartheta(s) = \vartheta_0$ straight line (orientation ϑ_0)

ORDER 1: $\vartheta(s) = \frac{1}{R} \cdot s + \vartheta_0$ circle (radius R)

ORDER 2: $\vartheta(s) = \frac{1}{2}\frac{d\kappa}{ds} \cdot s^2 + \frac{1}{R_0} \cdot s + \vartheta_0$ clothoid or Euler spiral

Even when explicit $x - y$ models are adopted in control, as in the work of Liu et al. [61], no particular advantage is presented, and instead the computation of curvature, lateral displacement and road slope require more effort.

At the same time, the major drawback encountered when working with intrinsic parametrizations is evident when they need to be re-projected on a Cartesian space, for example to get integrated with other information. In this case, it is necessary to integrate the curve over its whole length, operation performed numerically and thus fairly expensive. Moreover, it is not possible to retrieve the position of a line at a given x or y coordinate without solving a nonlinear equation, as no closed form solution exist. This applies also when trying to intersect it with other geometric elements in the Cartesian space.

To conclude then, we have seen that explicit representations register more preferences from most perception systems, while intrinsic models are more employed in control. This fact constitutes already a problem in itself, as it undermines the coupling between the two components of the system. Besides, both representations present their advantages and disadvantages, dependent on the computations to be performed and the scenarios observed. Therefore, while this discrepancy is not beneficial for the overall system, it is not clear which option should be preferred, and probably no unique solution exists. Nonetheless, during the development of both components, more cooperation should be achieved, analyzing and exploiting the common needs in each specific occasion.

3.1.3 PLANNERS AND CONTROL SYSTEMS

Perceived the environment and bearing in mind the considerations made on lines representations, what is left for the system is to reason on the information acquired, decide its following action and perform it. These tasks are usually assigned to a macro-component called *planner* (although other common names are *decision system* or *control system*).

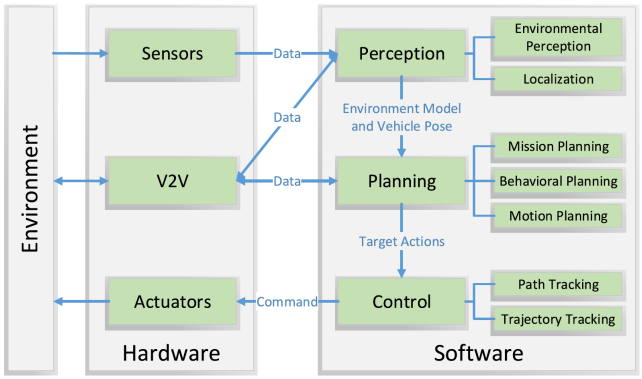


Figure 3.3: General architecture of an autonomous vehicle reported in [92]. Adapted from [92].

Its purposes are various. On a global scale, it can select from a map the best path to reach a destination and update it in time as new information is received. On a local scale, it is responsible for determining the best trajectory on the road to remain inside the appropriate lane, respect the traffic laws and avoid collisions on the way, be they with other vehicles or with unexpected obstacles.

At times, its responsibilities are split in two according to the time frame within which they are operated. In this fashion, we can alternatively consider only the higher level component as the proper *planner* and refer to the local one as the *controller* [92]. This latter architecture is depicted in Figure 3.3.

In general, Behere and Törngren [93] accurately present this topic and propose a detailed description of their functional architecture. As also depicted in Figure 3.4, they identify three macro-components, which they call perception, decision & control, and vehicle platform manipulation. Notably, they highlight in their work the changes of representation required form one stage to the next, prescribing all communications to be referred to the world coordinate frame at decision time, and referred then to the ego-vehicle for its actuation. It is interesting to notice moreover that they decouple the physical details of the actuation system from the decision stage with a dedicated abstraction layer.

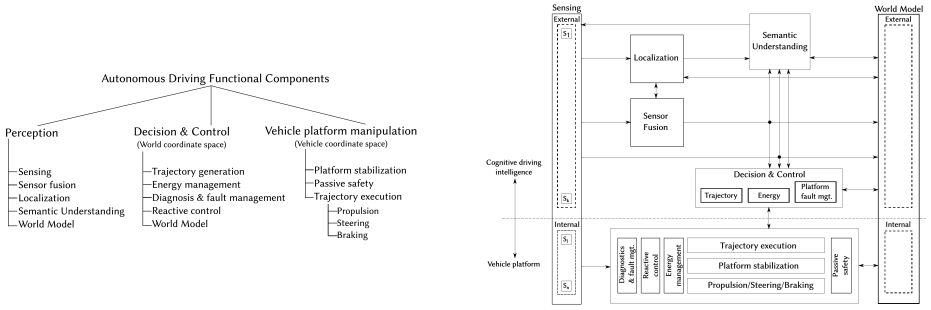


Figure 3.4: The functional architecture described by Behere and Törngren [93]. In particular, logic organization of their functional components (left) and final architecture (right). Adapted from [93].

Entering into the details of lane following, the responsibilities of the planner are more limited and have been studied in several works. Although some of these methods are now outdated, Chaib et al. [94] present a study on the performances of H_∞ control, self-tuning regulators, PIDs and fuzzy systems. From their experiments, they obtain comparable results, with a slight predominance of self-tuning controllers over the others. Regarding the mentioned fuzzy systems, they can be found in several solutions for lateral control, as the work of Pérez et al. [95], and Wang et al. [91]. In a similar context, Sotelo [96] presents his design of non-linear kinematics and lateral control law for their vision-based system.

Li et al. [97], on the contrary, develop a large hierarchical motion planning system and validate it to perform as lane follower, avoiding obstacles at need. In their perception algorithm, solely based on LIDARs, they retrieve each lateral line and, after some processing, determine the optimal path to follow. During their analysis, each candidate path is sampled and stored as a set of positions x and y, headings ϑ and local curvatures κ. Finally, Arrigoni et al. [92] propose a motion planner based on model predictive control. They devote particular attention to the optimization problem, adopting an algorithm based on accelerated particle swarm optimization (APSO). In their work, the road shape is described through third order polynomials in a curvilinear reference frame $s - y$ (Figure 3.5), where y is the lateral offset of the vehicle from the lane center.

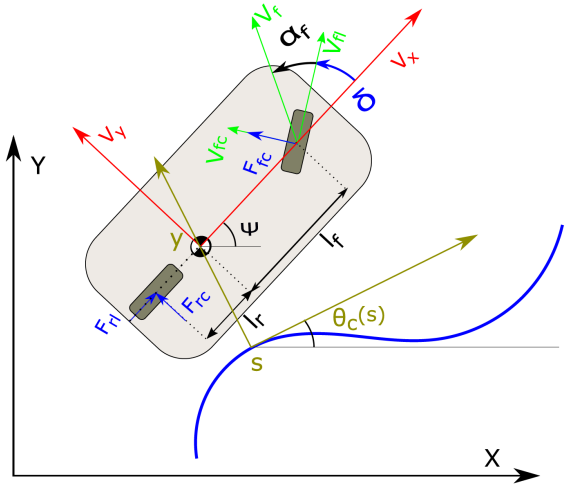


Figure 3.5: Curvilinear reference frame based on s and y used in [92]. Adapted from [92].

3.2 END-TO-END LANE FOLLOWING

In contrast with what we have seen for traditional systems, end-to-end lane following modules do not present a division in sub-components. Instead, a unique module performs the whole task. In particular, a complex neural network usually receives the information from some sensors, almost always including a camera, and is then able to determine the correct steering action to perform.

This technique was first adopted in the pioneering work of Pomerleau [10] on road following. He feeds the measurements of a monocular vision system and a LIDAR to a traditional neural network, training it to determine the best direction to take to remain on the road. Being an end-to-end system, the output of his network is a vector of 45 possible direction to be taken and thus directly indicates the steering command to execute.

Coming to modern days, the learning elements now adopted are convolutional neural networks and their architecture is usually more complex. In this regard, Chen and Huang [98] present a CNN-based system able to determine the steering angle needed to remain on the road. They do so using the comma.ai driving dataset [99], and

evaluating their software on prerecorded videos they achieve a frame-wise mean error of 3.26 degrees. However, as they do not test their system inside the loop–letting it perform the chosen steering action–it is hard to evaluate the actual performance of their method.

With a more complete system, Bojarski et al. [100] present their advancements on DAVE-2, a deep end-to-end module for lateral control developed by NVIDIA and inspired by the DARPA Autonomous Vehicle (DAVE, a terminated project developed by the U.S. Department of Defense, [101]). To train their model they record the image seen by a human driver and his steering commands. The scene is also captured using two additional, front-facing cameras mounted on each side of the windshield in order to perform data augmentation. After satisfactory results in simulation, they test their overall system on the road, driving autonomously 98% of the time in relatively brief drives. Although their system is not perfected, they argue that this approach, compared to traditional techniques, will eventually lead to better performances, as the learning machine can automatically assimilate an internal representation of the features most needed for the task.

First to combine lateral and longitudinal control is the work of Yu et al. [102]. To begin, they collect their training data using the Baidu street view platform and share them as the Baidu Driving Dataset (BDD).They then propose for the longitudinal control of their system a stacked convolutional LSTM able to directly generate the acceleration commands to be executed. Their lateral control is instead performed by large CNN and operates based on the estimated curvature instead of the steering angle. With proper training, this component is reported to converge to an acceptable mean squared error (MSE) after relatively few epochs, but it is hard to evaluate its effectiveness as it is not clear how inaccurate estimations of the curvature are propagated into the actual heading of the vehicle.

In conclusion, we deem important to point out that, as for end-to-end line detection systems (Section 2.2), the same safety concerns have been raised, with the aggravating difference that lane following systems are not only required to perceive the environment correctly, but also to act on it, with consequences potentially more dangerous.

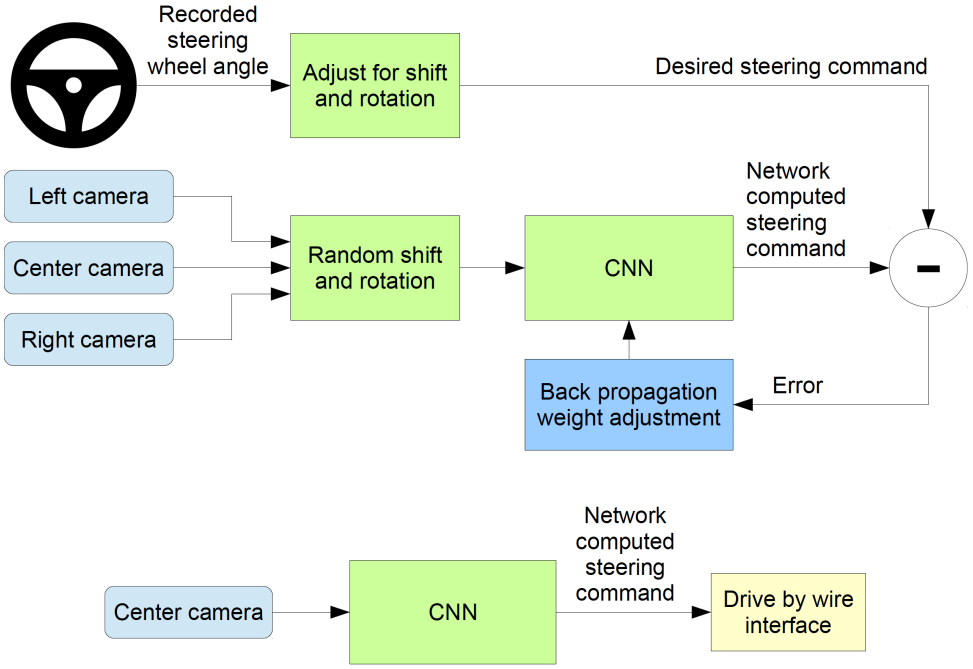


Figure 3.6: Training (top) and testing (bottom) configuration of the end-to-end system used in DAVE-2. Adapted from [100].

4

PROPOSED SYSTEM

In the presented analysis of the state of the art on the overall lane following problem, we outlined the difficulties of current systems and highlighted a lack of overall agreement in their architecture. In particular, we registered an affirmed misalignment between works in perception and control, which not only leads to inefficiencies, but also constitutes a possible limit to the potential of both components. As we pointed out, for instances, the representations commonly adopted in perception are often derived from works on line detection and are not consistent with the needs of the control counterpart. Moreover, this marked separation complicates the real-world testing of the complete algorithms, operation already hard to perform for articulated automotive systems.

This condition is somehow more relaxed for end-to-end solutions, not requiring explicit representations of the detected lines, but the safety concerns raised in their regards and the difficulties in certifying such techniques discourages us to pursue that path.

Given these observations, we propose the design and development of a novel perception system for the task of line following inspired by the requirements of its control counterpart. To this end, the system should be able not only to detect the ego-lane markings,but also to estimate shape and position of its centerline, and determine heading and lateral displacement of the vehicle with respect to it. These parameters are of fundamental importance from the point of view of control systems, as they allow for a correction of the trajectory in order to maintain a safely centered position on the ego-lane. The result of our work is then a system maintaining an internal representation analogous to the one used by the controller, promoting, and not impeding, the exchange of information between the two modules.

Given also the lack of testing in real-world scenarios for most lane following systems, we additionally require our work to be evaluated on real-world data, avoiding potentially misleading simulations.

With this in mind then, in the rest of this chapter we describe our approach, providing first an overall description of the architecture we propose, and entering, then, into the details of each component.

4.1 SYSTEM ARCHITECTURE

The main objective of our system is to provide the lateral control unit with the necessary information to safely act on the vehicle and maintain it within its ego-lane.

As already mentioned, this requires the knowledge of a few key aspects. At first, it is necessary to retrieve the shape of the road and its relative position respect to the vehicle, with specific attention to the line markings delimiting its ego-lane.With this information then, what is missing for the control is a precise estimate of the relative orientation and displacement of the vehicle with respect to its optimal trajectory, usually placed at the ego-lane center.

For this reason, the overall architecture of our system begins with a line detection stage. This is intended to identify the line markings in successive frames and additionally determine their position within

a vehicle-centered world reference frame. This is fundamental for the successive stage, dealing mostly with the estimation of the parameters required by the controller. In particular, given a representation of the two lateral lines, this second stage is appointed to estimate the position and shape of the centerline and track it over time. With this measurement, it can potentially determine how much the vehicle is distant and unaligned from the optimal trajectory, passing this information to the control section to correct its course. At this stage however, the noise associated with all the preceding estimates will have accumulated, and thus a filtering technique will be necessary to correct the final estimation and provide the controller with accurate information.

In pursue of havingour system work requiring only minimal information, we design it to rely just on monocular vision, being the most common choice in line detection as well as lane following systems. Cameras, indeed, have the ability to distinguish colors and other important features exploited by humans when driving. However, they introduce some difficulties when dealing with distance measurements, issues we need to cope with. Finally, although this could potentially be enough, we also assume odometry measurements are also available from the vehicle and can be used to filter our estimates in time.

4.2 LINE DETECTION

Assuming the vehicle is on the road, we begin our pipeline detecting the lines delimiting its ego-lane. To this end, we start from the traditional framework for line detection presented in Chapter 2 and customize it for two, concurrent reasons.

In order to match the best performing algorithms in the literature, we first require our system to employ up-to-date techniques. This is especially true for the low level operations, such as the extraction of features, often dependent on the physical capabilities of the machines adopted. For this reason, our features are extracted directly from raw images, and instead of maintaining an unprofitable preprocessing

stage, we move the cleaning and preparation of the extracted data afterwards, to what we define a postprocessing stage. Of course, the suppression of the preprocessing component is only made possible by the absence of substantial noise in the acquired images, as we assume the utilized cameras are in line with the current standards and feature up-to-date noise-correction apparatuses.

On the other hand, as the ultimate purpose of our line detection system is not to represent the lateral lines but to provide the necessary information for the subsequent centerline estimation stage, the internal representation used need not be the same of common line detection works. In particular, the models usually employed are easy to fit and clear to interpret by humans as output of this stage, but do not explicitly carry the right information on the position of the centerline itself. For this reason, in spite of being externally compliant with the usual system architecture, our model fitting stage exposes essential differences.

4.2.1 DATA ACQUISITION AND CONVENTIONS

The first, yet obvious step within the system, the necessary data need to be acquired. To this end, we expect the acquisition process to be performed with a vehicle, equipped with the necessary sensors. In particular, we expect an RGB camera to be mounted on top of it, facing forward. Additional sensor, such as Lidar, radar, global position systems (GPS) and inertial navigation systems (INS) could be installed, although will not be used for our work. Encoders and analogous instrumentation could be used for the generation of odometry measurements.

For the entire scope of our work, we represent the vehicle following the coordinate systems indicated by the ISO 8855 [103], with the origin on the ground below the center of mass of the vehicle, and axis x,y and z pointing respectively forward, left and up. The camera reference frame has its origin set in the optical center of the camera and the same orientation as the vehicle's frame. Finally, for the image plane, the origin of the system is set in the upper-left corner, with the u axis

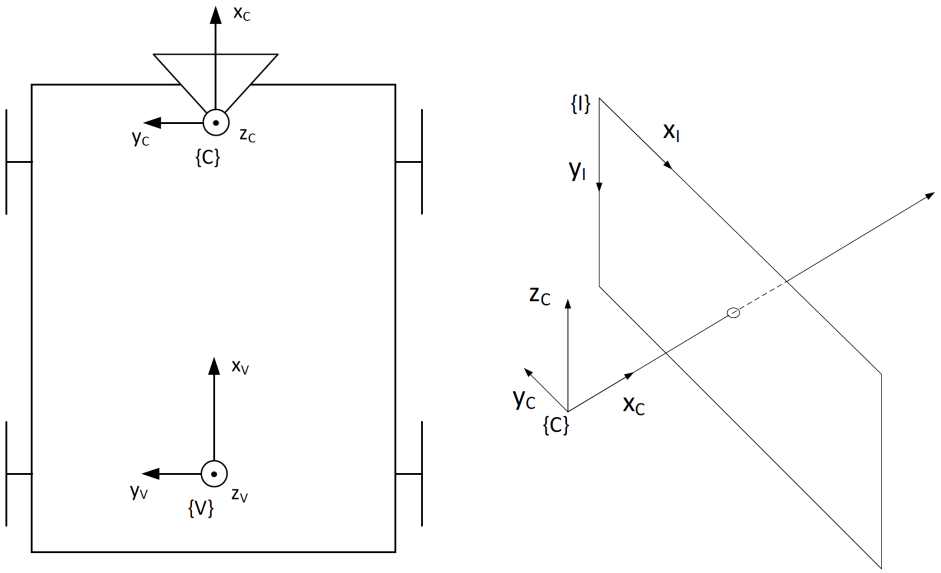


Figure 4.1: Coordinate systems adopted for vehicle {V}, camera {C} and image {I} reference frames. Adapted from [104]

pointing right and v pointing down. For clarity, these reference frames are depicted in Figure 4.1.

4.2.2 FEATURE EXTRACTION

In any computer vision system, the main purpose of feature extraction techniques is to highlight the characteristics of the image most relevant for the respective task to perform. In line detection, this ultimately translates into the identification of the image points most likely to represent a line marking. In the rest of this document, we refer to these points as *feature points*.

While older techniques employed image filters and often complicated computer vision algorithms to better select these points, modern approaches usually resort to machine learning methods. This is thanks to their capability to automatically learn what characteristics to look for in the image and, if well-trained, to do so regardless of illumination changes, adverse weather and clutter in the image.

