A STUDY ON LANE DETECTION METHODS FOR AUTONOMOUS DRIVING

A STUDY ON LANE DETECTION METHODS FOR AUTONOMOUS DRIVING

BY

PAOLO CUDRANO, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2019) McMaster University

(Computing and Software) Hamilton, Ontario, Canada

TITLE:	A study on lane detection methods for autonomous driving
AUTHOR:	Paolo Cudrano B.Sc., (Engineering of Computing Systems) Politecnico di Milano, Milan, Italy
SUPERVISOR:	Dr. Martin von Mohrenschildt
NUMBER OF PAGES:	ix, 136

To Mom and Dad.

Abstract

Machine perception is a key element for the research on autonomous driving vehicles. In particular, we focus on the problem of lane detection with a single camera. Many lane detection systems have been developed and many algorithms have been published over the years. However, while they are already commercially available to deliver lane departure warnings, their reliability is still unsatisfactory for fully autonomous scenarios.

In this work, we questioned the reasons for such limitations. After examining the state of the art and the relevant literature, we identified the key methodologies adopted. We present a self-standing discussion of bird's eye view (BEV) warping and common image preprocessing techniques, followed by gradient-based and color-based feature extraction and selection. Line fitting algorithms are then described, including least squares methods, Hough transform and random sample consensus (RANSAC). Polynomial and spline models are considered. As a result, a general processing pipeline emerged. We further analyzed each key technique by implementing it and performing experiments using data we previously collected. At the end of our evaluation, we designed and developed an overall system, finally studying its behavior.

This analysis allowed us on one hand to gain insight into the reasons holding back present systems, and on the other to propose future developments in those directions.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Martin von Mohrenschildt, for his guidance in my academic path at McMaster University. He was always available whenever I ran into a problem in my research or had a question writing this thesis. He has been a mentor in both my academic and personal life, supporting and motivating me in spite of any difficulty I found on my path. I was very lucky to have him as my supervisor.

I would also like to thank Dr. Saeid Habibi and the CMHT staff for welcoming me as part of their team and providing me with the technical support I needed to conduct my research.

An acknowledgment goes moreover to my supervisor at Politecnico di Milano, Dr. Matteo Matteucci, for his precious research advices during our correspondence.

My gratitude goes furthermore to the members of my Examination Committee, Dr. Mark Lawford and Dr. Hassan Z. Ashtiani, for their constructive feedback on my work.

I would also like to thank all the friends I have made in McMaster University, for helping me when needed and having fun together. A special thanks goes to Afroja Parvin for her support, first, during my time in Canada, and then, once back in Italy, and for constantly motivating me throughout this journey.

Thanks to all my lab mates at CMHT and in particular to my buddies Ahmed Doghri and Ben Miethig for all the nights and days spent working together. Thanks to Ben also for introducing me to the Canadian culture. Your presence, guys, has made my experience way better.

In conclusion, last but by no means least, I'm grateful to my beloved mother, for supporting me during my studies, in Italy and abroad, and for raising me the way she did, regardless of all the difficulties that came into our way. A final thanks goes to my late father, whose excitement for my experience abroad motivated me to this achievement.

Contents

Chapter 1

Introduction

Self-driving vehicles, autonomous road navigation and advanced driver-assistance systems are nowadays key topics in research and product development. In this framework, many interesting challenges are presented to computer scientists and engineers working in machine perception.

In order for a vehicle to autonomously navigate the road, one of its primary abilities must be to successfully orient itself on it. A multitude of vehicles and pedestrians share nowadays the road, according to rules and customs most of the times taken for granted, such as the presence of lane demarcations or the functioning of traffic lights. All these elements are however essential for guaranteeing a harmonious usage of the road and the safety of each user. In this context, knowing where each line marking is located on the pavement becomes a strong necessity for safely driving in presence of other vehicles and in compliance with the traffic regulations.

In this thesis, we focus exactly on the problem of lane detection. We employ a single camera and aim to identify the road markings in front of a vehicle. A considerable amount of research has been carried out on this problem and the literature presents

many approaches to reach this objective [1, 2, 3]. However, it is still by no means clear how to build a well functioning lane detection system. Whereas indeed lane detection technologies are already commercially available for lane departure warning systems, the requirements on their reliability for fully autonomous vehicles are not yet met.