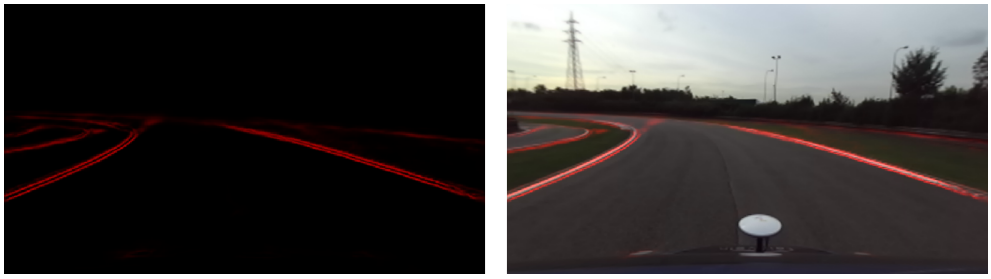


Figure 4.2: Raw predictions of our CNN on a road image. For clarity, the predictions are reported in isolation (left) and overlaying on the image itself (right). Brighter pixels are associated to higher predicted values, thus higher confidence of the network in the presence of a line marking.

For this reason, we exploit a convolutional neural network (CNN) specifically trained for this task and already available to us[1]. Such network is based on U-Net [105], an architecture originally developed for biomedical applications and later become popular for any segmentation task. Taking in input a front-facing image of the road recorded from any vehicle, its output is a single-channel integer image representing the expected position of each road marking on the pavement. Although not precise in a statistical sense, each pixel on this image can be interpreted as a weight, from 0 to 255, indicating how much confident the network is of the presence of a road marking in such position (Figure 4.2). Usually, this values are binarized through thresholding, in order to obtain a simple image mask, but we leave this task to our postprocessing stage for reasons that will then become clear.

The architecture of our network (Figure 4.3) follows the structure of the original U-Net, with the only difference that the input is downscaled to 512x256 to improve the network speed on low-power devices. With this configuration indeed, the network allows predictions up to almost 100 Hz on a NVIDIA Jetson Xavier, a known embedded computing board used in its development and testing.

The network is trained from scratch on the Berkeley DeepDrive Dataset [106], an extensive dataset for autonomous driving widely used for line markings detection as well as for the detection of road

1 The network was developed and trained by Simone Mentasti, Department of Electronics Information and Bioengineering, Politecnico di MIlano.

object and driveable area and for the task of instance segmentation. It provides more than 1100 hours of driving featuring different weather and illumination conditions as well as driving scenario, making it perfect to achieve the robustness required for autonomous driving task.

In this regard, we point out that since the network was trained on raw road images, it intrinsically learned to cope with most artifact, such as noise and clutter on the image, changes in illumination conditions and even obstruction. It is for this reason that we can safely avoid a preprocessing stage in our pipeline.

4.2.3 FEATURE POSTPROCESSING

The features extracted at the previous stage are still raw and could not be used by our model fitting module unless two particular operations are first performed.

On one hand, these features, which are now just an image of weights, need to be interpreted and converted into a representation that clearly indicates which points are to be considered as part of a line marking. To do so, usually a thresholding technique is adopted, transforming the image into a binary mask. Implicitly, this operation assumes that pixels with values higher than the fixed threshold are likely enough to actually represent a line marking and can thus be taken as feature points.

At the same time, on the other hand, the scene we are currently processing is represented only on a front-view image. However this is very impractical to be used for line detection. What the literature commonly adopts is instead the bird's eye view representation (BEV). The projection of our front-view image and the relative feature points are depicted in Figure 4.4 and Figure 4.5. This is achieved through a projective transformation that rectifies the ground plane of the image exploiting the camera parameters. For this operation we assume then, of course, that the camera has been calibrated.

Before we develop in details this computation, it is important to notice that the order of application of these operations is not invariant

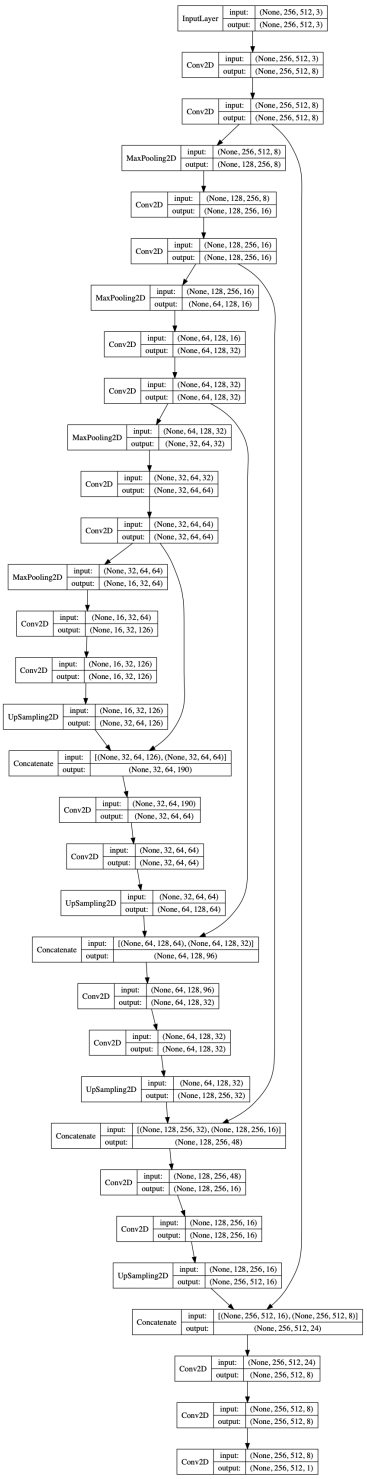


Figure 4.3: Overall architecture of the CNN used in our pipeline for feature extraction.

Figure 4.4: A front-view image acquired with our vehicle and the corresponding BEV representation.

Figure 4.5: Projection of the CNN predictions into BEV (right), together with the original predictions (left) for comparison.

and significantly changes the result. The only way to avoid losing information in the process is to first project the image weights into the BEV plane, and to perform the thresholding once there, as shown in Figure 4.6. Indeed, the computation of the BEV requires an inverse warping, operation that entails interpolation. If we were then to perform the thresholding operation before the image warping, the overall result would not be binary, as the interpolation would generate many values in the whole range allowed. As a consequence, we would need to apply an additional threshold, losing information content and computational time.

At last, the resulting mask in BEV, which we will define as M_{bev}, provides already a good representation of the line markings in the scene. However, we notice two minor issues. On one hand, we could register several small false detections throughout the image, accentuated in more challenging scenarios. On the other hand instead, the contour of

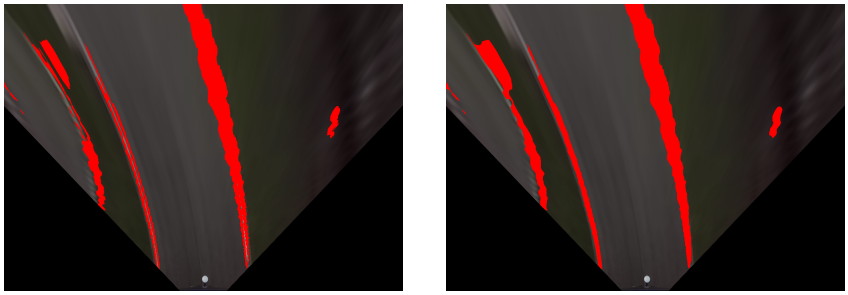


Figure 4.6: Results of thresholding (left) and subsequent morphological post-processing (right) on the features predicted by the network.

the detected regions appears often undulating and clearly not smooth, issue that could slightly affect the rest of the algorithm. Therefore, to improve even more the performances of the subsequent stages, we perform an additional step and apply a chain of two morphological closing operations to the M_{bev}, obtaining M'_{bev} (Figure 4.6). This final mask and the feature points it contains will finally be sent to the next module along our pipeline.

4.2.3.1 *BEV computation*

We spend now a moment to properly define the computation of the BEV image, since it often recurs in our work as in line detection systems in general, and it is thus worthy of a proper explanation.

We begin with a few, basic definitions. We define a point in the image and in the world as, respectively,

$$\mathbf{x}_c = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \in \mathbb{P}^2, \quad \mathbf{X} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \in \mathbb{P}^3,$$

with W and w homogeneous coordinates in the respective spaces.

Additionally, we consider the calibration matrix of the camera as

$$P = \begin{pmatrix} | & | & | & | \\ \mathbf{P}_X & \mathbf{P}_Y & \mathbf{P}_Z & \mathbf{P}_W \\ | & | & | & | \end{pmatrix},$$

and we assume this is known since the camera is calibrated. We remind the reader that, by the definition of camera matrix,

$$\mathbf{x}_c = P\mathbf{X} \tag{4.1}$$

The idea now is to design a projective transformation H_{bev} that maps every point in the current image plane to a point in a new image plane, which will become the BEV plane. In this plane, all ground points are rectified, meaning that the perspective distortion to the shape of every object on the ground, including the line markings, will be removed. This can only be obtained under the strong assumption that every point in the original image belongs to the ground plane $\pi_G : Z = 0$. Of course, this does not hold for every pixel, as many external regions could be inadvertedly included in the scope of the transformation. For the objects in those regions, the resulting shape will be then even more distorted. This does not constitute an issue however, as in the subsequent stages of our pipeline we will only consider the pixels coming from line markings, and therefore this imprecise pixels will implicitly be excluded by the processing.

We notice that, for a point on this ground plane π_G, the following holds

$$\begin{aligned} \mathbf{x}_c &= \begin{pmatrix} | & | & | & | \\ \mathbf{P}_X & \mathbf{P}_Y & \mathbf{P}_Z & \mathbf{P}_W \\ | & | & | & | \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ W \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} | & | & | \\ \mathbf{P}_X & \mathbf{P}_Y & \mathbf{P}_W \\ | & | & | \end{pmatrix}}_{P_{\pi_G}} \underbrace{\begin{pmatrix} X \\ Y \\ W \end{pmatrix}}_{\mathbf{x}_{\pi_G}} \end{aligned} \tag{4.2}$$

To simplify the computation, we additionally define two parameters, controlling respectively the portion of ground plane to be included in the BEV image, ROI_w, and the resolution of the resulting image, R_{bev}:

$$\mathbf{ROI}_w = \begin{pmatrix} \mathrm{ROI}_{w_{x_{min}}} \\ \mathrm{ROI}_{w_{x_{max}}} \\ \mathrm{ROI}_{w_{y_{min}}} \\ \mathrm{ROI}_{w_{y_{max}}} \end{pmatrix}, \quad \mathbf{R}_{bev} = \begin{pmatrix} \mathrm{R}_{bev_w} \\ \mathrm{R}_{bev_h} \end{pmatrix}$$

Given all this, the projective transformation that maps image points $\mathbf{x}_c$ into the corresponding BEV points $\mathbf{x}_{bev}$ can be obtained as follows.

$$\mathrm{P}_{\pi_G}^{w \leftarrow c} = \mathrm{P}_{\pi_G}^{-1} \tag{4.3}$$

$$\mathrm{H}_{coords} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.4}$$

$$\begin{aligned} s_x &= \frac{\mathrm{R}_{bev_w}}{\mathrm{ROI}_{w_{y_{max}}} - \mathrm{ROI}_{w_{y_{min}}}} \\ s_y &= \frac{\mathrm{R}_{bev_h}}{\mathrm{ROI}_{w_{x_{max}}} - \mathrm{ROI}_{w_{x_{min}}}} \end{aligned} \tag{4.5}$$

$$t_x = s_x \cdot \mathrm{ROI}_{w_{y_{max}}}$$

$$t_y = s_y \cdot \mathrm{ROI}_{w_{x_{max}}}$$

$$\mathrm{H}_{view} = \begin{pmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

$$\mathrm{P}_{\pi_G}^{bev \leftarrow w} = \mathrm{H}_{coords}\ \mathrm{H}_{view} \tag{4.7}$$

$$\mathrm{H}_{bev} = \mathrm{P}_{\pi_G}^{bev \leftarrow w}\ \mathrm{P}_{\pi_G}^{w \leftarrow c} \tag{4.8}$$

such that

$$\mathbf{x}_{bev} = \mathrm{H}_{bev}\ \mathbf{x}_c \tag{4.9}$$

4.2.4 MODEL FITTING

With the image mask received from the previous stage, we can now work on representing the feature points it contains with an appropriate line model. Here, two problems arise.

First, we soon notice that, in general, we expect to find and model two lines from a single image, corresponding to the left and right boundaries of the ego-lane. However, looking at the feature mask, it contains no information on which point is associated to which line. This becomes even more complicated when multiple lanes are present in the scene, as for example on a large highway. It is thus fundamental to devise a method to preemptively distinguish between different line markings in the same image.

As a second problem, an appropriate line model has to be selected to properly model the detected markings. As we analyze this issue later, we must recall from the introduction to this chapter that, for our purposes, what is most valuable is the support given to the centerline estimation, and not to the measurement of the specific position of each line. Finally, a model fitting procedure has also to be applied. However, as its choice could potentially depend on the line model adopted, we discuss both problems contextually.

4.2.4.1 *Feature points selection*

The image mask obtained from the previous stage contains a large number of feature points, and at the same time does not provide any information on how these can be grouped together into different lines. Moreover, noise and false detections could be also present.

To put the situation in order,the purpose of this component is then to select a reduced and highly reliable group of points as representative of each line marking, doing so for each line of interest on the road. In our case, this translates into the left and right boundaries of the ego-lane.

This procedure has several requirements. On one hand, the most important task to perform is the identification of which feature points

are associated to any single line, separating them accordingly.However, in more complex scenarios this is not enough. Indeed, some times it is also necessary to separate some feature points even within the same line. We can think for example at the case of a split in the road, where only the inner section of the line has to be considered. On the contrary, at times it is also necessary to aggregate points that seem to belong to different lines. This is the case, instead, of dashed road lines, where each segment needs to be manually connected to the others in systems where this process is not automatically executed at the feature extraction step. them.

Even more so, selecting only a limited a number of reliable points for each line of interest instead of using a large unprocessed group, has the side advantage to speed up the model fitting and potentially improve the estimation.

For all these reasons, we proceed with the design of a specific algorithm. In doing so, a particular yet simple consideration helps us remarkably in our job. To properly appreciate it, we first need to point out a few facts. First of them is that, in our overall system, we are not trying to detect *any* line in the scene, but we are instead looking only for road lines, usually lying on the pavement and respecting certain constraints in terms of shape and position. Moreover, we are observing the scene from a very specific and known view-point, which is the front-view of a vehicle. We can alsoassume that the vehicle itself is on the road and, most of the times, within a lane. All this information can then certainly be exploited to our advantage.

Indeed, in this context,we soon notice that the line markings belonging to the ego-lane can be easily individuated in the region of the road closer to the vehicle. From there, we can see them extending further on the pavement, following the shape of the roadway.

If we could then always detect them in the lower end of the image, we could then easily retrace them upwards and be sure to be always following the right marking, independently of its shape. Notice also that this is intrinsically their function, to visually indicate us how the road developsfrom our current position. Furthermore, to maintain

the size of the lines constant during our search, this can better be performed on the BEV image.

This is exactly the idea behind our algorithm, which is able to search for a line in the lower end of the image and follow it upwards along its shape thanks to a mechanism of moving windows. For this reason, we refer to it as the Window-based line following (WLF) algorithm.

As the algorithm starts, two windows w_l and w_r are initialized at the bottom of the BEV, respectively to the left and right of the ego-vehicle. Proceeding independently, their purpose is to follow the respective line of feature points on the BEV plane without any constraints on its expected shape. At each position visited by a window, an analysis of the enclosed image patch is conducted and the centroid of the largest connected component found is selected. This point will be saved and subsequently passed to the next stage for the actual fitting, as the most representative point of the windowed region. For the analysis of the connected components we refer to the algorithm proposed by Grana et al. [107]. In addition, to ensure that the line is properly followed, the orientation and size of the window are adaptively changed, rotating it as the line bends and shrinking or enlarging its width to match the size of the most relevant connected component found. After a centroid P_i is then selected, the window is translated upwards in the BEV, with a direction and orientation determined by the position of the last few points detected. If, at some point, the window is located on a completely empty patch, its size is temporarily increased and the operation is repeated for a few times, in order to find the closest features and recenter. However, if after a few iterations none are found,the line is considered lost and the algorithm terminates. Otherwise, the computation ends as the windows touch the upper edge of the BEV image. At that time, the sequence of centroids collected along the way $\{\mathbf{P}_{l_i} = (x_i, y_i)\}_i$ is passed to the next component. For clarity, we refer to these points as *line points*. These points are limited in number and their position is very reliable. Nevertheless, a small amount of noise could be registered as the lines in the image mask present irregular borders.

To conclude this section, the overall process is depicted in Figure 4.7.

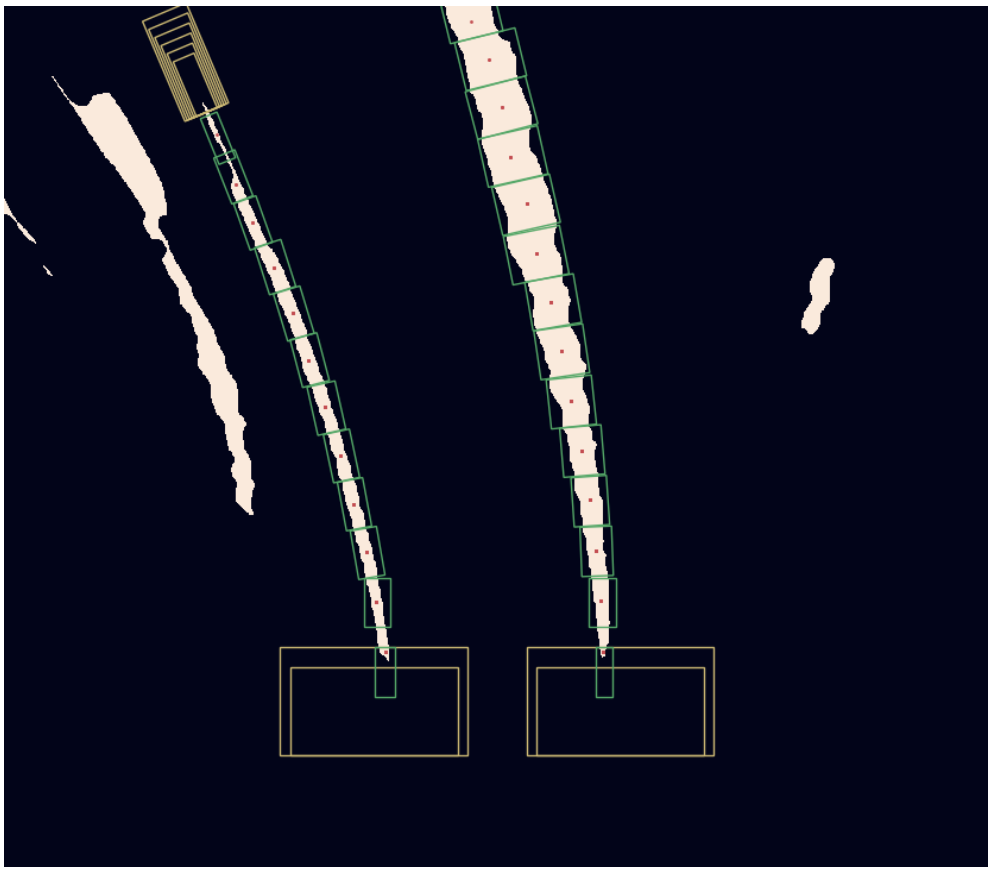


Figure 4.7: Window-based line following (WLF) algorithm. Given the post-processed feature points (beige), a window (yellow) is placed where a line is expected to be found (bottom). In this case, as no feature points is detected, the window is iteratively enlarged. When a feature is matched, the window (now green) is shrunk and the line point (red) is selected. Iteratively, the window is moved in the expected direction of the line and a new line point is searched and found. The width of the window is adaptively changed in order to contain all only the relevant feature points. If no feature points are found in a patch (yellow, top of left line), a recovery process tries again to enlarge the window and search for possible candidates in a neighborhood. In this example, no point is found, and thus the searched is stopped.
Notice how the algorithm is capable of recognizing the correct lines regardless of the presence of other spurious detections in the scene (on the left). Notice also how, thanks to the adaptive window width, the selected line points appear fairly aligned and centered, regardless of the conditions of the feature mask (right line).

4.2.4.2 *Line fitting: x–y frame*

Given the measured sequence of line points $\{\mathbf{P}_{l_i} = (x_i, y_i)\}_{i=0,\dots,n}$ for each line l, we can finally find the best model able to represent them. We carry on this process independently for each interested line and, for this reason, in the remainder of this section we will refer to a generic detected line as l. A natural choice in line detection systems is to adopt a representation within the world reference frame. We point out that the similarity transformation between BEV and world space is given by the relation

$$\mathbf{X}_{\pi_G} = \left(P_{\pi_G}^{bev \leftarrow w}\right)^{-1} \mathbf{x}_{bev} \tag{4.10}$$

In this frame, the easiest choice is to apply a polynomial model. However, first and second order polynomials are often not able to capture the whole structure/shape of the line, while third order are highly sensitive to noise and tend easily to overfit. Despite this drawback could be attenuated with a more robust fitting algorithm, these models will never have enough expressive power to achieve the higher precision we need in our work.

In response to this known issue, cubic spline are often employed. For our experiments, we focus on cubic smoothing splines [108] as the spline model more versatile and at the same time efficient.

Indeed, they are able to capture each detail of the road geometry and to attenuate the effect of small random noise among the line points. However, their representation is too complex to be of any assistance for our purposes.

4.2.4.3 *Line fitting: s–ϑ frame*

To be able to manipulate a simple model, and at the same time correctly describe the road, another class of models can be explored. For this reason, we consider here intrinsic models, and in particular the Whewell representation. In this study, we will generically consider a single marking l, and to lighten up the notation we will stop mentioning it explicitly when this does not lead to any ambiguity.

Considering a generic line marking l, to transform its line points into the Whewell intrinsic framework, we need to perform several

steps. For the time being, we fix for convenience the origin of this new coordinate system on the first line point detected, $\mathbf{P}_0 = (x_0, y_0)$. From here, we can easily measure the euclidean distance Δs_i between each line point $\mathbf{P}_i$ and its successor $\mathbf{P}_{i+1}$. At the same time, we can also evaluate the orientation ϑ_i with respect to the x axis of their connecting segment $\overline{\mathbf{P}_i\mathbf{P}_{i+1}}$.

Assuming the line points are sampled close enough, we can then approximate the cumulative arc-length s_i of each point $\mathbf{P}_i$ as

$$s_i = \int_{\mathbf{P}_0}^{\mathbf{P}_i} ds \approx \sum_{k=0}^{i-1} \Delta s_k \tag{4.11}$$

In this fashion, we construct a sequence of points $S^{(s,\vartheta)} = \{\mathbf{P}_i^{(s,\vartheta)} = (s_i, \vartheta_i)\}_i$ in the $s-\vartheta$ (or Whewell) space.

This sequence, while describing each line point with the same precision, has now the advantage to be representable as a 1-dimensional function

$$\vartheta = f(s) \tag{4.12}$$

independently from its original shape and orientation.

If we model f as a polynomial, we obtain a simple model yet with sufficient expressive power (see Section 2.1.3), and approach at the same time the representations used in lateral control.

Despite this potential improvement, however, if we do proceed as described, we do not obtain good results. It is clear then that a substantial issue is lying underneath. What is happening indeed is that the points in $x-y$ are affected by a small amount of random noise, which displaces them from the precise location of the line marking. These artifacts are completely normal and likely attributable to disturbances in the acquisition or false positives in the feature extraction. However, despite being imperceptible in the $x-y$ frame, they severely disrupt the fitting in $s-\vartheta$. As points oscillate around the line indeed, their absolute position remains almost unchanged/untouched, but their relative orientation, ϑ, severely fluctuates.

The only way to solve this issue and still exploit the advantages of the Whewell space would be if the noise itself was eliminated beforehand.Fortunately, we have already encountered a model very robust

to noise: smoothing splines. What we propose then is to employ the advantageous characteristics of such model and couple them with the simplicity of the $s - \vartheta$ implementation. In particular, we design a preprocessing step where a smoothing spline removes the mentioned noise and realigns each point on their best-fitting line. Then, instead of maintaning this model and its complexity, we just sample the reconstructed line within the fitting bounds and project the obtained noise-free points into our $s - \vartheta$ frame, where we can now construct the desired polynomial representation.

Following the considerations in [92], in the final version of our software the specific model adopted is a third degree polynomial, which can be represented as

$$\vartheta = f(s) = as^3 + bs^2 + cs + d \tag{4.13}$$

In particular, each coefficient is thought to be modeling a specific aspect of the curve, making the model very flexible. First, d is responsible for the initial orientation of the curve, at the origin. Second, c controls instead the curvature of the line, determining how sharp a bend in the road is at each distance from the origin. At last, we can think of a and b as respectively the acceleration and the velocity by which the curvature changes along the road, and they provide together an extensive range of variability to model a large variety of configurations.

As the points are sampled from a single, smooth line, the fitting procedure adopted can be as simple as linear least square. This technique is indeed known to be incompatible when the data used may contain outliers. In our situation, this is not possible, and therefore least square remains a valid and computationally-optimal alternative. The fitting with this technique can be described as follows.
Given the model

$$\vartheta(s) = \boldsymbol{w}^T \boldsymbol{\phi}(s) = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} s_i^3 \\ s_i^2 \\ s_i \\ 1 \end{pmatrix} \tag{4.14}$$

and given the data

$$\Phi = \begin{pmatrix} — & \boldsymbol{\phi}_1^T & — \\ & \vdots & \\ — & \boldsymbol{\phi}_n^T & — \end{pmatrix}, \quad \boldsymbol{\phi}_i = \begin{pmatrix} s_i^3 & s_i^2 & s_i & 1 \end{pmatrix}^T \tag{4.15}$$

$$\vartheta = \begin{pmatrix} \vartheta_1 \\ \vdots \\ \vartheta_n \end{pmatrix}^T \tag{4.16}$$

then,

$$\hat{\mathbf{w}}_{LS} = \left(\Phi^T \Phi\right)^{-1} \Phi^T \vartheta \tag{4.17}$$

As a last remark, we point out however that this model is valid only within its fitting interval and it does not allow extrapolations if not for very restricted sections. Indeed, although polynomial intrinsic models are widely used to model short tracts of the road, they cannot be used to represent long sections. The limitation of these representations is in fact that outside of a certain interval, they grow to infinity in the theta-dimension, and this characteristic reflects back on the Cartesian frame with the generation of spirals. One aspect to monitor then when using these models is the formation of undesired spirals when working in regions where no points were collected.

4.2.5 TEMPORAL CONSISTENCY

Up to now, our system is capable of processing an acquired image and detect the line markings within it on a frame-by-frame basis. However, as it is used to process image sequences, most of the information that could be saved from one frame and exploited in the next is currently lost. For this reason, in general, line detection systems feature a mechanism to enforce the temporal consistency of their detections. This also gives more robustness to the overall system in case the acquisition process fails for some, isolated frames or too much noise in the images prevents a correct detection.

In our system, the temporal consistency is enforced in several ways. First, the information from our past estimates is used to facilitate the feature points selection and improve the results of the WLF algorithm. In particular, if a line was detected in the last frame, the search for a new one is started exactly where the previous was found, saving computational time and avoiding the uncertainty of blindly searching the whole region. Moreover, when during the search a line is lost, because no feature points are found within a window even after enlarging it, instead of quitting the search and keep only the points so far collected, we can start a recovery procedure. This is based on the idea that, if a line was there previously detected, it should now still be found in the same region. The loss of track could then have been caused, for example, by a temporary object partly obstructing the view of the line, an imperfection in the acquired image that impedes its detection (e.g. lens flare) or the simple discoloration of the line marking due to wear and weather conditions. In all such cases, what the algorithm could do is to remember the shape of the previous line and move along it until new feature points are discovered close to such path. By doing so the algorithm assumes that when this happens the temporary disturbance has been surpassed and the new features enable it to reconnect with the lost line.

The other, more structured way to exploit the temporal information, is to use it on the line models themselves, introducing a tracking framework to filter their measurements. These algorithms can reduce, on one hand, the impact of noise and false detections, while on the other are also capable of stabilizing the lines and predicting their shapes and positions even if a few measurements are missed. Notice however that their task is not as trivial, as the vehicle is in motion on the road. For this reason, not only new portions of the lines become visible only in new frames, but also the aspect of the past ones is successively distorted by the consecutive shifts of view-point.

To better comprehend the details of these latter solutions then, in the following sections we analyze at first how to stabilize the position of each line, tracking their origins, and then how their actual shape can be filtered in time.

4.2.5.1 *Origin tracking*

At this point, the origin O of each line is refreshed at each frame with the position of the closest line point detected

$$O = \mathbf{P}_0 = (x_0, y_0) \tag{4.18}$$

However, we have no guarantees on the correctness of such point, which could be affected by noise, or on how far from the vehicle it will be detected each time. Moreover, leaving this mechanism unaltered would generate several inconsistencies in the overall measurements if the line shape is instead tracked, as it will be in the next section. For these reasons, this behavior must be corrected. We do so introducing a simple tracking algorithm.

In principle, we could track the complete position of the origin, (x_0, y_0), but this would be complex and redundant. To understand this claim, we can think of a plausible line marking l and fix an x coordinate close to the vehicle, x_f. Regardless of its shape then, we can always determine at which ordinate y_f, l intersects the line $x = x_f$. This is the same as measuring the lateral offset of l with respect to a fixed forward distance from the vehicle and perpendicularly to the x axis. In this way, we can fix x_f at a reasonable distance (for us, $x_f = 2m$) and perform the tracking only on the single quantity y_f.

Of course then, the initial origin O of our line must be preemptively moved to lie on the line $x = x_f$. This however only requires a simple computation, as we can do it with a horizontal translation of the line model in the intrinsic $s - \vartheta$ space. The only value needed to compute is then the arc-length s_f associated to the intersection point. As there is no closed-form correspondence between $s - \vartheta$ and $x - y$, we would need to solve an integral equation of the form

$$\int_0^{s_f} \cos\vartheta(\sigma)\, d\sigma = x_f \tag{4.19}$$

Nevertheless, this potentially expensive computation is quickly solved with the usage of a lookup table: at first, the closest point to $x = x_f$ in the table is directly accessed, obtaining its arc-length. Then, starting from it, we can iteratively integrate the formula with

significantly small increments $\Delta\sigma$ and stop right after passing x_f. Notice that this operation could achieve arbitrary precision as long as the increments $\Delta\sigma$ are kept small enough.

Once s_f is determined, then the model of the line is translated horizontally of such quantity, applying a transformation

$$\tau : s \mapsto s - s_f \tag{4.20}$$

At last, we can track the offset y_f. The algorithm used is a simple sliding window filter. From our tests, a simple moving average (MA) achieves good performances with a regular driving, while a weighted moving average (WMA), more dynamic, is needed for a more aggressive driving style. Formally, the two methods are described respectively as

$$\begin{aligned} \hat{y}_{f_{k,MA}} &= \frac{1}{n}\sum_{i=0}^{n-1} y_{f_{k-i}} \\ \hat{y}_{f_{k,WMA}} &= \frac{2}{n(n+1)}\sum_{i=0}^{n-1}(n-i)y_{f_{k-i}} \end{aligned} \tag{4.21}$$

where k is considered the current state and n is the characteristic parameter of the filters, indicating the maximum number of successive samples to consider in the computation, which is performed iteratively.

The result of filter y'_f, together with the fixed abscissa x_f, are finally assigned as new origin O^{tr} of the tracked line, and this closes the tracking of the line position. What is left now, is to track its shape and this will be covered by the next section.

4.2.5.2 *Line tracking*

As the vehicle moves on the road, the shape of the line markings in front of it changes. However, this change is slow and any point part of a line marking remains several seconds in the scene before disappearing. We can then assume that a line detected in one frame will be not the same but very similar to the line detected in the subsequent one. It is thanks to this consideration that it is possible and simple to introduce a tracking algorithm into the pipeline.

As already mentioned in Section 2.1.4, the literature on line detection is mostly oriented towards Bayesian filters, treating especially Kalman filters (KF) and extended Kalman filters (EKF). However, these methods rely on a model of the vehicle dynamics and expect measurements of its kinematic behavior over time, information that is not always available and that, as we will shortly see, is not indispensable.

What we propose is instead a line filtering approach that entirely relies on vision and is based on the recursive least squares (RLS) adaptive filter [109].

While the Bayesian techniques only look at the measured parameters of a model, and smooth then their changes over time, this method directly acts on the detected points, searching a line that best fits the new measurements but is also in accordance with the old ones.

In particular, we design this adaptive filter to receive in input, at each time step, the set of line points observed in the respective frame. With these, its overall model estimate is updated, following a weighted least squares scheme. Entering the filter with a full weight, points are considered to lose importance as they age, and thus their weight is exponentially reduced over time. The reduction factor μ is typically named, for this reason, the *forgetting factor* of the filter. With this mechanism, the filter is able to maintain memory of any past experience and at the same time to always focus on the most recent data. Computationally moreover, the filter is not required to physically keep memory of each point ever seen, and instead a smart update rule is widely known.

For a cubic polynomial model as ours, this goes as follows. Recalling at first Equation 4.13, i.e.

$$\vartheta = f(s) = as^3 + bs^2 + cs + d \tag{4.13 rev.}$$

of parameters $\boldsymbol{w}$

$$\boldsymbol{w}_t = \begin{pmatrix} a_t & b_t & c_t & d_t \end{pmatrix}^T \tag{4.22}$$

Given also the observations, at time t,

$$\boldsymbol{s}_{t_i} = \begin{pmatrix} s_{t_i}^3 & s_{t_i}^2 & s_{t_i} & 1 \end{pmatrix}^T, \quad i = 1, \dots, n \tag{4.23}$$

$$\vartheta_t = \begin{pmatrix} \theta_{t_i}^3 & \theta_{t_i}^2 & \theta_{t_i} & 1 \end{pmatrix}^T \tag{4.24}$$

and finally assuming our real process to be constituted of a deterministic term, which we seek, and a stochastic one, which we want to remove,

$$\vartheta(t) = S_t w_t + \eta(t), \qquad \eta(t) \sim \mathcal{N}(0, \Sigma) \tag{4.25}$$

we can then incrementally update our model parameters w by computing, for each $i = 1, \dots, n$

$$e_{t_i} = \vartheta_{t_i} - w^T s_{t_i} \tag{4.26}$$

$$\tilde{R} = \left(1 + \frac{1}{\mu} s_{t_i}^T R\, s_{t_i}\right)^{-1} \tag{4.27}$$

$$R = \frac{1}{\mu}\left(R - \frac{1}{\mu} R\, s_{t_i}\, \tilde{R}\, s_{t_i}^T R\right) \tag{4.28}$$

$$\Delta w = e_{t_i} \cdot R\, s_{t_i} \tag{4.29}$$

$$w = w + \Delta w \tag{4.30}$$

We remark that the proposed approach does not operate directly on the parameters of the model. Indeed, the underlying idea is to perform, at each step, a weighted least square estimation over all the data received, with weights exponentially decreasing in time. Of course the recursive formulation then makes the computation feasible and faster than many alternatives. The main advantage of this approach is that no assumption is needed on the behavior of the parameters and it is instead only the accumulation of points through time to smooth the results. Moreover, since it works with points, only a single parameter, μ, needs to be tuned.

4.3 LANE PARAMETERS ESTIMATION

Given the complete and stable representation of the two lateral lines, we can now estimate the fundamental parameters needed by a control unit to maintain the vehicle within its lane. This stage is usually

not present in line detection works and is the core of our technique, realizing the link between perception and control.With exactly this in mind, we aim then to estimate the fundamental parameters needed by a control unit to maintain the vehicle within its lane. These are in particular the shape of the centerline, defining the trajectory to follow, and the heading and lateral displacement of the vehicle with respect to it. With their knowledge in fact, a planner can study the optimal trajectory and recognize how distant it is from such goal, in order to appropriately plan how to correct its course acting on the steering.

4.3.1 CENTERLINE SHAPE

We begin from the estimation of the centerline. Given a representation of the two lateral road lines, our goal is to model the line passing exactly between them and, in our case, defining the center of the road.

In principle, this is not a trivial task. Indeed, as no parallelism between the two lines was enforced, it is hard to even give a formal definition of such line. Even then, no closed form solutions or helpful analytical manipulation are available, and only numerical techniques can be adopted.

If the lines were parallel, the centerline could be defined as the locus of points at equal distance from each line along rays orthogonal to them. In this case, one could then scan one line and, for each visited point, cast an orthogonal ray until the other line is met. As the lines are parallel, this ray will intersect both lines orthogonally. Thus, the midpoint of this ray will belong to the centerline.

Regardless, we did not enforce the parallelism of the two lines, because of two important considerations. First of all, this assumption would limit the expressive power of our system, since road lines are by design not parallel in several scenarios, such as in correspondence of highways on-ramps and off-ramps. However, what made us desist from it were mostly the slight variations of the extrinsic parameters of the camera registered when driving on the road. These fluctuations are very common and regard mostly the pitch angle of the camera. Their generation is usually attributable to the encounter of bumps on

the road or changes of load weight of the vehicle respect to when the camera was calibrated. Although these oscillations are not an issue for the rest of the system, which is robust enough to adapt them, they have the undesirable effectof representing parallel lines as converging or diverging in the BEV plane.

Given these considerations, we can see two possible ways to achieve our goal and estimate the position and shape of the centerline. The first, more mechanical but also computationally expensive, is based on the concept of distance transform (DT). Formally, the distance transform is a computer vision operation able to compute, for each pixel in a binary image, its minimum distance to any white patch. If we were to represent then the two lateral lines as a binary mask, we could apply this tool to collect the points with a maximum distance. Indeed, between the two lines, the values of the DT will be increasing until a ridge is formed. With another computer vision operation, such ridge could then be isolated. To cope with the possible insurgence of noise and false detection, we could at last apply our WLF algorithm and finally use the detected points to fit a model to the centerline.

This technique however, although straightforward, would be severely expensive from a computational point of view, as each aforementioned operation is highly demanding.

Moved also by this reason, with this work we propose (and describe in the following section) an alternative solution, exploiting the intrinsic parametrization of the lines—additional motivation for choosing this type of representations.

4.3.1.1 *Lateral lines re-projection*

Given two lateral lines in the Whewell representation $(s - \vartheta)$, we ask ourselves how to find their centerline, as the line running exactly between them. In the following we present our solution.

First of all, instead of relying on the models characterizing each line, we aim at employing directly the line points used for their fitting. These indeed still contain all the information needed on the shape of each line, but are substantially easier to manipulate than their counterparts, representations too compact for the task.

In this regard, our proposed solution employs together the line points coming from both lines and aims at computing a model that best represents both together, as a sort of "mean" model of the two. This consideration comes from the observation that the shape of the centerline is clearly influenced by the shape of both lines.