For this reason, we propose with this work to better investigate the issue and understand what are the elements making this task so arduous. In our study we will analyze the methodologies currently applied by others, while developing an end-to-end system ourselves. We will use such system to devise and perform experiments on different techniques in the context of this work, while setting the ground for potential future extensions.

We approach the problem with the following steps.

1. Identify the common structures adopted in monocular lane detection systems, outlining a general processing pipeline.
2. Research the literature to identify the key processing algorithms.
3. Implement the key algorithms and test them on our own data.
4. With the experience gained, propose and implement an overall system for lane detection.

Specifically, our analysis begins in Chapter 2, with a presentation of the general framework in which the problem is collocated and a review of the literature on the generic processing pipeline for lane detection systems using vision. For each stage composing it, different implementation choices are reported, along with their sources.

What follows, in Chapter 3, is a collection of experimental reports, created throughout our investigation and representing the core of our study. In each of them, we focus our attention on a specific aspect of the system or of its implementation, analyzing the associated literature, describing the methodology adopted and deriving the appropriate conclusions from the results obtained. As a consequence of this sharp subdivision, not only the reports can be read continuously, following the structure of this chapter, but they can also be considered independently, to analyze in depth, for example, only one of the treated problems.

As described in the subsequent Chapter 4, it is exactly by working on these experiments and on their careful analysis that our complete system was incrementally built. With the intention of presenting its construction, we begin this chapter discussing the implementation choices we made at each stage of the processing pipeline. We then proceed focusing on its technological requirements, both on a hardware and software level, and finally conclude with an analysis of the results produced.

At the end, Chapter 5 draws our final conclusions on lane detection algorithms and particularly on the difficulties they have to tackle, presenting at the same time some of the alternatives that can be implemented in the future as extensions or improvements of our work.

Chapter 2

Processing Pipeline and Previous Work

Taking into consideration the literature on computer vision systems for lane detection, lane departure warning and road detection, some common paradigms can be observed. As highlighted in [1, 2], the considered algorithms share the overall traits of a common pipeline. As depicted in Figure 2.1, we then expect such complex systems, employing one or more cameras and occasionally supported by LiDAR, GPS, IMU or vehicle OBD, to be composed of the following four macro-components:

1. Preprocessing
2. Feature extraction
3. Model fitting
4. Tracking

These stages are primarily applied to the images obtained from the cameras. The information from possible additional sensors is instead considered as accessory,

although it can be important for the robustness of the final system. Nevertheless, in our work we don't treat such integration and we focus instead only on monocular systems.

With the above in mind, in the rest of the chapter we analyze the four mentioned components, studying for each of them how they can be introduced in our work.

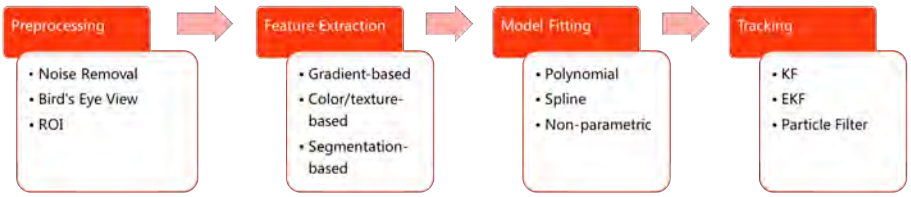


Figure 2.1: Illustration of the computer vision pipeline adopted in the literature for the lane detection problem using monocular computer vision. The four main stages are highlighted, each with a list of examples of the tasks performed in such stage.

2.1 Preprocessing

As a first step after collecting a frame, it is necessary to clean and transform the image before we can proceed to extract relevant features. In this stage, several different operators are applied, as a function of the next ones.

Probably the most common of the transformations applied is the removal of noise. According to the type of scene observed and of sensor used, the image could present noise of different nature. The two most common examples are Gaussian noise and salt-and-pepper noise. In the most basic approach, they can be tackled respectively by means of Gaussian blur and median filter. Nevertheless, more modern techniques such as non-local means filtering, Gaussian scale mixtures and several non-linear filters are also often adopted [4, 5].