To do this however, the line points must be first projected, together, into a space where they are both comparable and, at the same time, only their respective shape is highlighted. This operation, key to this technique, can be performed in different ways, working in the $x - y$ frame as well as in the $s - \vartheta$ frame. In the following, we begin presenting some of these variations of the general algorithm, concluding with a detailed description of the one we ultimately designed for our final system.

To begin, a first idea could be to remain in $x - y$ and try to project each point towards the center of the lane. However, ti is not obvious how to proceed. In an optimal scenarios, with parallel lateral lines, the optimal solution would require to scan one line and cast a ray from each visited point until it intersects with the other line. The midpoint then between the original starting point and this intersection can be considered part of the centerline. Collected then several points with this method, a line model could be fit. This method however, although precise, requires a high number of intersection operations, which are computationally demanding to perform for a line in intrinsic coordinates.

Looking at alternatives, we devise and test the following possible transformations.

1. The two lateral lines are divided in horizontal stripes of fixed length by several horizontal segments. As these segments span from the left marking to the right one, their midpoint is then taken as candidate projected point.

2. After the same horizontal division used in item 1, from each end of the dividing segment we cast a ray orthogonal to the respective lateral line. We consider then the angle formed by these two rays and construct its bisector. We take then as candidate pro-

jected point, the intersection between this line and the respective horizontal segment.

3. We proceed as in item 2 to obtain the bisector, but we then intersect it with both lateral lines. The midpoint between these two meeting points is taken as candidate projected point.

From this study, we see that none of these simple methods is able to complete satisfactorily the task. In particular, the most accurate technique seems to be item 1, although it miserably fails as soon as a line with large curvature is tested.

Discourages by these results, we design a more complex process, starting from another interesting consideration and working, this time, in the $s - \vartheta$ frame. Before we begin, we notice that, given a line in $s - \vartheta$ with fixed curvature (i.e. a circle), the following holds

$$s = R\,\vartheta \tag{4.31}$$

This is valid however only in a very restricted number of cases, so technically it is of no particular help for us. However, for the time being, let's consider it valid. Moreover, taking into account the center of curvature $\mathbf{C}$ of a line, i.e. the point lying on the normal direction and at a distance equal to the radius of curvature of the line, let's additionally make the following assumptions:

- the two lateral lines (l_l,l_r) and the centerline (l_c) share the same center of curvature ($\mathbf{C}$):

$$\mathbf{C}_{l_l} \equiv \mathbf{C}_{l_r} \equiv \mathbf{C}_{l_c} \equiv \mathbf{C} \tag{4.32}$$

- the center of curvature varies smoothly from one frame the next

$$\mathbf{C}^t \approx \mathbf{C}^{t-1} \tag{4.33}$$

With this setup, we can define a simple procedureto project each line point towards to road center, without computing intersections or similar expensive computations. Indeed, although we will reason in the Cartesian space, what we will actually do is only to define

a linear transformation for the points in both lines within the $s - \vartheta$ frame. This transformation aims at rescaling each line in order to make their shapes comparable, and at that point fit the mentioned "mean" model. In particular, it is supposed to rescale the arc-distances between successive points, as if each point was translated closer to the road center.

In practice, repeating this steps for both lateral lines—here generically indicated as l—the procedure prescribes the following.

1. Compute $\mathbf{C}^t$ from l_c^{t-1}, using its heading and radius of curvature.
2. Find the line l_{l_0} passing through $\mathbf{C}^t$ and $\mathbf{P}_{l_0}$, the first line point in l.
3. Find the line l_{l_1} passing through $\mathbf{C}^t$ and $\mathbf{P}_{l_1}$, the second line point in l.
4. Compute $R_l = \|\mathbf{P}_{l_0} - \mathbf{C}^t\|_2$.
5. Compute $\Delta\vartheta_{l_1} = l_{l_0} \bullet l_{l_1}$.
6. Compute $\Delta s_{l_1} = R_l \cdot \Delta\vartheta_{l_1}$.
7. Compute $\Delta s_{c_1} = R_c \cdot \Delta\vartheta_{l_1}$.
8. Obtain the ratio $r_{s_l} = \frac{\Delta s_{l_1}}{\Delta s_{c_1}}$.
9. Define for convenience

$$s_{c_0} = R_c \cdot \Delta\vartheta_{l_0} \tag{4.34}$$

At this point, with the ratio in Equation 4.34, we can then define a coordinate transformation from the lateral line to the centerline and vice versa, all remaining into the parametric framework:

$$s_{c_i} = s_{c_0} + r_{s_l}\left(s_{l_i} - s_{l_0}\right) = s_{c_0} + r_{s_l} \cdot s_{l_i} \tag{4.35}$$

To recap then, with this transformation we are ultimately able to take all the points detected on both lines and collapse them onto the centerline in order to proceed with its fitting. We know that this

procedure is based on false assumptions, but as it produces good results.we maintain it in the pipeline as probably such assumptions, although not correct, were still good enough to produce a reasonable approximation of what we are seeking.

At last, the candidate points obtained are collected and passed to a model fitting algorithm. However, this operation is not as simple here as a least squares fitting, like it was for the lateral lines. Indeed, the points from the two lines, although reprojected and lying in the same space, are still representing two different models, and thus could be different between each other. In order for our centerline to reflect characteristics of the shape of both lines then, we take the following steps. At first, we model with a simple polynomial the projected points from each line, with a quick least squares fitting. We then sample the two obtained curves in $s - \vartheta$ then, and for each pair of values we take their average. As a result, we obtain a set of points, still in $s - \vartheta$, that represent at best both lines and that can be easily fit with a cubic polynomial, obtaining at last our final centerline model.

4.3.1.2 *Centerline origin*

As parametric models only describe the shape of a line, we are still missing where to set the origin of our estimated centerline O_c. Notice that this task is crucial, as the centerline will actually appear in the center only if this point is chosen correctly. To obtain then an estimate as much accurate as possible, we proceed with some steps.

To obtain an initial, rough estimate, the origin is at first maintained from the previous time-step

$$O_c^t \leftarrow O_c^{t-1}$$

With this value, a temporary centerline is generated. While we can assume for this line to have a correct shape, its origin is considered approximated, and we are thus aware it could be slightly unaligned with the actual road shape.

Nevertheless, we consider this estimate good enough and search for the position on the centerline $\tilde{P} = (\tilde{x}, \tilde{y}) = l_c(\tilde{s})$ in which a normal line

$\tilde{l}$ is set to pass through the center of mass of the vehicle CM. Formally, this point can be defined through the system:

$$\begin{cases} CM \in \tilde{l} \\ \tilde{P} \in \tilde{l} \\ \tilde{l} \perp l_c \end{cases} \tag{4.36}$$

What we have obtained is a line passing through the center of mass CM and orthogonal to the temporary centerline. As this approximated version is not completely reliable, we can assume the actual centerline origin will be close, but not on, the intersection point found. What we can assume is that the orthogonal line found will instead remain the same. If this is the case, what we could do as our next step is to exploit this orthogonal line and find its intersection with the lateral curves, as they can be directly measured and relied on. At last, the final estimate of the centerline origin can be obtained calculating the position of the midpoint between such intersections. This point indeed is by definition exactly in the middle of the road, assuming the orthogonal line was correct.

4.3.2 HEADING AND LATERAL DISPLACEMENT

If we obtained a model for the centerline as we described in the previous section, then the computation of the additional parameters needed by control, heading and lateral displacement, is trivial.

At first, we notice indeed that the heading of the vehicle is already represented by ϑ. All we need to do is find the point where to compute it. As it turns out, this is not as simple, since, for the control of the vehicle, we want to perform our measurements along the line passing through its center of mass CM. As this requires us to pass from intrinsic to extrinsic coordinates, no closed form formulas are available, and we have to solve a simple nonlinear equation. In particular, with reference to Figure 4.8, we need to look for a line $\tilde{l}$, passing through CM and crossing the centerline l_c perpendicularly; formally, we search

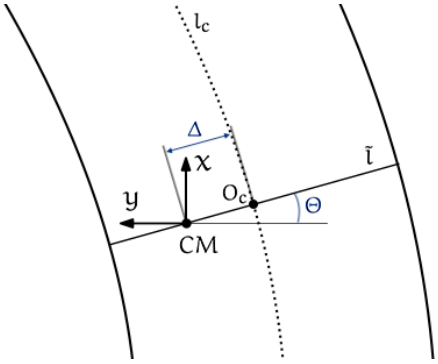


Figure 4.8: Derivation of the parameters to be estimated, Theta and Delta.

for a value $\tilde{s}$, corresponding to a point along the centerline O_c, such that:

$$\begin{cases} CM \in \tilde{l} \\ O_c \in \tilde{l} \\ \tilde{l} \perp l_c \end{cases} \tag{4.37}$$

Once this point is found, heading Θ and lateral displacement Δ can be found trivially, as:

$$\Theta = \vartheta(\tilde{s}) \tag{4.38}$$

$$\Delta = \begin{cases} +||O_c - CM||_2 & \text{if } O_{c_y} \geqslant 0 \\ -||O_c(\tilde{s}) - CM||_2 & \text{if } O_{c_y} < 0 \end{cases} \tag{4.39}$$

Notice that the sign of Δ is defined by convention.

4.3.3 TEMPORAL CONSISTENCY

if we were to take a moment and evaluate the performance of our system up to now, we would be extremely disappointed. Indeed, on one hand, the shape and position of the centerline would be oscillating and unsatisfactory, while on the other, more importantly, the parameters estimated would not match the reality.

Part of the problem is certainly lying in the fact that temporal consistency hasn't been enforced on the whole centerline estimation. Besides,

it is true that at least the lateral lines are tracked, providing some degree of robustness, but it is also true that we actually estimate the centerline only based on the line points, and not the model, of each detected line. To solve this issue, at first, we introduce a tracking framework for the centerline analogous to the one presented in Section 4.2.5.2 and based on the RLS method. This is perfectly fit for this task, as of points computed to represent the centerline are highly reliable thanks to their derivation and thus readily usable inside the filter.

With this system in place, the situation significantly improved, with the centerline maintaining now good shape. The other parameters however, are still far from satisfactory.

We suspect the cause of this issue has to be foundinto the numerous approximations we had to introduce to estimate their values. Besides, their estimation comes at the end of a long pipeline, and since the path between sensors and estimates is very long, several noise sources could have slipped in unnoticed. Even more so, the estimation of these parameters is performed completely through multiple indirect measurements, starting from the position of the lateral lines (direct)and passing from the centerline (indirect) before reaching the actual variables.For all these reasons, what comes to our minds is that maybe a tracking framework should be introduced to smooth out their estimation. As the values to be estimated cannot be directly measured and are instead computed indirectly, Due to their need of indirect measurements, what comes to mind is theBayesian filter framework. Moreover, as the measuring process for these values is very complex and largely nonlinear, the extended Kalman filter (EKF) seems to be the best option to select.

4.3.3.1 *EKF for parameters tracking*

We set up an extended Kalman filter to smooth the estimation of some of the parameters required by the control system. Before we enter into its details, Figure 4.9 depicts the main actors at play for this estimation.

Our state is for these reasons composed by θ, heading of the vehicle relative to the centerline, ρ, signed normalized lateral displacement,

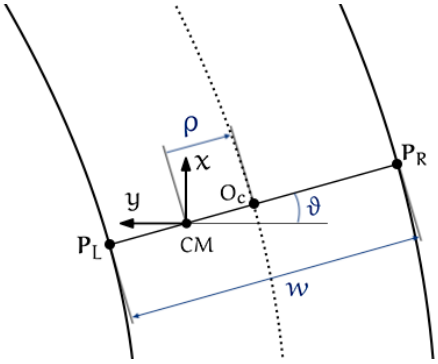


Figure 4.9: Reference for the variable used in the EKF. In particular, in blue we can find the state of the filter, θ, ρ and w, while in black are depicted the measurements $\mathbf{P}_L$, $\mathbf{P}_R$ and other reference quantities.

and w, width of the road. Notice that, this way, we formally split the lateral displacement $Delta$ into the two variables ρ and w, one measuring direction and percentage of the offset, and the other capturing the actual width of the road. We made this choice for two reasons: more importantly, this change simplified the definition of the measurement function, important to obtain fast convergence; and in addition, this allows us we obtain the additional estimate of w, useful support for some control systems.

As we do not consider the motion of the lines in time and assume instead their positions to be nearly fixed from one frame to the next,the state equation of our filter is the one of a simple random walk model.

As for the measurements instead, we consider the intersection points $\mathbf{P}_L$ and $\mathbf{P}_R$, already computed in section Section 4.3.1.2, at the end of the computation of the centerline.

As a consequence, the state space model representation of our system can be shown as

$$\mathbf{x} = \begin{pmatrix} \theta \\ \rho \\ w \end{pmatrix}, \qquad \mathbf{z} = \begin{pmatrix} x_{P_L} \\ y_{P_L} \\ x_{P_R} \\ y_{P_R} \end{pmatrix} \tag{4.40}$$

$$\mathbf{x}^t = \mathbf{x}^{t-1} + w_t, \qquad w_t \sim \mathcal{N}(0,\, Q)\,. \tag{4.41}$$

$$\mathbf{z}^t = h(\mathbf{x}^t) + v_t, \qquad v_t \sim \mathcal{N}(0,\, R)\,. \tag{4.42}$$

where the measurement function h is

$$h(\mathbf{x}) = \begin{pmatrix} x_{CM} - \frac{w}{2}(1-\rho)\sin\theta \\ y_{CM} + \frac{w}{2}(1-\rho)\cos\theta \\ x_{CM} + \frac{w}{2}(1+\rho)\sin\theta \\ y_{CM} - \frac{w}{2}(1+\rho)\cos\theta \end{pmatrix}, \tag{4.43}$$

with $CM = (x_{CM}, y_{CM})$ center of mass of the vehicle.

As a result of its introduction, the EKF significantly improves the performances of the system. However, we can still find some scenarios where, regardless of the filtering technique used, we have not enough information to make reasonable estimations. For this reason, in the next section we explore the possibility to expand the information we use, opening up to the introduction of the odometry into our system.

4.3.3.2 *Odometry introduction*

Regardless of how much we tune our system or improve our algorithm, certain scenarios seem to remain impossible to estimate, both in the centerline and, often, also in the lateral lines. Among them, typical is the case of double curves. As much as we refine the tuning of our system to achieve smoother transitions and keep the estimates, if wrong, at least under control, we notice that there will always be another possible curve where this new configuration is instead not enough to maintain the estimates controlled.

The reason for this limitation lies in the fact that, for how much advanced it could be, our system is not able to look backwards. Therefore, we can still try to adapt the estimates and smooth the changes in order to match the variations of shape most seen on the road, but there are potentially infinite road configurations over which our filter can do nothing.

Thinking for example at a double curve, when this is completely ahead of the vehicle, its estimation is simple and accurate, as the whole line can be observed. As the vehicle proceeds forward then, its measurements change, adapting to the double curve, while the tracking smooths out this transition, delaying the effect of newer detections in favor of older ones. This produces a smooth an acceptable estimation. However, when the vehicle will have begun the second curve, and the first will be entirely at his back the situation becomes more interesting. If we consider the measured line, this clearly fits the curve ahead. However, if we extrapolate it backwards, it will behave as if in presence of only one single curve, as it is not aware of what behind the vehicle, leading to completely wrong estimates. When we introduce tracking in the mix instead, this will use the information from the past to reduce the curvature of the measured, new line. As the filter updates its model at fixed time intervals, its parameters simply indicate for how long each measurement still influences its final estimate. If then we assume that, in this particular case, the algorithm is perfectly tuned, the combination of new detection and previous measurements with its parameters will generate exactly the real shape of the double curve. In this case however, we can now just think of traversing the same curve, with the same trajectory, but at lower speed. In this case, the time taken by the vehicle to reach the same point in the curve is larger. Thus, if the filter maintains the same tuning, many past measurements of the first curve that were before still fresh in its memory, are now fading away. As a result, its extrapolation performance will be less pronounced. Since this thought experiment can be constructed for any road and any system configuration, it demonstrates us that there always exist some configurations where our tracking is not enough to produce correct estimates, and it is clear that this aspect constitutes a

structural limitation of the current algorithm and not just a problem in its setup.

To complicate things, this issue has strong repercussions also on the estimation of Θ and Δ. Indeed, as shown, these parameters are always computed keeping the center of mass of the car CM as reference, and this point is obviously inside the car and thus behind the observed scene. To perform these measurements, we then have to extrapolate backwards our line estimates of a few meters. Therefore, although this short distance cannot completely disrupt the values read, if the line model we are following does not reflect the reality this computation is clearly faulty.

In search of a solution, we realize that while it is not possible for us to observe the lines behind us, we have an advantage: as we are moving forward on the road, what is behind of us now, must have been ahead of us before. In other words, as while we are driving on our lane we can see its shape for decades of meters ahead, instead of forgetting this information as soon as we move past it, we can exploit it and store it for future usage. With this data then, after some time we can model non only the road ahead, but also the one already behind our vehicle.

The only problem in doing so is that we have no information on our motion or, as it is the same, on the motion of the lines with respect to us. Nevertheless, a possible source of this information is the odometry captured by the vehicle. With it, we can proceed as follows. At each time step, while estimating the lateral lines, the line points detected are contemporarily stored. At the next time step then, we can project these points backwards using the odometry measures in order to reflect the motion of our vehicle, and then add them to the new detections to perform the fitting. As we move forwards, more and more points are accumulated, representing regions of the road not observable anymore. This way, the region where our lateral lines are fit grows also backwards. As we move further down the road however, it is not convenient to store these points indefinitely, as we would need a complex model to represent the shape of the road for longer tracts. Since we just need to know the position of the lines nearby

the vehicle, what we can then do is to maintain only past line points within 5-10m from it. This way, our simple models could still well represent the overall lines, with the benefit that we are only fitting actual data and not resorting to extrapolation. As we are resorting to the odometry measures, notice moreover that these data are usually noisy and tend to soon drift away from the correct trajectory. However, as we only keep points for the last few seconds of the journey, what we only need is the relative displacement occurred within a short distance. Therefore, despite measurement errors are still present, these are not large enough to negatively impact on our estimates.

To conclude, with the introduction of the odometry into the pipeline, our algorithm is complete and performs as intended. As a consequence, in the following chapter a formal evaluation of its performances will be presented.

5

RESULTS AND EVALUATION

Once the system is ready we can proceed with its evaluation, in order to validate the techniques we adopted and at the same time compare their performances with the rest of the literature.

The validation of a system can in general be done in several ways, according to the type of problem it solves, the tools at one's disposal and the nature of the system itself. As it is the engineer's job to solve practical problems, engineering systems often require a practical evaluation. When the field of study has a solid history to support its members, standard validation techniques are usually available, in the form of software and procedures designed specifically for that purpose. In Computer Science however, and in particular along the branches of Artificial Intelligence, a long tradition is yet to be formed. It is probably for this reason that when we came to this important stage in the life of our system, we could not find the desired support from the literature, and had to rely, at last, only on our own means.

Thus, since no relevant acclaimed dataset is available on the field, we employ, at last, our own research vehicle and the data collected with it for this purpose. In the remained of this chapter then, we first explore the details of our technical setup and the characteristics of the dataset collected with it, and only after, at last, we perform and describe our evaluations.

5.1 EXPERIMENTAL SETUP

As already mentioned, our entire evaluation is based on the data collected with the instrumented vehicle shown in Figure 5.1. For our purposes, this is equipped with the following sensing instrumentation.

CAMERA The images are recorded using a ZED stereo-camera with a resolution of 672x376. For reasons not dependent on our work, the acqusition is performed at 100 Hz, and a subsequent down-sampling brings the framerate down at 33 Hz.

GPS To record trajectory of the vehicle and lane coordinates, as we will soon mention, a Swiftnav RTK GPS is employed.

ENCODERS Finally, wheel and steering encorder units are mounted to retrieve the odometry measurements.

For what concerns our software system instead, its development is based mostly on the Python programming language, for its propensity to allow a very fast prototyping of new experiments and solutions. In fact, if a research system is implemented from scratch, as it happens for ours, several times no clear solution of a problem is known, and being able to quickly experiment on different alternatives is crucial for the final success of the problem.

This potential advantage however is usually paid in computational time and efficiency, as it is the case of our system. Although some components have been directly implemented in C++, most of our software was developed as a research tool, for the fast implementation and testing of new ideas, and not for running in a production environment. What this also means is that a substantial margin for optimization

Figure 5.1: Instrumented experimental vehicle used to acquire the dataset.

is present, and despite not performing at real-time at the moment, it offers full potential for future improvements of this kind. In fact, the CNN described in Section 4.2.2 is already capable of running at 13 ms per frame on a regular embedded platform, while a lighter version in our possession of the same network can even reach 5 ms per frame. Besides, the WLF algorithm, strictly iterative, is currently implemented in full using Python, which is known for its slow capability of processing iterations. By the same token, several computer vision operations are performed without particular optimizations and with no support for GPU computation. If all these aspects were to be fixed, for example implementing each component within a C++ framework, then great speedups should be registered. Even more so, all our experiments have been completed resorting to limited computational resources. In particular, in different situations we employed respectively

- a MacBook Pro (13- inch, 2016), with a 2 GHz Intel Core i5 CPU and 16 GB 1867 MHz LPDDR3 RAM;
- a Jetson Xavier embedded board;
- a server environment with 1 dedicated 12 GB GeForce GTX 1080 Ti out of 8 shared, 40 shared 2.20 GHz Intel Xeon E5 CPU, and shared 256 GB RAM.

5.1.1 DATASET

The dataset we employ for our entire evaluation has been acquired by our research group [1] and is one of few where the information recorded allows for the construction of a proper ground truth for the lateral control environment.

All data within it have been acquired on the Aci-Sara Lainate (IT) racetrack and test track. With a length of almost $1,5$ km, this circuit presents an optimal configuration for real street testing, thanks to its long straights, ample radius curves and narrow chicanes, together with several lane splitting and intersections.

The 8 laps recorded feature speeds ranging from 3 m/s up to 15 m/s and a spectrum of different driving styles, from the observance of an optimal trajectory in the middle of the road to a strongly oscillating one, registering heading changes for up to 40° and substantial lateral oscillations. To further differentiate the samples, two runs were also performed driving in the opposite direction, while in a couple of them an alternative route to avoid one of the chicanes. This mixture guarantees the heterogeneity of the dataset, suitable for testing the robustness of the algorithm in all driving scenarios. Figure 5.2 depicts the trajectories recorded in each ride to highlight their variability.

As the recordings were taken in sequential days, they do not capture, on the contrary, inclement weather conditions. In most of the recordings moreover, the illumination is more than acceptable, and all of them are performed during day-time.

As introduced above, among the measurements performed in each ride, the position of the vehicle is registered with fine-grained precision thanks to the RTK GPS adopted. With this information, the trajectory of the vehicle is reconstructed and utilized as ground truth. In addition to this information, the position of the lateral line markings is preemptively recorded thanks to the GPS receiver installed on the

1 The dataset here described has not been collected by the author, and his contribution was minimal in its overall realization. Said dataset and the modalities of its collection are here described only as the dataset is intensively used in the development and evaluation of the author's work and it is thus pertinent to this discussion.

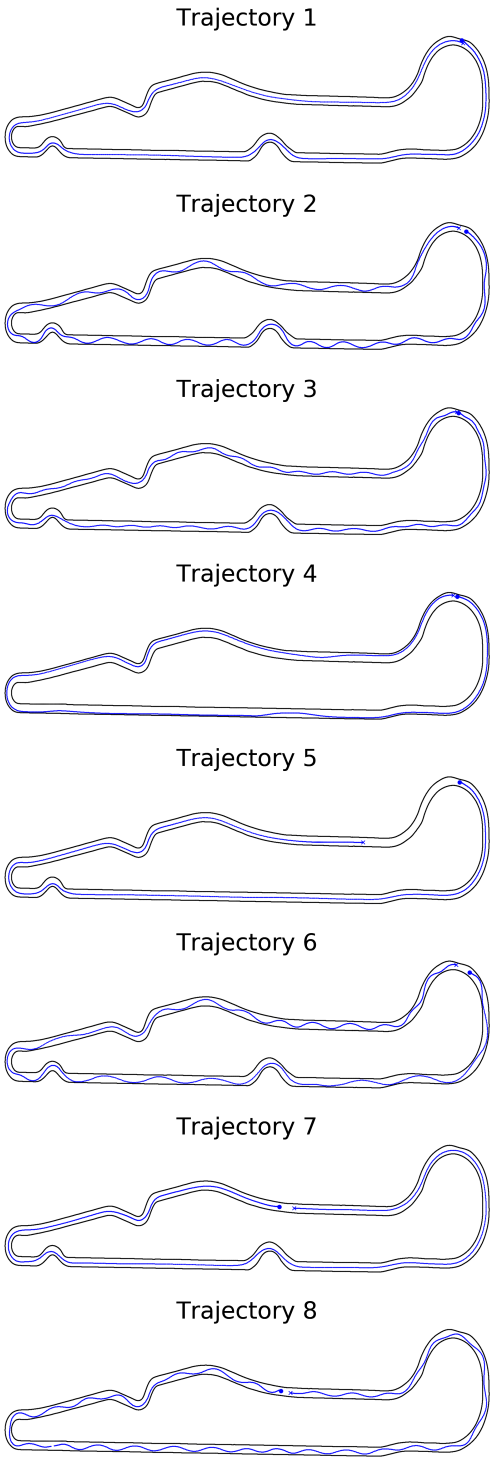


Figure 5.2: The 8 trajectories recorded in the dataset, each featuring different characteristics as driving style, average speed and lateral position of the vehicle.

experimental vehicle. From these data, the ground truth for Θ, Δ and the centerline shape can be generated. This process requires to map the desired area, discretize the overall route in several control points and finally, for each of them, to define the polynomials required by the lateral control algorithm. With the measured track boundaries, the road centerline is computed as the medium value of the two boundaries and it is then reshaped to guarantee a sample point each $ds = 0.5\,m$. This specific value for ds consents to bypass the oversampling of the GPS signals, while still ensuring smoothness and accuracy of the road map. At regular intervals ds then, third order polynomials are used to represent the centerline, each extending for the following 30 meters. It can be statistically demonstrated that this technique is a reasonable approximation of the centerline and obtains good results even well at distances of $30\,m$ from the vehicle. While this model will describe the ground truth for the centerline shape, the lateral displacement of the vehicle is computed as its distance to the closest sampled centerline point and its heading is extracted from the estimated trajectory as their tangent direction.

5.2 SYSTEM EVALUATION

Given the extensive dataset at our disposal, we set off to evaluate our system. To do so, we run our full pipeline on each of the 8 trajectories, recording our measurements of the lateral lines and estimates of centerline, heading and lateral displacement, together with the overall internal state of the algorithm.

Obtained all the results, we can proceed to evaluate them in two ways. At first, these data can be visualized, and we can use this representation to perform a qualitative analysis of the strengths and deficiencies of our system in each different observed scenario. Indeed, during a complete driving test many situations arise, and while it is true that some of them could trigger inaccurate estimates, it is very important to associate these episodes with their most likely cause. Once this is done, since we are also provided with a ground truth, we

can enter into the details of each scenario and perform a quantitative analysis of the output.

Notice that both steps are important and not mutually exclusive. Indeed, let's assume for example that, at some point of a driving test, a particular scene or environmental condition completely disrupts the estimation process of our system. From such event, if we only relied on a single evaluation method, two possible unwanted situations could arise.

On one hand, an overall quantitative analysis could remain unaffected if the disruption has place on a short time scale, and we could thus overlook the problem. This however would leave its original cause at large, free to come back in more serious situations, leading to system failures and potentially even crashes.

On the other hand, if the disruption was prolonged in time but associated to a single, long-lasting and unforeseeable reason (e.g. temporary signs direct the vehicles outside of their lanes in the proximity of a crash site), then an evaluation of the system based only on the quantitative measurements would unjustly penalize it. Instead, the data acquired during that time frame should be isolated, and the evaluation should also be performed considering only the unaffected part. Moreover, with this approach, if the disturbances are are determined to be due to a system malfunction, then specific effort could be devoted only to solving that particular issue.

5.2.1 QUALITATIVE RESULTS

With the amount of information registered from each driving test, we proceed by steps. At first, we can look into the output of the single line detection system, monitoring its behavior on each different scenarios encountered and highlighting any possible issue they could present, together with their achievements.

We start by describing, through Figure 5.3, the general output of our system on a frame taken on a straight road and presenting no particular difficulties. In the image, we highlighted the lines detected and modeled by our algorithm, together with their representation

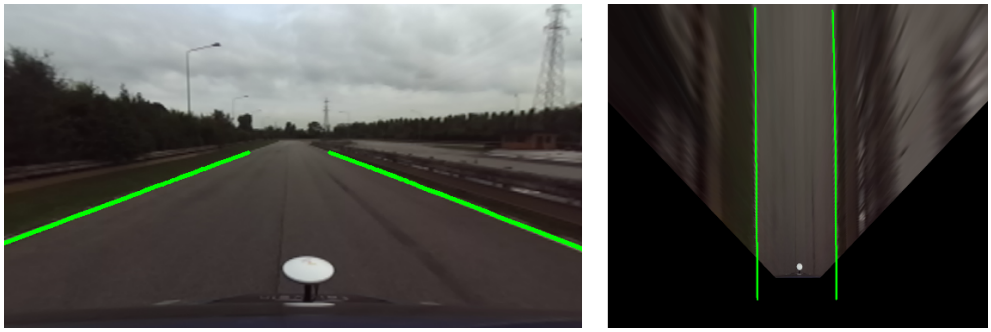


Figure 5.3: Line detection on a straight road.

Figure 5.4: Line detection on an extensive road bend.

in BEV. Moving to a more interesting example, in Figure 5.4 we can observe the system dealing instead with a bend in the road. Thanks to its versatile intrinsic representation of the lines and the harmonic collaboration of each component in the pipeline, the algorithm is able to precisely describe both lines.

As we confirmed our system can perform line detection on the most basic scenarios, we can then introduce the estimation of the centerline and complete the overall picture.

To this end, Figure 5.5 proposes once more the simple scenario analyzed in Figure 5.3, but adds our estimated centerline. As we will do again in the rest of this chapter, the visualization range of the BEV is set here to include, in black, a portion of the road behind the camera. In this way, we can then use this image to display all the elements of the estimated road geometry. In this case, the position of the center of mass CM the vehicle is included, together with the normal line passing through it (indicated as $\tilde{l}$ in Section 4.3.1.2).

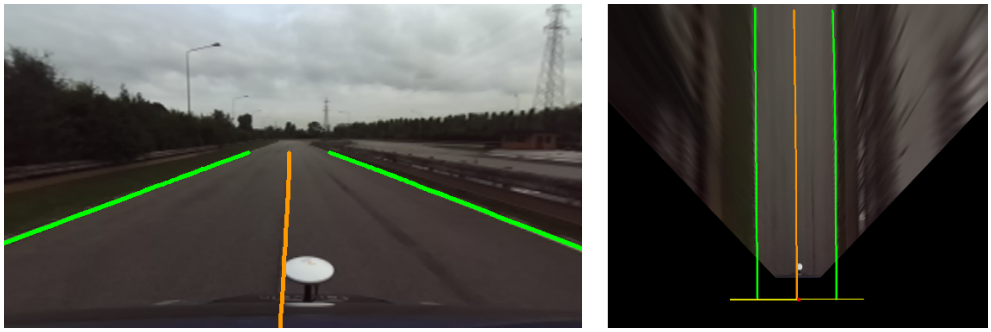


Figure 5.5: Visualization of the complete output of our system, including line detection (green) and centerline estimation (orange). Notice that, in the BEV image, the position of the center of mass of the vehicle (red) and the orthogonal line passing through it (yellow) are shown for completeness.

As we mentioned in Section 4.3.3.2, the introduction of the odometry was crucial for the the system to correctly model the road in non-trivial scenarios. To highlight then its importance, in Figure 5.6 we consider the case of a double bend. When the vehicle has yet to enter the bend and this is completely ahead of it, the odometry measures are actually not needed, as the versatile line models adopted by the system were already fully capable of representing it. What we show here however is how the system is now capable of modeling the entire bend even when already half-way through it and most of it is not visible anymore from the camera.

With these examples, we observed how our system behaves appropriately in general driving situations. Nonetheless, we observed some particular scenarios still able to challenge it.

Above all, the three chicanes in the track represent an almost insurmountable obstacle for our algorithm. Analyzing all the results as in Figure 5.7, we found this to be associated mostly with three important issues. The first one is merely technical and is related to the field of view of our vision system, often unable to capture large parts of the line because of its high curvature. Although we have no power on this issue, we can instead act on the second one, which has to do with our feature extraction stage. The CNN we employ, indeed, often misses important detections when inside a chicane, probably influenced by never seeing a road structure so irregular during training. At last, even

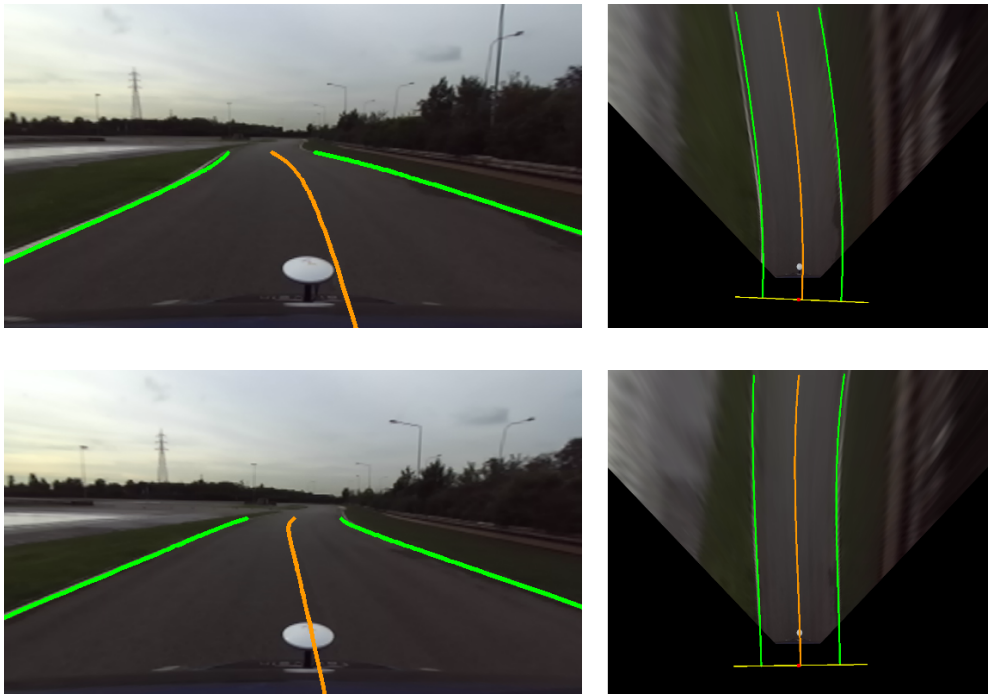


Figure 5.6: Line detection and centerline estimation in presence of a double bend. Thanks to the odometry measurements, the system is able to remember the shape seen in the past (top) and adapt its estimate of future scene (bottom) accordingly (notice the change of concavity, typical of double bends, which could not be estimated from the visible scene).

if the features are correctly detected, we experienced difficulties not in modeling the actual line, task fairly simple with our intrinsic line representation, but instead in updating the tracked estimates fast enough. The abrupt shape changes of these road elements, in particular, pose a serious challenge to our line tracking systems.

Aside from these curves, minor issues were also registered in correspondence of a sudden break in one of the line markings, due for example to the intersection with another artery of the track (Figure 5.8). Nevertheless, the system displayed a good degree of robustness to this disturbances and was able to continue its task with only minimal and temporary consequences.

To conclude then, this analysis convinces us that, at least on a qualitative point of view, the system is capable of performing as required in the most common driving scenarios. Some situations still

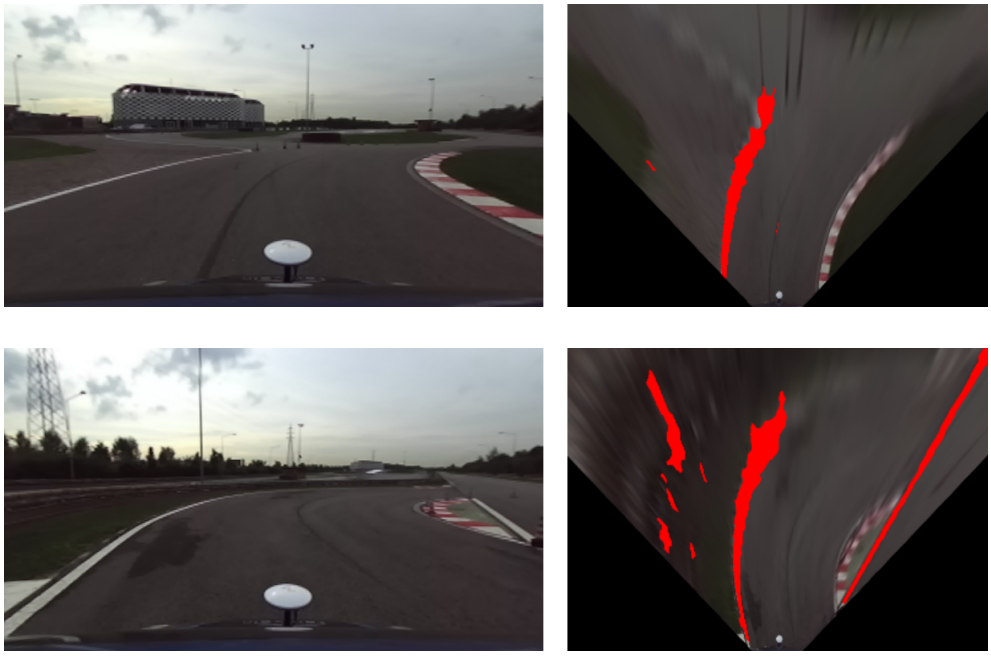


Figure 5.7: Examples of the issues registered around the chicanes. In this particular scenes, the faults are attributable to the CNN, which fails to identify the line (top) or identifies a wrong line (bottom).

affect it, as intersections with other roads, but it is robust enough to recover and continue its task. Some road configurations however completely overcome its capabilities in occasion of strong curvature changes or uncommon road geometries (e.g. in a chicane).

5.2.2 QUALITY OF THE ESTIMATION

Reassured about the strengths of our system and aware of its weaknesses, we can consciously proceed with a quantitative evaluation of its capabilities. In particular, as the system was designed to aid the planning and control unit in the task of lane following, we directly evaluate its final output: the estimated heading Θ and lateral displacement Δ of the vehicle.