Another well spread preprocessing technique for lane detection systems, and road-related systems in general, is the rectification of the image by means of perspective warping, obtaining the so-called Bird's Eye View (BEV). As sketched in Figure 2.2, this consists in applying a projective transformation to the front-view image in order to virtually move its point of view perpendicularly above the scene [6, 7, 8]. In this way, several road features are enhanced and their geometrical properties are restored. On a stereo-vision system, this process is trivial and commonly taken for granted. For monocular camera systems instead, as in our case, a precise calibration of the sensor is required and significantly important for the effectiveness of the transformation.

Sometimes based on the above BEV, and some other times derived from more complex parameters, the selection of a Region of Interest (ROI) is often beneficial. By removing part of the image, this technique shifts the attention to a specific area. This can reduce the computational needs of the system and contemporary remove undesired features, preventing them from leading to false detections in the successive stages.

As simple as it might seem, the image is also often downsized to reduce the computational requirements as well as further attenuate noise artifacts. A trade-off in this case must be found, however, in order to preserve the important features for the subsequent analysis.

A common issue in real world images is also the variability of illumination conditions, influenced by the weather, the time of the day, and the presence of shaded regions on the road. Since some algorithms are based on color properties of the image, this can be an important aspect to consider. We must point out however that a full adaptability to such changes is still an open problem in the field.

Lastly, in some cases the detection and tracking of the vanishing points in the scene, and in particular of the horizon line [2], are pursued as a further preprocessing operation. Authors as in [9, 10, 11] find this result to be a useful support for complementary tasks, as the definition of their ROI or the adaptive re-calibration of their cameras. In other cases instead [12, 13], the vanishing points can even become fundamental for the line fitting.

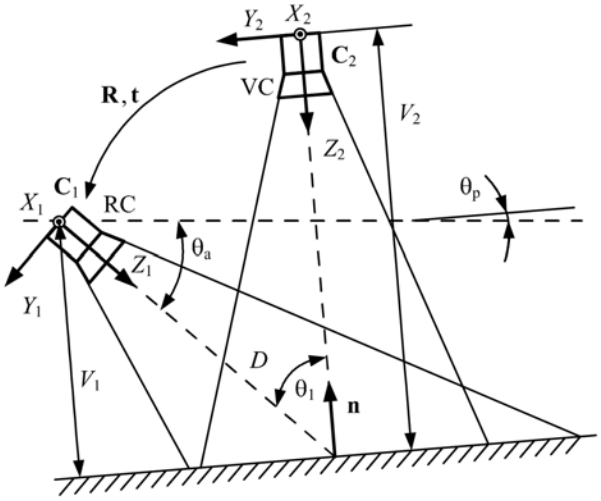


Figure 2.2: Illustration of the Bird's Eye View transformation, where the front-view camera C_1 is reprojected perpendicularly to the road in C_2, in order to capture the same view from above. Adapted from [8, Figure 2].

2.2 Feature extraction

After preparing the image, it is important to identify the relevant information it contains and extract it in the form of *features*. However, according to the system analyzed, the nature of the extraction techniques can vary significantly [14]. Moreover, their adoption is strictly linked to the implementation of the subsequent stages.

In general, three categories of feature extraction techniques can be identified, each based respectively on gradient features, color features and segmentation. They are briefly explored below. Notice that such techniques are not necessarily mutually exclusive and that, occasionally, combination of multiple techniques are adopted. However, methods for coherently merging their outputs are still to be found, as discussed in [15].

2.2.1 Gradient-based

Gradient-based techniques are very popular in the literature. They rely on the assumption that lane markings are characterized by strong and visible edges in the image. They are usually computed in the grayscale space and exploit the sharp color discontinuities between road lines and pavement. Standard edge detectors such as the Sobel and Canny operators are often considered for their fast execution [16, 17]. Nevertheless, they require an accurate tuning of their parameters, which often undermines their adaptability to different illuminations and weather conditions. Their lack of robustness to noise is sometimes attenuated by steerable filters [18], where the orientation of each line is taken into account to reduce the amount of false detections. A more complex algorithm is finally the Line Segment Detector (LSD) [19], often used also in visual perception systems (e.g. for the estimation of vanishing points). This technique detects each line segment with a specific local contours analysis, resulting in increased robustness, even though at the price of a slightly lower computational efficiency.

2.2.2 Color-based

Although often underestimated in other applications, color is an extremely valuable feature for the detection of lane marking. In fact, as humans, we define and distinguish lane markings just because of their strong color discontinuity from the rest of the road pavement. Moreover, at times their color is even carrier of additional information to the driver, such as for the distinction between yellow and white lines in North America [20, 21].

A first attempt in isolating color features is the application of a sharp threshold in a given color space. However, the variability of the illumination conditions induces a strong fluctuation of the color spectrum captured by the camera. For this reason, although the actual color of the markings is usually strictly constrained by regulations, fixed thresholding methods don't present the necessary degree of generalization. For achieving such results, an adaptive mechanism must be designed, as shown in [9]. No standard methodology is however widespread.

2.2.3 Segmentation-based

Segmentation is the process of identifying different regions in an image according to a similarity criteria. In autonomous driving systems, they are often employed to divide a recorded road scene into different sub-scenes to be further analyzed.

In our context, this technique could be beneficial to isolate the pavement from the rest of the scene [22, 23]. It can also be useful as an auxiliary mean of feature selection, for example by filtering out false positives not belonging to the road [24]. Furthermore, attempts with this technique have also been made to directly isolate the lane markings [25].

Image segmentation is usually performed applying clustering algorithms to a feature space derived from the image. Such space could be based only on the pixel color or it could also contain additional information on the pixel position. Depending on the specific goal, it could moreover be further elaborated [23]. Examples of commonly adopted clustering algorithms are k-means [23] and mean shift [26]. Significant effort has been put also in the application of convolutional neural networks for this task [17, 27, 28].

2.3 Model fitting

The core objective of lane detection systems is to describe the road markings with a model, as for example determining their position and shape on the road. This is the objective of the model fitting stage.

Different models and fitting techniques are available, according to the features used and the output required.

A first coarse characterization, as highlighted in [2], considers whether the markings are described with single polynomial lines (*parametric*), piecewise-defined polynomials (*semi-parametric*) or lines even less constrained and non necessarily smooth (*non-parametric*). While the latter are rarely adopted, polynomials and splines are widely used. Notice moreover that non-parametric models often rely on custom tracking algorithms to work, as in [29].

Specifically, polynomial models are usually considered for their strong constraints, enforcing very smooth lines [30, 31, 32]. Usually indeed only low-order polynomials are used, generating at most cubic functions. Such constraints are a great advantage in presence of noise, since the usage of only few parameters - and thus of a strongly biased

model - maintains the error variance confined. Nevertheless, the risk of underfitting is not negligible, especially on curve roads or in complex scenarios.

Piecewise-defined models instead are clearly richer and can capture a large variety of scenarios thanks to their higher number of parameters. However, more attention must be put into the definition of their constraints, in order to avoid overfitting and thus infeasible results. Among the several types of splines available, cubic splines have been successfully tried in different occasions [33, 34], possibly for their ease of use and their great expressive power. In [35, 12, 36] B-splines are instead applied, to better capture curvy road shapes.

We must point out that different models clearly entail different prior knowledge on the structure and shape of the roads, although variability is present also in this regard. Highways in fact, on one hand, present mostly straight patterns and limited curvatures, to support a sustained cruise velocity. Moreover, the bends in the road are usually designed to constrain the change of curvature and thus avoid non-smooth experiences for the driver. A typical curve used in this context is the Euler Spiral, or clothoid, as mentioned in [37], which reports a study of different road curvature designs and their characteristics. If this is the case for highways however, urban roads, on the other hand, usually follow less strict constraints on their smoothness, because they need to obey to other type of constraints, these ones imposed by the morphology and the topography of the particular urban area in which they are found. We can think for example at the extreme case of a small European mountain village, where the shape of roads is almost totally dictated by the historical development of the town and of its buildings and is usually challenging to navigate.