As we need to assess the response of the system in a long interval, it is important to select an appropriate evaluation metric. Since we do not require it to be differentiable, and we have no reason to impose

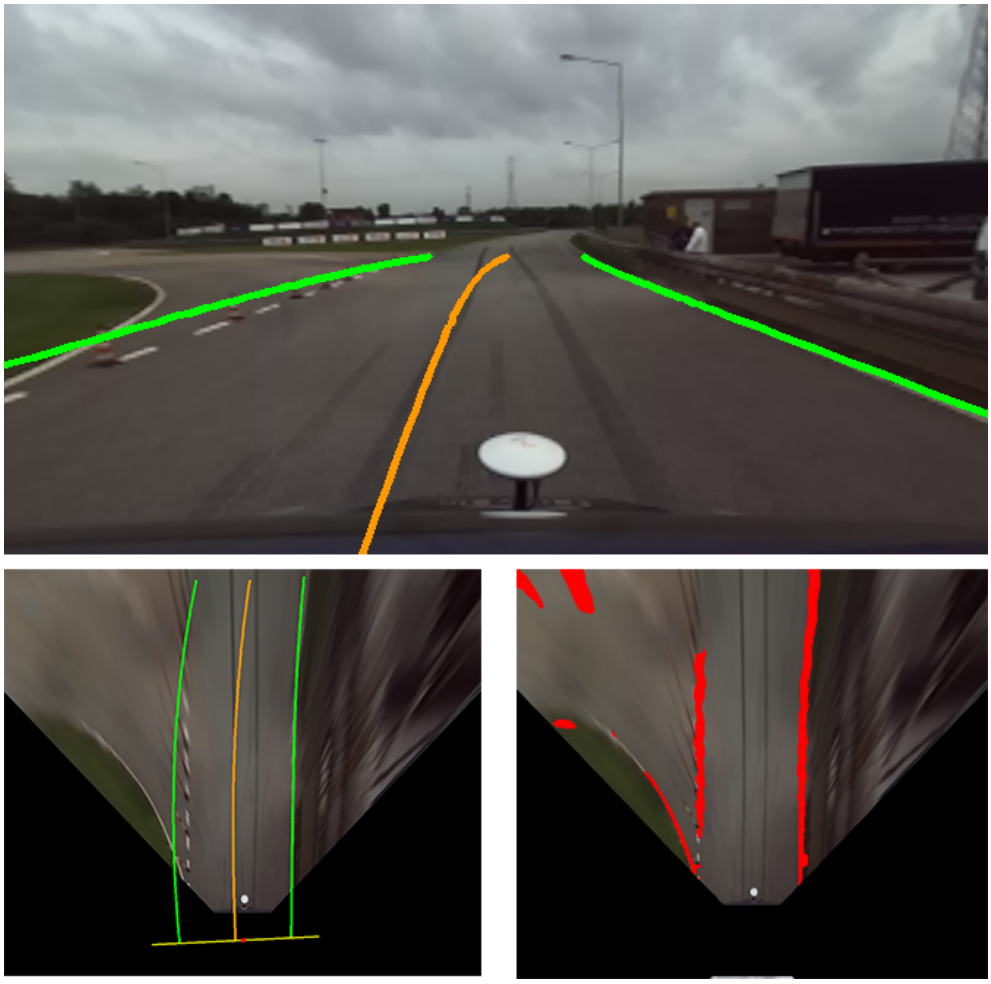


Figure 5.8: A split on the road together with a wore out marking deceive the algorithm for a few frames. While it is soon able to recover the tracked estimates during these instants are slightly perturbed.

different weights on each error according to its magnitude, we adopt the mean absolute error (MAE), defined as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5.1}$$

where y_i and $\hat{y}_i$ are respectively the predicted and true values, and n is the number of points considered. As opposed to other alternative metrics moreover, the MAE provides also a clearer interpretation of its results, being just the average error registered among the data.

	$MAE(\Theta)$ [deg]	$MAE(\Delta)$ [m]
Driving style: straight		
Trajectory 1	5.503	1.996
Trajectory 4	4.753	1.706
Trajectory 5	5.400	1.698
Trajectory 7	2.551	0.739
	4.501	1.518
Driving style: oscillating		
Trajectory 2	6.571	1.813
Trajectory 3	5.915	1.708
Trajectory 6	6.528	1.986
Trajectory 8	5.165	0.908
	6.039	1.592
	5.343	1.559

Table 5.1: MAE for the estimation of heading and lateral displacement over each entire trajectory in the dataset. The trajectories are split according to their driving scenarios, and cumulative measurements over the same category and for the overall dataset are also indicated for clarity.

As we are all set, we begin with an evaluation of the overall performances of the system on the entire dataset. Table 5.1 reports the MAE for the estimation of heading and lateral displacement over each of the 8 trajectories analyzed. These are divided into two main driving styles, *straight* and *oscillating*, in order to facilitate the interpretation of their results.

With mean errors above 5 degrees for the heading and 1.5 m for the lateral displacement, we are not particularly satisfied of these results. However, we recall the issues and adverse scenarios we observed in Section 5.2.1, and in particular, the dooming effect of the chicanes on the performance of an otherwise acceptable system. To better evaluate this effect, we then analyze our estimated values and their deviation

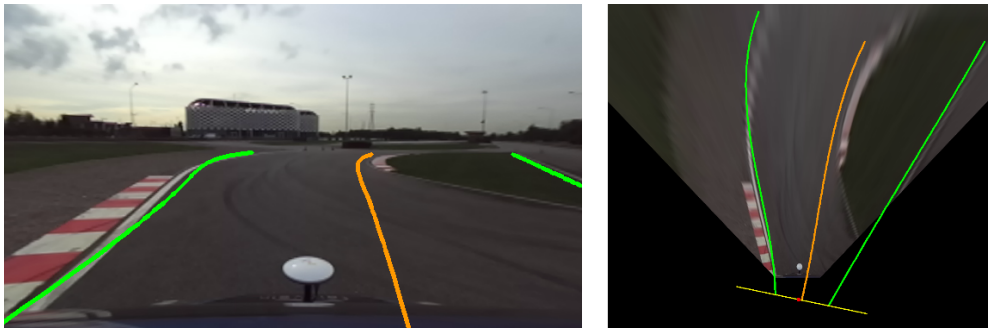


Figure 5.9: Frame captured for *Trajectory 7* at t = 105 s, showing how peaks in the estimation errors are often correlated with the crossing of a chicane.

from the ground truth data during time, reporting in Figure 5.11 and Figure 5.12 our findings for each one of the 8 trajectories.

As suspected, looking at the representations of the absolute error (Figure 5.12), we can see the presence of roughly three peak regions, each associated to a chicane in the road. This is confirmed also by the values assumed by the ground truth (Figure 5.11) within those regions, suggesting also that the sharp change of curvature and orientation of the chicanes might even be disrupting the ground truth computation. Notice also that large errors are reported also at the beginning and end of each trajectory, and this is also probably attributable to issues in the ground truth computation.

Contrarily, outside of these regions, where the driving scene are more common and expected by the algorithm, the estimated parameters seem to be consistently following the ground truth.

To clarify this situation, we can then inspect the conditions bringing the system to failure, analyzing the scene recorded in the corresponding instants and the line there detected. Examining for instance the overall recording in *Trajectory 7*, a clear peak is present in both heading and lateral displacement right before t = 105 s. Checking the data available, at that point in time we can observe that the vehicle was indeed crossing a chicane Figure 5.9.

As this evaluation is then negatively biased by the presence of unnatural scenarios, what we can do is to validate our results only on isolated sections of the whole trajectory. Continuing on the instance of *Trajectory 7*, we can then isolate a driving section where only traditional

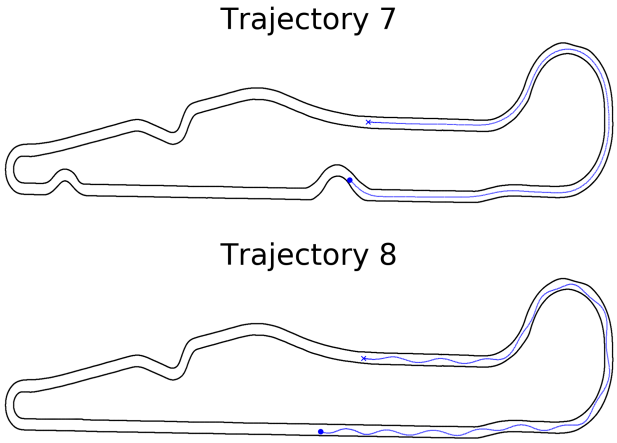


Figure 5.10: Restricted evaluation tracks. They do not feature any chicane, but still include the most interesting testing scenarios: straight road, curve road and double bend.

road shapes are found. Considering then only the portion of the lap where $t > 105$ s (Figure 5.10), the respective section of the track includes, in order: a straight tract, a double bend and a long curve. Given this multitude of road shapes, this part of track remains a perfect scenario for testing our system. We can also perform the same operation on the results shown in *Trajectory 8*, in order to represent both driving styles. Also in this driving scenario, the last chicane is passed for $t > 105$ s. The results of this restriction, reported in Table 5.2 and depicted in Figure 5.13, are now severely improved, with a mean heading error of only 1.8 degrees for the straight driving and 3.8 degrees for the oscillating one. The lateral displacement remains instead stationary below 1 m.

We can already appreciate the precision obtained with our system, especially considering it only relies on vision and vehicle odometry. However, it would be interesting to compare its performances with another lane following system, or even better with the actual requirements of a lateral control module, but to the best of the author's knowledge, no work in the field has been published featuring a comparable evaluation.

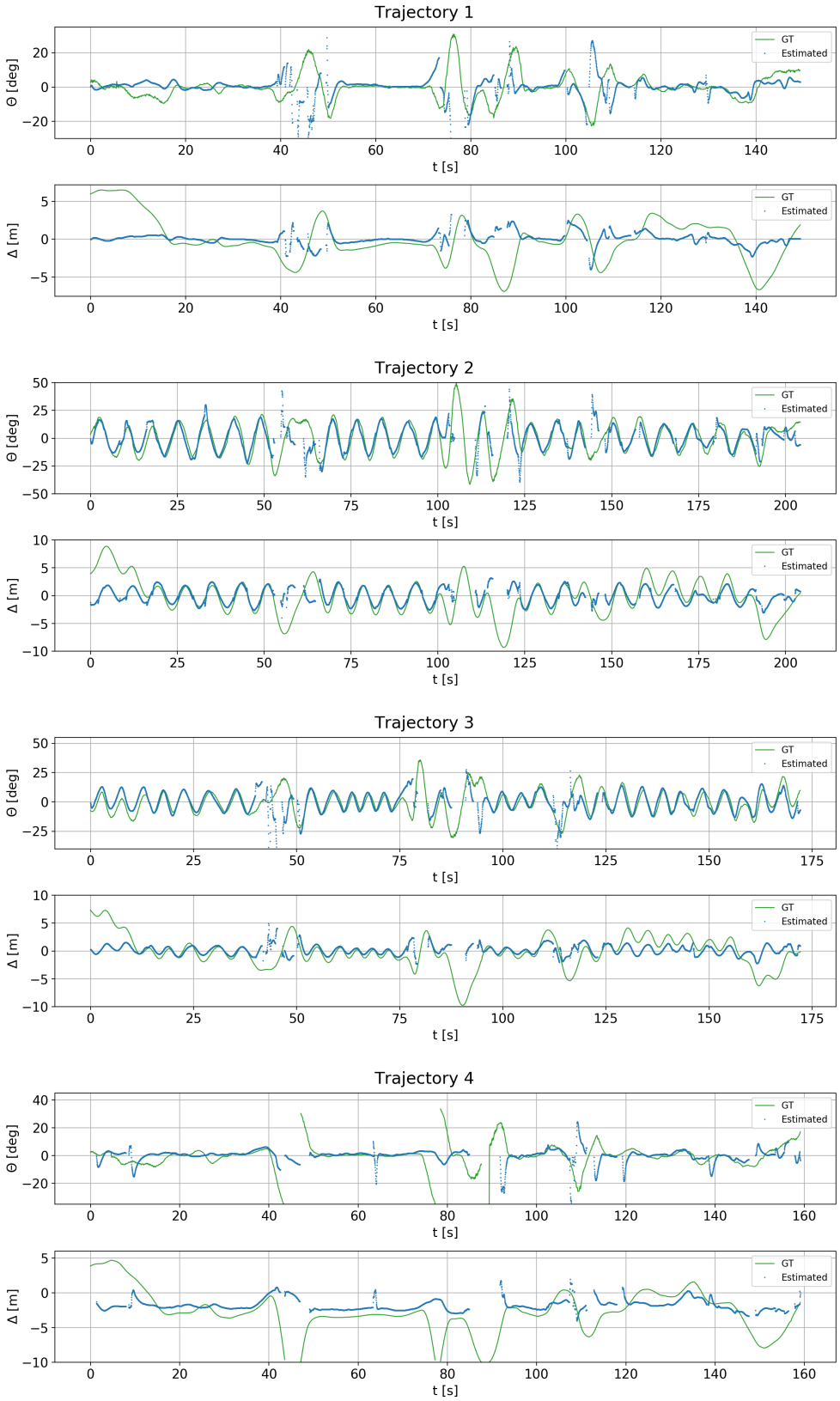


Figure 5.11: Behavior of the estimated heading and lateral displacement in time, compared to the ground truth provided with the dataset.

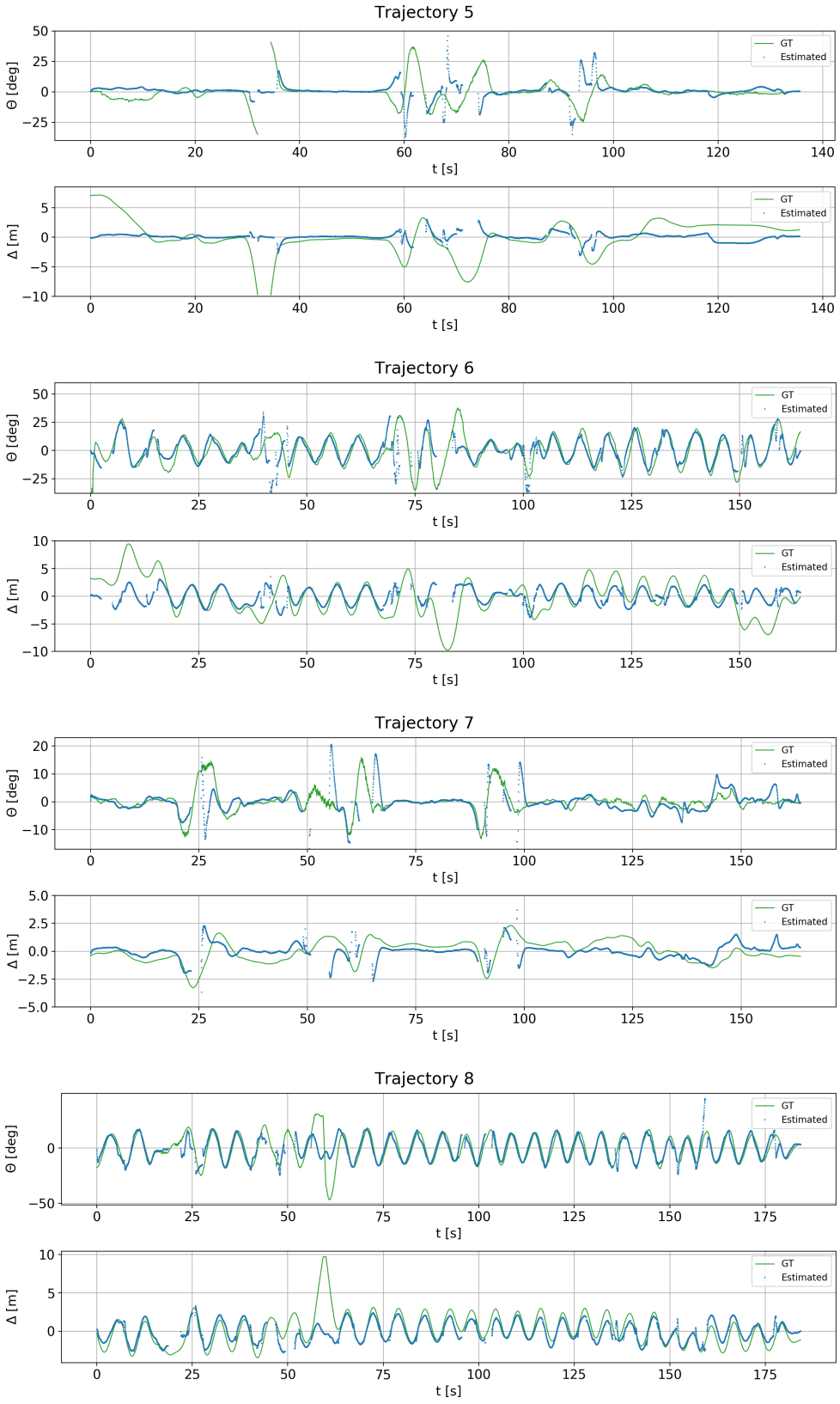


Figure 5.11: Behavior of the estimated heading and lateral displacement in time, compared to the ground truth provided with the dataset.(cont.)

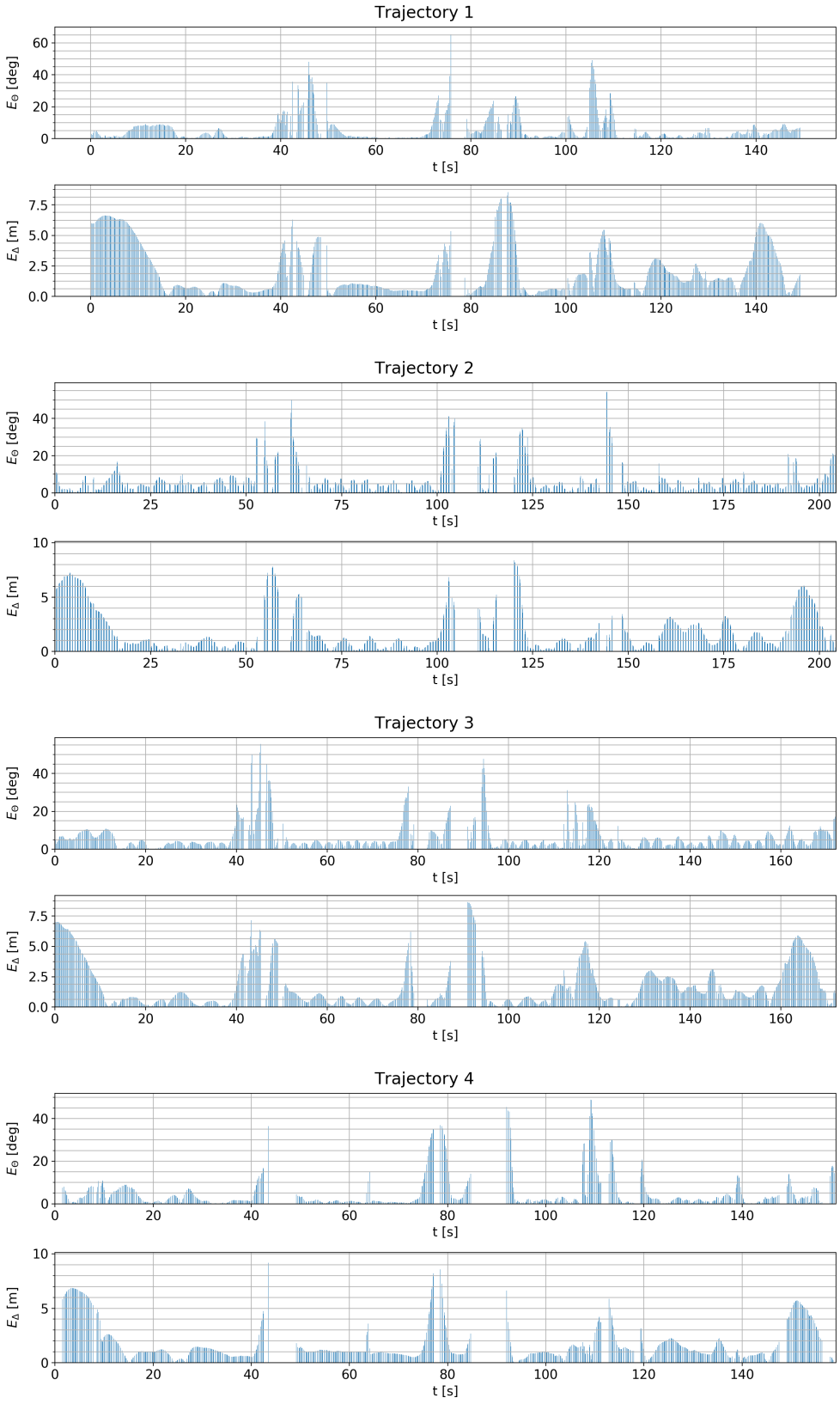


Figure 5.12: Absolute error of the estimation of heading and lateral displacement.