It is furthermore important to mention that the modeling of the line markings

could be improved by the introduction of additional constraints. For example, we usually search for at least two line markings, defining the lane our vehicle is in, and for some application we expand the search to even several lines. If the lines belong all to the same roadway, and no road intersection is present, they must then all appear parallel: this is a very valuable prior information that could be exploited at this stage, as it is done in [38]. Other similar assumptions could be on the type of road traversed: a system working on highway roads will probably not need to look for strong road bends or for intersections, and will be more prone to look for straight lines, thus nonconforming candidates can be discarded. Additional considerations can be done also on the difference between left and right line, as exploited in [29, 33].

We must finally remark on a crucial aspect of using a model relying on parameters (be it parametric or semi-parametric). Given a set of feature points extracted in the previous stage, such points are usually in large number and often still containing noisy detections. In such cases, before fitting the actual model, it could be important to reduce their number and possibly filter them with a fruitful criteria, in order to decrease the computational resources needed for their processing and at the same time eliminate part of the noise. In our experiences, we often had to design specific algorithms for this purpose. Other approaches rely instead, for example, on robustness properties of the fitting algorithms adopted, as for the use of the RANSAC algorithm in [33, 34].

In conclusion, a final fitting algorithm has to be selected. The most common solutions involve derivations of the least squares method (as in [39, 25]), the Hough transform (as in [40, 41, 31, 42]) and the RANSAC algorithm (as in [43, 32]). At times, also combinations of them are used, as for [44].

2.4 Tracking

Performing our analysis of the lane markings frame by frame can be effective, but highly inefficient. A large amount of information retrieved in a previous frame, in fact, can be exploited to guide the detection in subsequent frames, shortening the computational time and increasing the robustness of the system. The way of achieving this improvements is by means of tracking.

Tracking is the process of analyzing the position of an object recorded over time and estimating its trajectory [45]. In our context, it consists in considering the sequence of images we captured in the past frames, along with the detections we performed, and using them to correct our future detections and estimate the lane markings with more accuracy and robustness. The adoption of a tracking procedure allows us not only to maintain our estimates within a reasonable confidence level, coping with the noisy images received by the camera, but also to make acceptable predictions on the position of the markings in case of a missing frame or of a temporary failure of our feature extraction system (e.g. because of a sudden illumination change). Moreover, this prediction ability can be exploited to reduce the search space in any of the previous stages, from the preprocessing (e.g. for the ROI) to the model fitting (e.g. for limiting the search only to plausible models).

Several algorithms are available and have been long studied in many engineering fields. For our problem in particular, the literature is almost completely relying on the Bayesian filtering framework, with Kalman filter [16, 46], extended Kalman filter [47] and, particularly, particle filter [48, 29, 33] being the most common algorithms adopted. Details on their implementation can be found in the literature, as for example in [49].

Any of such algorithms is based on two fundamental concepts: the motion model and the sensor model, which produce estimates respectively of the dynamic behavior of the system and of its perception capabilities. Finding a precise definition for these components is quite difficult because of the large amount of variability to take into account. For this reason, usually approximations are used, as in [29], where the position of the lane markings is assumed to remain fixed over time to reduce the complexity of the system.

Chapter 3

Experimental reports

3.1 Histogram-based edge point selection

3.1.1 Introduction

The application of an edge detector to the Bird's Eye View image enhances lane candidates pixels. However, in several road scenarios (e.g. urban environments), it is also likely to introduce a fair amount of false positives due to clutter, occlusions and perspective distortions. Moreover, the number of resulting edge points is too large to be directly used in the model fitting phase without a strong impact on the computational performances of the system. There is need therefore for a method able on one side to restrict the attention on a smaller number of candidates, and on the other to significantly reduce the presence of false positives. We attempt to tackle these problems by exploiting the continuity of lane markings along the road with an histogram-based method.

3.1.2 Previous work

An approach based on histograms of edge points, as presented below, is not commonly found in the literature of computer vision or of the newborn autonomous driving field. However in web tutorials as [50], an analogous procedure is adopted. After a preprocessing stage on the Bird's Eye View of the road, the pixel values of a binarized image are cumulated vertically, forming a histogram to be used in the estimation of the closest position of the lane marking. [50] successively applies a sliding window technique to follow the trace of the lane marking, strongly relying on its histogram-based first estimation. As our scenarios present more noise and diversity, coming from more general contexts, we can't rely on this first estimate, and thus our adaptation of such procedure comes into play.