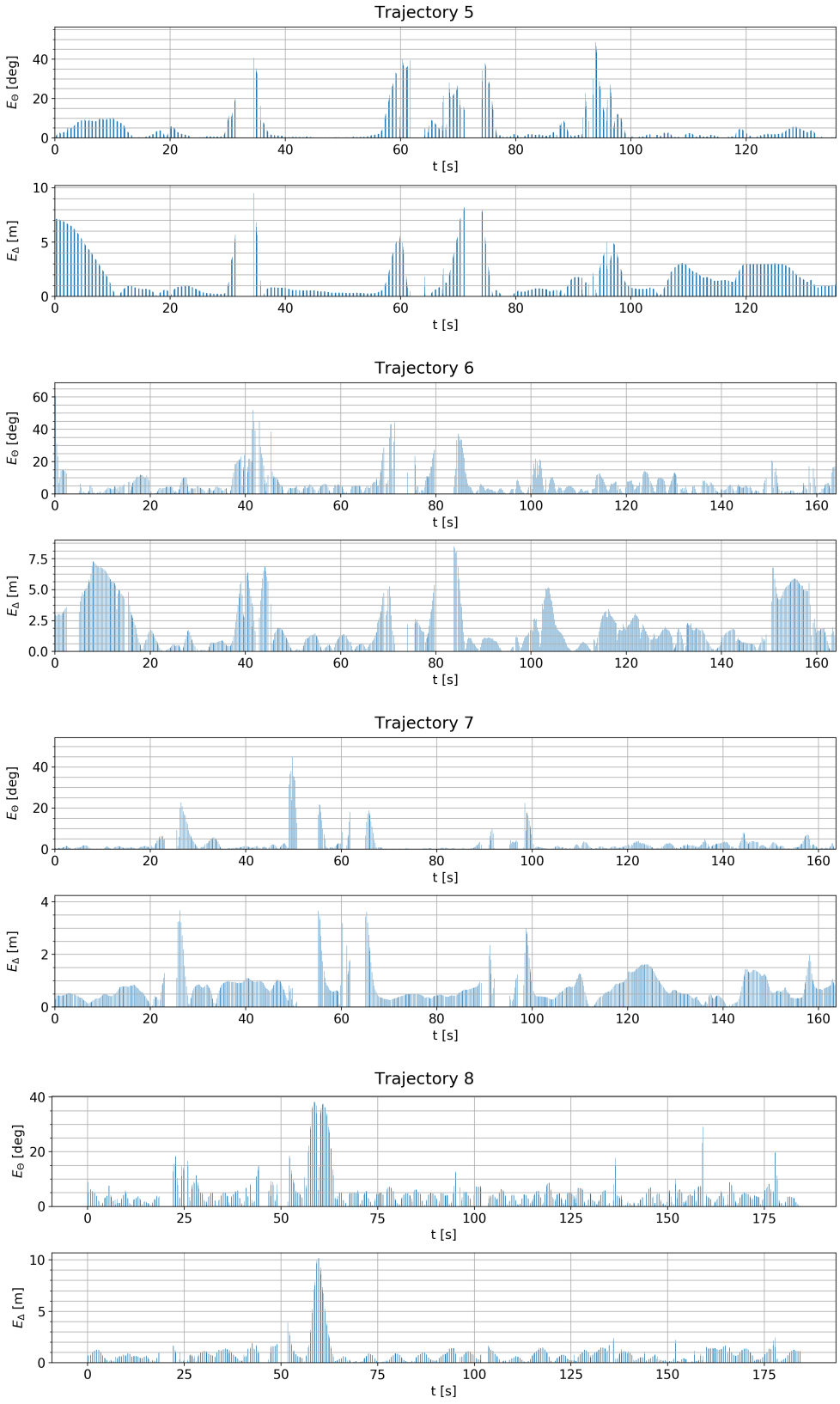


Figure 5.12: Absolute error of the estimation of heading and lateral displacement. (cont.)

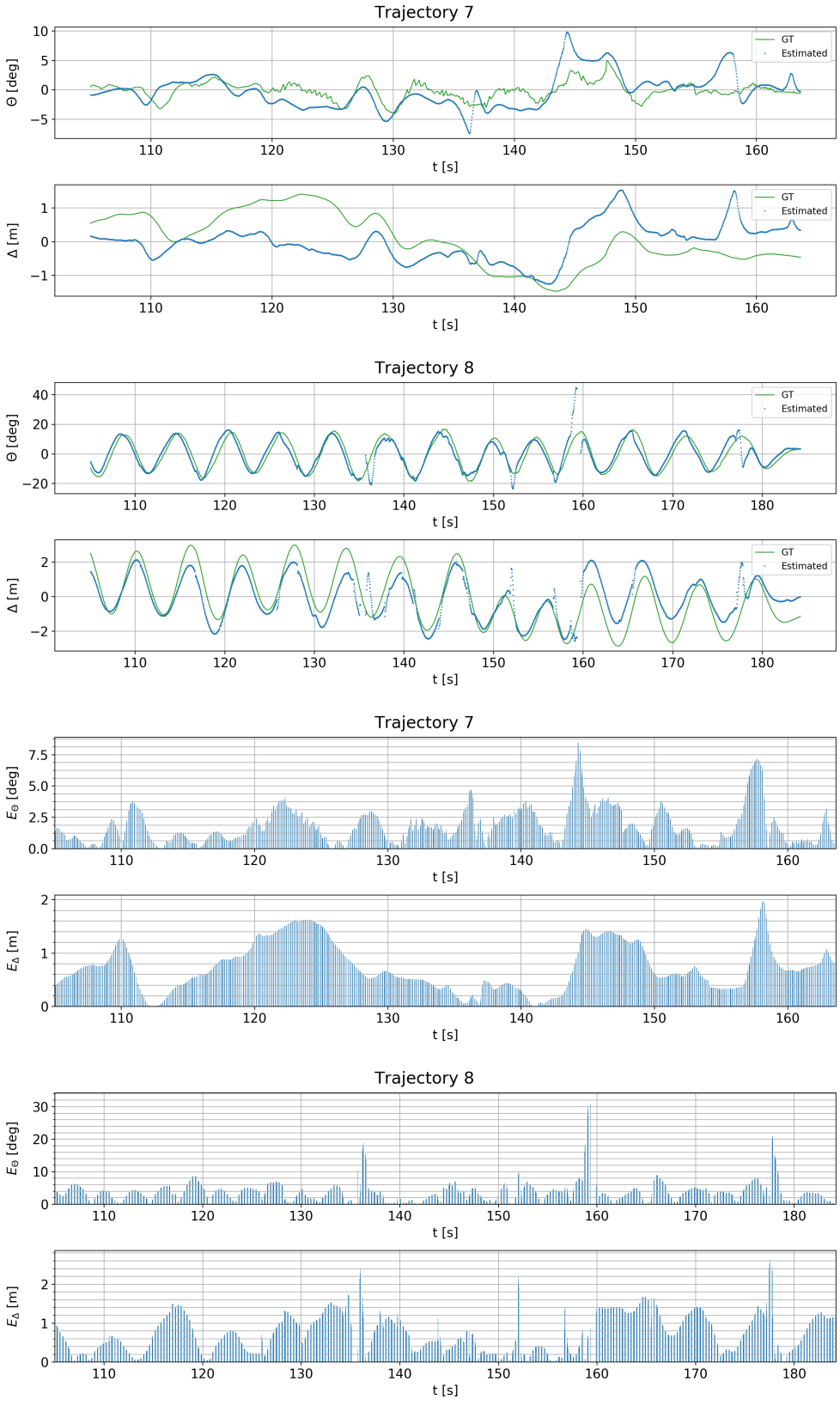


Figure 5.13: Estimated heading and lateral displacement for the evaluation on a restricted part of the track (top), and absolute error of the estimation (bottom).

	$MAE(\Theta)$ [deg]	$MAE(\Delta)$ [m]
Driving style: straight		
Trajectory 7 (restr.)	1.817	0.738
Driving style: oscillating		
Trajectory 8 (restr.)	3.799	0.743

Table 5.2: MAE for the estimation of heading and lateral displacement over restricted trajectories.

6

CONCLUSIONS AND FUTURE WORK

Lane following systems are categorized at the lowest of 5 levels of autonomy by the international standards, and are supposed to be one of the simplest building blocks to achieve full autonomy. However, as we started our research full of hope, the more we made our way into the field, and the more open space we found.

We began our work with the objective of studying the functioning of lane following systems and developing a solution to be coupled to control models already present in the literature. Moreover, we intended to do so with vision, as cameras are widely available and affordable, and at the same time are the only sensor able to precisely retrieve color and texture, important for our task.

To this end, we analyzed the state of the art and presented the details on the approaches adopted in the literature. With this step, we gained knowledge on the overall topic of lane following and on the most common ways of solving it. We noticed also here the first

discrepancies between different systems, as some of the works are more focused on the control part while others pursue the whole task. Also for this reason, the systems analyzed in the literature did not provide any clear architecture. We could instead notice a substantial lack of datasets for validation , together with general reluctance for real-world testing, as opposed to computer simulations.

We designed our perception system as composed of two stages. At first a custom designed line detector is adopted to retrieve the position and shape of the lateral lines. Then, a second component uses them to produce an estimate of position and shape of the road centerline, together with heading and lateral displacement of the vehicle.

In particular, we started the development of our line detection algorithm in accordance to the processing pipeline usually adopted, but soon performed several changes. Working on raw images, our first pipeline component oversees the extraction of features from the acquired frame. These are then postprocessed and finally a model is fit and tracked. Of particular relevance is our choice for the line model adopted, as we do not rely on Cartesian representations and recur instead to the Whewell intrinsic formulation. This is never seen in line detection, but is instead often adopted in control, for its good properties in modeling roads and trajectories. We can think of this choice as, in some sense, the missing link between the field of line detection and that of control.

With a method to detect the lateral lines and directly represent them in a convenient form, the subsequent stage is where the estimation of the parameters needed for control is made. This task represents a novel estimation technique for the centerline and guarantees more robustness to the overall estimation with the implementation of a Bayesian filter.

Completed the system, we evaluated its performance on real data, designing its tests to cover as many different driving scenarios as possible. Remarkably, our system is capable of maintaining good performances in most of the analyzed scenarios, producing reliable estimates of the centerline shape and position, together with the heading and lateral displacement of the vehicle. Furthermore, even more

important is the fact that we can obtain such results relying only on vision and vehicle odometry.

While this is clearly the most relevant result of our work, additional contributions can be found not only in the system developed but also in the approaches adopted . In particular, we demonstrate the possibility of linking perception and control, here through the specific line models adopted, while at the same time showing the competitiveness of this choice. The system realized satisfies then our original goal of unifying the perception task in lane following problems under a single system.

On a separate note, we point out that, in the process of creating this system, we also developed a fully functional line detector, capable of working also independently from the rest of the systme. Only when coupled with control problems however, its choice of the intrinsic models gives greater returns.

Given our positive results, it is however possible to identify several improvements for what concerns the computational load of the system and the dataset used for its validation. In particular, we consider above all a complete enhancement of the system to achieve real-time performances. This would require at first a porting of the main algorithms to C++, language more suited for real-time embedded platforms. Secondarily, several optimization could be performed on the overall implementation, regardless of the language adopted. Indeed, the architecture of the current system is designed to allow for fast prototyping of new solutions and variants, as it is often needed in a research environments, but it does so without considering the computational overhead introduced. Achieving a running frequency of 30 Hz, the system could then be integrated with a control software, realizing its final purpose. This configuration would at first allow us to test its performances in-the-loop and, subsequently, on our actual experimental vehicle.

With reference instead to the possible testing environments for lane following systems, testing on real data remains the best choice. The dataset we adopted however presented several difficulties. To begin, ti only provided 8 different recordings, often not enough to test multiple alternatives of the same algorithm without overfitting them.

Moreover, the generation of its ground truth is deliberately declared approximated, thus leading to systematic errors in the performance evaluations. Even more so, the possibility to test a new working algorithm on an isolated track is of course beneficial, but for the final deployment of this type of systems, eventually also highways-driving and finally urban-driving need to be tackled. In this context, it would then be largely beneficial for the research community the generation and diffusion of a customized dataset, grouping hundreds of different trajectories, recorded in different scenarios and roads.

At last, we would like to address some considerations to the set of sensors needed for the task. In parituclar, at the moment we are only using a monocular camera and the information from the wheels and steering encoders of the vehicle. Nevertheless, in some occasions, the data from the encoders could not be provided, and thus the algorithm would have to robustly adapt and cope with the lack of information, possibly failing. What could instead be done, is to resort to other components to generate the odometry. The most common methodology in this sense, given also our usage of a camera, is known as visual odometry. Implementing this technique in our work would allow us to rely only on images for the overall estimation, making the system portable to potentially any road vehicle.

BIBLIOGRAPHY

[1] SAE International. *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*. 2018. DOI: https://doi.org/10.4271/J3016_201806. URL: https://doi.org/10.4271/J3016_201806 (cit. on pp. 1, 3).

[2] Nidhi Kalra and Susan M Paddock. "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" In: *Transportation Research Part A: Policy and Practice* 94 (2016), pp. 182–193 (cit. on p. 2).

[3] Daniel J Fagnant and Kara Kockelman. "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations." In: *Transportation Research Part A: Policy and Practice* 77 (2015), pp. 167–181 (cit. on p. 2).

[4] NHTSA National Highway Traffic Safety Administration. *U.S. Department of Transportation 2012 Traffic Safety Facts FARS/GES Annual Report*. 2012. URL: https://crashstats.nhtsa.dot.gov/ (visited on 11/13/2019) (cit. on p. 2).

[5] Istat Istituto nazionale di statistica. *Incidenti stradali Anno 2018*. 2019. URL: https://www.istat.it/it/archivio/232366 (visited on 11/13/2019) (cit. on p. 2).

[6] Fábio Duarte and Carlo Ratti. "The impact of autonomous vehicles on cities: A review." In: *Journal of Urban Technology* 25.4 (2018), pp. 3–18 (cit. on p. 2).

[7] Wenwen Zhang, Subhrajit Guhathakurta, Jinqi Fang, and Ge Zhang. "Exploring the impact of shared autonomous vehicles on urban parking demand: An agent-based simulation approach." In: *Sustainable Cities and Society* 19 (2015), pp. 34–45 (cit. on p. 2).

[8] Chana J Haboucha, Robert Ishaq, and Yoram Shiftan. "User preferences regarding autonomous vehicles." In: *Transportation Research Part C: Emerging Technologies* 78 (2017), pp. 37–49 (cit. on p. 2).

[9] Saeed Asadi Bagloee, Madjid Tavana, Mohsen Asadi, and Tracey Oliver. "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies." In: *Journal of modern transportation* 24.4 (2016), pp. 284–303 (cit. on p. 2).

[10] Dean A Pomerleau. "Alvinn: An autonomous land vehicle in a neural network." In: *Advances in neural information processing systems*. 1989, pp. 305–313 (cit. on pp. 2, 24, 35).

[11] Guna Seetharaman, Arun Lakhotia, and Erik Philip Blasch. "Unmanned vehicles come of age: The DARPA grand challenge." In: *Computer* 39.12 (2006), pp. 26–29 (cit. on pp. 2, 3).

[12] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*. Vol. 56. springer, 2009 (cit. on p. 3).

[13] SAE International. *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*. 2014. DOI: https://doi.org/10.4271/J3016_201401. URL: https://doi.org/10.4271/J3016_201401 (cit. on p. 3).

[14] SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. 2016. DOI: https://doi.org/10.4271/J3016_201609. URL: https://doi.org/10.4271/J3016_201609 (cit. on p. 3).

[15] Christiaan Hetzner. "Audi, BMW, others frustrated by hurdles slowing launch of self-driving cars." In: *Automotive News Europe* (2019). URL: https://europe.autonews.com/automakers/audi-bmw-others-frustrated-hurdles-slowing-launch-self-driving-cars (visited on 11/13/2019) (cit. on p. 3).

[16] Sandipann P Narote, Pradnya N Bhujbal, Abbhilasha S Narote, and Dhiraj M Dhane. "A review of recent advances in lane detection and departure warning system." In: *Pattern Recognition* 73 (2018), pp. 216–234 (cit. on pp. 4, 10).

[17] Hui Zhou and Han Wang. "Vision-based lane detection and tracking for driver assistance systems: A survey." In: *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. IEEE. 2017, pp. 660–665 (cit. on pp. 4, 7, 9).

[18] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. "Recent progress in road and lane detection: a survey." In: *Machine vision and applications* 25.3 (2014), pp. 727–745 (cit. on pp. 7–10, 12, 13).

[19] Alberto Hata and Denis Wolf. "Road marking detection using LIDAR reflective intensity data and its application to vehicle localization." In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2014, pp. 584–589 (cit. on p. 7).

[20] Jörg Kibbel, Winfried Justus, and Kay Furstenberg. "Lane estimation and departure warning using multilayer laserscanner." In: *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.* IEEE. 2005, pp. 607–611 (cit. on p. 8).

[21] Philipp Lindner, Eric Richter, Gerd Wanielik, Kiyokazu Takagi, and Akira Isogai. "Multi-channel lidar processing for lane detection and estimation." In: *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2009, pp. 1–6 (cit. on p. 8).

[22] Albert S Huang, David Moore, Matthew Antone, Edwin Olson, and Seth Teller. "Finding multiple lanes in urban road networks with vision and lidar." In: *Autonomous Robots* 26.2-3 (2009), pp. 103–122 (cit. on pp. 8, 10, 13, 17, 18).

[23] Basel Fardi and Gerd Wanielik. "Hough transformation based approach for road border detection in infrared images." In: *IEEE Intelligent Vehicles Symposium, 2004*. IEEE. 2004, pp. 549–554 (cit. on p. 8).

[24] Jiang Ruyi, Klette Reinhard, Vaudrey Tobi, and Wang Shigang. "Lane detection and tracking using a new lane model and distance transform." In: *Machine vision and applications* 22.4 (2011), pp. 721–737 (cit. on pp. 9, 11, 13, 14, 19, 20).

[25] Yong Ding, Zheng Xu, Yubin Zhang, and Ke Sun. "Fast lane detection based on bird's eye view and improved random sample consensus algorithm." In: *Multimedia Tools and Applications* 76.21 (2017), pp. 22979–22998 (cit. on pp. 10, 17).

[26] Jihun Kim and Minho Lee. "Robust lane detection based on convolutional neural network and random sample consensus." In: *International conference on neural information processing*. Springer. 2014, pp. 454–461 (cit. on pp. 10, 15, 18).

[27] U Seger. "HDR imaging in automotive applications." In: *High Dynamic Range Video*. Elsevier, 2016, pp. 477–498 (cit. on p. 10).

[28] Xiangjing An, Erke Shang, Jinze Song, Jian Li, and Hangen He. "Real-time lane departure warning system based on a single FPGA." In: *EURASIP Journal on Image and Video Processing* 2013.1 (2013), p. 38 (cit. on p. 10).

[29] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002 (cit. on pp. 10, 13, 18).

[30] Christopher Rasmussen. "RoadCompass: following rural roads with vision+ ladar using vanishing point tracking." In: *Autonomous Robots* 25.3 (2008), pp. 205–229 (cit. on p. 11).

[31] Tsung-Ying Sun, Shang-Jeng Tsai, and Vincent Chan. "HSI color model based lane-marking detection." In: *2006 IEEE Intelligent Transportation Systems Conference*. IEEE. 2006, pp. 1168–1172 (cit. on p. 11).

[32] Jongin Son, Hunjae Yoo, Sanghoon Kim, and Kwanghoon Sohn. "Real-time illumination invariant lane detection for lane departure warning system." In: *Expert Systems with Applications* 42.4 (2015), pp. 1816–1824 (cit. on pp. 11, 14).

[33] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003 (cit. on p. 11).

[34] Ivan S Kholopov. "Bird's Eye View Transformation Technique in Photogrammetric Problem of Object Size Measuring at Low-altitude Photography." In: *International Conference" Actual Issues of Mechanical Engineering" 2017 (AIME 2017)*. Atlantis Press. 2017 (cit. on p. 11).

[35] Anuar Mikdad Muad, Aini Hussain, Salina Abdul Samad, Mohd Marzuki Mustaffa, and BURHANUDDIN YEOP Majlis. "Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system." In: *2004 IEEE Region 10 Conference TENCON 2004.* IEEE. 2004, pp. 207–210 (cit. on p. 11).

[36] Kun Zhao, Mirko Meuter, Christian Nunn, Dennis Müller, Stefan Müller-Schneiders, and Josef Pauli. "A novel multi-lane detection and tracking system." In: *2012 IEEE Intelligent Vehicles Symposium*. IEEE. 2012, pp. 1084–1089 (cit. on pp. 11, 17, 19).

[37] Ruyi Jiang, Reinhard Klette, Tobi Vaudrey, and Shigang Wang. "New lane model and distance transform for lane detection and tracking." In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2009, pp. 1044–1052 (cit. on p. 11).

[38] Marcos Nieto, Jon Arróspide Laborda, and Luis Salgado. "Road environment modeling using robust perspective analysis and recursive Bayesian segmentation." In: *Machine Vision and Applications* 22.6 (2011), pp. 927–945 (cit. on p. 11).

[39] Seung-Nam Kang, Soomok Lee, Junhwa Hur, and Seung-Woo Seo. "Multi-lane detection based on accurate geometric lane estimation in highway scenarios." In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE. 2014, pp. 221–226 (cit. on p. 11).

[40] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. "Towards end-to-end lane detection: an instance segmentation approach." In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 286–291 (cit. on pp. 11, 16, 20, 21).

[41] Yang Yang Ye, Xiao Li Hao, and Hou Jin Chen. "Lane detection method based on lane structural analysis and CNNs." In: *IET Intelligent Transport Systems* 12.6 (2018), pp. 513–520 (cit. on p. 12).

[42] Shengyan Zhou, Yanhua Jiang, Junqiang Xi, Jianwei Gong, Guangming Xiong, and Huiyan Chen. "A novel lane detection based on geometrical model and gabor filter." In: *2010 IEEE Intelligent Vehicles Symposium*. IEEE. 2010, pp. 59–64 (cit. on pp. 11, 18).

[43] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. "Lane detection using B-snake." In: *Proceedings 1999 International Conference on Information Intelligence and Systems (Cat. No. PR00446)*. IEEE. 1999, pp. 438–443 (cit. on pp. 11, 17).

[44] King Hann Lim, Kah Phooi Seng, Li-Minn Ang, and Siew Wen Chin. "Lane detection and Kalman-based linear-parabolic lane tracking." In: *2009 International Conference on Intelligent Human-*

Machine Systems and Cybernetics. Vol. 2. IEEE. 2009, pp. 351–354 (cit. on pp. 11, 13, 16, 19).

[45] Yue Wang, Dinggang Shen, and Eam Khwang Teoh. "Lane detection using spline model." In: *Pattern Recognition Letters* 21.8 (2000), pp. 677–689 (cit. on pp. 11, 13).

[46] Shinq-Jen Wu, Hsin-Han Chiang, Jau-Woei Perng, Chao-Jung Chen, Bing-Fei Wu, Tsu-Tian Lee, et al. "The heterogeneous systems integration design and implementation for lane keeping on a vehicle." In: *IEEE Transactions on Intelligent Transportation Systems* 9.2 (2008), pp. 246–263 (cit. on p. 12).

[47] Pei-Yung Hsiao, Chun-Wei Yeh, Shih-Shinh Huang, and Li-Chen Fu. "A portable vision-based real-time lane departure warning system: day and night." In: *IEEE Transactions on Vehicular Technology* 58.4 (2008), pp. 2089–2094 (cit. on p. 12).

[48] Abdelhamid Mammeri, Azzedine Boukerche, and Zongzhi Tang. "A real-time lane marking localization, tracking and communication system." In: *Computer Communications* 73 (2016), pp. 132–143 (cit. on p. 12).

[49] Massimo Bertozzi and Alberto Broggi. "GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection." In: *IEEE transactions on image processing* 7.1 (1998), pp. 62–81 (cit. on p. 13).

[50] Joel C McCall and Mohan M Trivedi. "An integrated, robust approach to lane marking detection and lane tracking." In: *IEEE Intelligent Vehicles Symposium, 2004*. IEEE. 2004, pp. 533–537 (cit. on p. 13).

[51] Hao Li and Fawzi Nashashibi. "Robust real-time lane detection based on lane mark segment features and general a priori knowledge." In: *2011 IEEE International Conference on Robotics and Biomimetics*. IEEE. 2011, pp. 812–817 (cit. on p. 14).

[52] Juan Pablo Gonzalez and Umit Ozguner. "Lane detection using histogram-based segmentation and decision trees." In: *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 00TH8493)*. IEEE. 2000, pp. 346–351 (cit. on pp. 14, 18).

[53] ZuWhan Kim. "Robust lane detection and tracking in challenging scenarios." In: (2008) (cit. on pp. 14, 15, 17, 18, 20).

[54] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. "An empirical evaluation of deep learning on highway driving." In: *arXiv preprint arXiv:1504.01716* (2015) (cit. on p. 15).

[55] Bei He, Rui Ai, Yang Yan, and Xianpeng Lang. "Accurate and robust lane detection based on dual-view convolutional neutral network." In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2016, pp. 1041–1046 (cit. on p. 15).

[56] Ping-Rong Chen, Shao-Yuan Lo, Hsueh-Ming Hang, Sheng-Wei Chan, and Jing-Jhih Lin. "Efficient Road Lane Marking Detection with Deep Learning." In: *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*. IEEE. 2018, pp. 1–5 (cit. on pp. 15, 16).

[57] Ben Southall and Camillo J Taylor. "Stochastic road shape estimation." In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 1. IEEE. 2001, pp. 205–212 (cit. on p. 15).

[58] Amol Borkar, Monson Hayes, and Mark T Smith. "Robust lane detection and tracking with ransac and kalman filter." In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2009, pp. 3261–3264 (cit. on pp. 15, 18, 19).

[59] Junhwa Hur, Seung-Nam Kang, and Seung-Woo Seo. "Multi-lane detection in urban driving environments using conditional random fields." In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2013, pp. 1297–1302 (cit. on p. 16).

[60] Vijay Gaikwad and Shashikant Lokhande. "Lane departure identification for advanced driver assistance." In: *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2014), pp. 910–918 (cit. on p. 17).

[61] Jing-Fu Liu, Jui-Hung Wu, and Yi-Feng Su. "Development of an interactive lane keeping control system for vehicle." In: *2007 IEEE Vehicle Power and Propulsion Conference*. IEEE. 2007, pp. 702–706 (cit. on pp. 17, 25, 27, 32).

[62] Soonhong Jung, Junsic Youn, and Sanghoon Sull. "Efficient lane detection based on spatiotemporal images." In: *IEEE Transactions on Intelligent Transportation Systems* 17.1 (2015), pp. 289–295 (cit. on p. 17).

[63] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. "Lane detection and tracking using B-Snake." In: *Image and Vision computing* 22.4 (2004), pp. 269–280 (cit. on p. 17).

[64] Stefan K Gehrig, Axel Gern, Stefan Heinrich, and Bernd Woltermann. "Lane recognition on poorly structured roads-the bots dot problem in California." In: *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*. IEEE. 2002, pp. 67–71 (cit. on p. 17).

[65] Joel C McCall and Mohan Manubhai Trivedi. "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation." In: (2006) (cit. on pp. 18, 19).

[66] Raphaël Labayrade, Jerome Douret, Jean Laneurit, and Roland Chapuis. "A reliable and robust lane detection system based on the parallel use of three algorithms for driving safety assistance." In: *IEICE transactions on information and systems* 89.7 (2006), pp. 2092–2100 (cit. on pp. 18, 19).

[67] Seokjun Kang and Dong Seog Han. "Traffic Lane Estimation using Road Width Information." In: *2017 IEEE 7th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE. 2017, pp. 53–54 (cit. on p. 18).

[68] Jung Gap Kuk, Jae Hyun An, Hoyong Ki, and Nam Ik Cho. "Fast lane detection & tracking based on Hough transform with reduced memory requirement." In: *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2010, pp. 1344–1349 (cit. on p. 18).

[69] Qing Li, Nanning Zheng, and Hong Cheng. "Springrobot: A prototype autonomous vehicle and its algorithms for lane detection." In: *IEEE Transactions on Intelligent Transportation Systems* 5.4 (2004), pp. 300–308 (cit. on p. 18).

[70] Christian Lipski, Bjorn Scholz, Kai Berger, Christian Linz, Timo Stich, and Marcus Magnor. "A fast and robust approach to lane marking detection and lane tracking." In: *2008 IEEE Southwest Symposium on Image Analysis and Interpretation*. IEEE. 2008, pp. 57–60 (cit. on p. 18).

[71] Pei-Chen Wu, Chin-Yu Chang, and Chang Hong Lin. "Lane-mark extraction for automobiles under complex conditions." In: *Pattern Recognition* 47.8 (2014), pp. 2756–2767 (cit. on p. 19).

[72] Min Tian, Fuqiang Liu, and Zhencheng Hu. "Single camera 3D lane detection and tracking based on EKF for urban intelligent vehicle." In: *2006 IEEE International conference on vehicular electronics and safety*. IEEE. 2006, pp. 413–418 (cit. on p. 19).

[73] Yong Zhou, Rong Xu, Xiaofeng Hu, and Qingtai Ye. "A robust lane detection and tracking method based on computer vision." In: *Measurement science and technology* 17.4 (2006), p. 736 (cit. on p. 20).

[74] Vijay John, Zheng Liu, Seiichi Mita, Chunzhao Guo, and Kiyosumi Kidono. "Real-time road surface and semantic lane estimation using deep features." In: *Signal, Image and Video Processing* 12.6 (2018), pp. 1133–1140 (cit. on p. 20).

[75] Philip Koopman and Michael Wagner. "Autonomous vehicle safety: An interdisciplinary challenge." In: *IEEE Intelligent Transportation Systems Magazine* 9.1 (2017), pp. 90–96 (cit. on p. 21).

[76] Sadayuki Tsugawa. "Vision-based vehicles in Japan: Machine vision systems and driving control systems." In: *IEEE Transactions on industrial electronics* 41.4 (1994), pp. 398–405 (cit. on p. 23).

[77] Alireza Khodayari, Ali Ghaffari, Sina Ameli, and Jamal Flahatgar. "A historical review on lateral and longitudinal control of autonomous vehicle motions." In: *2010 International Conference on Mechanical and Electrical Technology*. IEEE. 2010, pp. 421–429 (cit. on p. 23).

[78] Richard S Wallace, Anthony Stentz, Charles E Thorpe, Hans P Moravec, William Whittaker, and Takeo Kanade. "First Results in Robot Road-Following." In: *IJCAI*. Citeseer. 1985, pp. 1089–1095 (cit. on p. 24).

[79] Matthew Turk, DAVIDG Morgenthaler, Keith Gremban, and Martin Marra. "Video road-following for the autonomous land vehicle." In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. IEEE. 1987, pp. 273–280 (cit. on p. 24).

[80] Darwin Kuan and UMAKANT Sharma. "Model based geometric reasoning for autonomous road following." In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. IEEE. 1987, pp. 416–423 (cit. on p. 24).

[81] Jürgen Manigel and W Leonhard. "Vehicle control by computer vision." In: *IEEE Transactions on industrial electronics* 39.3 (1992), pp. 181–188 (cit. on p. 24).

[82] Bakhtiar B Litkouhi, Allan Y Lee, and Douglas B Craig. "Estimator and controller design for lanetrak, a vision-based automatic vehicle steering system." In: *Proceedings of 32nd IEEE Conference on Decision and Control*. IEEE. 1993, pp. 1868–1873 (cit. on p. 24).

[83] Ju Yong Choi, Seong Jae Hong, Kyoung Taik Park, Wan Suk Yoo, and Man Hyung Lee. "Lateral control of autonomous vehicle by yaw rate feedback." In: *KSME international journal* 16.3 (2002), pp. 338–343 (cit. on p. 24).

[84] Dean Pomerleau and Todd Jochem. "Rapidly adapting machine vision for automated vehicle steering." In: *IEEE expert* 11.2 (1996), pp. 19–27 (cit. on pp. 24, 26).

[85] Sadayuki Tsugawa, Hiroaki Mori, and Shin Kato. "A lateral control algorithm for vision-based vehicles with a moving target in the field of view." In: *in the Field of View," in IEEE International Conference on Intelligent Vehicles*. Citeseer. 1998 (cit. on p. 24).

[86] Jing-ming Zhang and Dian-bo Ren. "Lateral control of vehicle for lane keeping in intelligent transportation systems." In: *2009 International Conference on Intelligent Human-Machine Systems and Cybernetics*. Vol. 1. IEEE. 2009, pp. 446–450 (cit. on p. 25).

[87] Soichi Ibaraki, Shashikanth Suryanarayanan, and Masayoshi Tomizuka. "Design of Luenberger state observers using fixed-structure H/sub/spl infin//optimization and its application to fault detection in lane-keeping control of automated vehicles." In: *IEEE/ASME Transactions on Mechatronics* 10.1 (2005), pp. 34–42 (cit. on p. 25).

[88] Lars Hammarstrand, Maryam Fatemi, Ángel F García-Fernández, and Lennart Svensson. "Long-range road geometry estimation using moving vehicles and roadside observations." In: *IEEE Transactions on Intelligent Transportation Systems* 17.8 (2016), pp. 2144–2158 (cit. on pp. 25, 27, 31).

[89] *Whewell equation*. 2019. URL: `https://en.wikipedia.org/wiki/Whewell_equation` (visited on 11/25/2019) (cit. on p. 30).

[90] Sung Gu Yi, Chang Mook Kang, Seung-Hi Lee, and Chung Choo Chung. "Vehicle trajectory prediction for adaptive cruise control." In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2015, pp. 59–64 (cit. on p. 31).

[91] Xinyu Wang, Mengyin Fu, Hongbin Ma, and Yi Yang. "Lateral control of autonomous vehicles based on fuzzy logic." In: *Control Engineering Practice* 34 (2015), pp. 1–17 (cit. on pp. 31, 34).

[92] Stefano Arrigoni, Edoardo Trabalzini, Mattia Bersani, Francesco Braghin, and Federico Cheli. "Non-linear mpc motion planner for autonomous vehicles based on accelerated particle swarm optimization algorithm." In: *2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*. IEEE. 2019, pp. 1–6 (cit. on pp. 33–35, 57).

[93] Sagar Behere and Martin Törngren. "A functional reference architecture for autonomous driving." In: *Information and Software Technology* 73 (2016), pp. 136–150 (cit. on pp. 33, 34).

[94] Salim Chaib, Mariana S Netto, and Said Mammar. "H/sub/spl infin//, adaptive, PID and fuzzy control: a comparison of controllers for vehicle lane keeping." In: *IEEE Intelligent Vehicles Symposium, 2004*. IEEE. 2004, pp. 139–144 (cit. on p. 34).

[95] Joshué Pérez, Vicente Milanés, and Enrique Onieva. "Cascade architecture for lateral control in autonomous vehicles." In: *IEEE Transactions on Intelligent Transportation Systems* 12.1 (2011), pp. 73–82 (cit. on p. 34).

[96] Miguel Angel Sotelo. "Lateral control strategy for autonomous steering of Ackerman-like vehicles." In: *Robotics and Autonomous Systems* 45.3-4 (2003), pp. 223–233 (cit. on p. 34).

[97] Xiaohui Li, Zhenping Sun, Dongpu Cao, Zhen He, and Qi Zhu. "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications." In: *IEEE/ASME Transactions on Mechatronics* 21.2 (2015), pp. 740–753 (cit. on p. 34).

[98] Zhilu Chen and Xinming Huang. "End-to-end learning for lane keeping of self-driving cars." In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1856–1860 (cit. on p. 35).

[99] Eder Santana and George Hotz. "Learning a driving simulator." In: *arXiv preprint arXiv:1608.01230* (2016) (cit. on p. 35).

[100] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. "End to end learning for self-driving cars." In: *arXiv preprint arXiv:1604.07316* (2016) (cit. on pp. 36, 37).

[101] Y Lecun, E Cosatto, J Ben, U Muller, and B Flepp. "Dave: Autonomous off-road vehicle control using end-to-end learning." In: *DARPA-IPTO Final Report* (2004). URL: `http://net-scale.com/doc/net-scale-dave-report.pdf` (cit. on p. 36).

[102] Hao Yu, Shu Yang, Weihao Gu, and Shaoyu Zhang. "Baidu driving dataset and end-to-end reactive control model." In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 341–346 (cit. on p. 36).

[103] *Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary*. Standard. Geneva, CH: International Organization for Standardization, Dec. 2011 (cit. on p. 42).

[104] Nick Schneider and Marius Cordts. *Cityscapes Calibration*. 2016. URL: `https://github.com/mcordts/cityscapesScripts/blob/master/docs/csCalibration.pdf` (visited on 04/30/2019) (cit. on p. 43).

[105] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 (cit. on p. 44).

[106] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. "Bdd100k: A diverse driving video database with scalable annotation tooling." In: *arXiv preprint arXiv:1805.04687* (2018) (cit. on p. 44).

[107] Costantino Grana, Daniele Borghesani, and Rita Cucchiara. "Optimized block-based connected components labeling with decision trees." In: *IEEE Transactions on Image Processing* 19.6 (2010), pp. 1596–1609 (cit. on p. 53).

[108] Ryan Tibshirani. *Advanced Methods for Data Analysis (36-402/36-608), Lecture notes: Smoothing Splines*. 2014. URL: https://www.stat.cmu.edu/~ryantibs/advmethods/notes/smoothspline.pdf (visited on 11/28/2019) (cit. on p. 55).

[109] Marco C Campi. "Exponentially weighted least squares identification of time-varying systems with white disturbances." In: *IEEE Transactions on Signal Processing* 42.11 (1994), pp. 2906–2914 (cit. on p. 62).