3.1.3 Methods

We consider a Bird's Eye View image of a noisy road to which an edge detector has been applied. We predominantly adopt the known Canny edge detector, but the same procedure and results would likely follow also from other gradient-based detectors.

Of the edge pixels detected, we consider as *true positives* the ones belonging to the lane markings, and as *false positives* the rest, coming for instance from road objects, part of the image external to the road and noise produced by the acquisition apparatus. We won't make use of it in this section, but we define in like manner as *false negatives* the lane marking pixels not detected, and as *true negatives* the remaining ones.

From the analysis of the edge images obtained, we notice as expected that the false positives have a chaotic arrangement in the image, while the true positives are

substantially vertically aligned. We design therefore our algorithm to exploit this characteristic.

We split the edge image in horizontal stripes of some tenths of pixels each, assuming the profile of the lane markings won't change significantly in such a short region. For each stripe, we compute the number of edge pixel present in each column, and form a histogram of such values, as in Figure 3.1. We notice that if a vertical lane marking is present in a strip, the histogram will most likely present a corresponding peak. We extract therefore the maximum values from such histogram, and use their x location, combined with the y coordinate of the center of the stripe, as an extracted point. Since some peaks might be also formed only by false positives, as many as 3–5 peaks are considered per stripe, in order to assure high *recall* (inclusion of many of the true positives).

Figure 3.2 illustrates the outcomes of the mentioned steps. The method relies on two hand-tuned parameters, N_peaks, number of maxima to select for each histogram, and $Height\_stripe$, height in pixel of each image stripe.

3.1.4 Results

We evaluate the algorithm in two ways: by the quality of its results in terms of precision and recall of the selected points and through the application of a simple fitting model, based on RANSAC, in order to relate such results to our final purposes.

Points quality

From Figure 3.2 we observe a significant reduction in the number of points to be considered in the following stages (model fitting), along with an enhancement of the

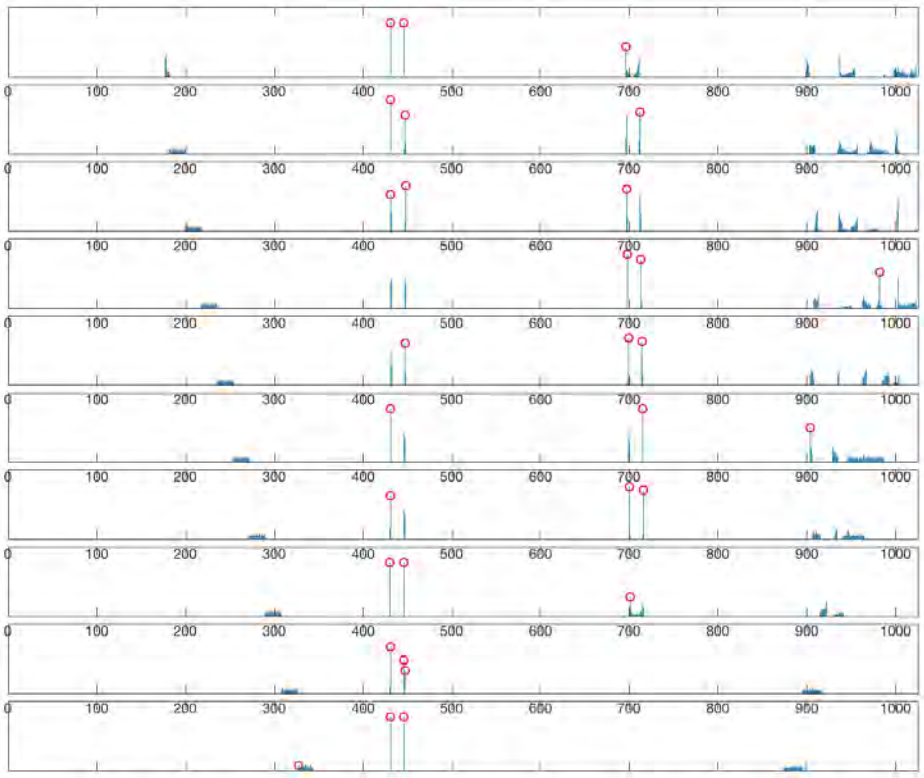
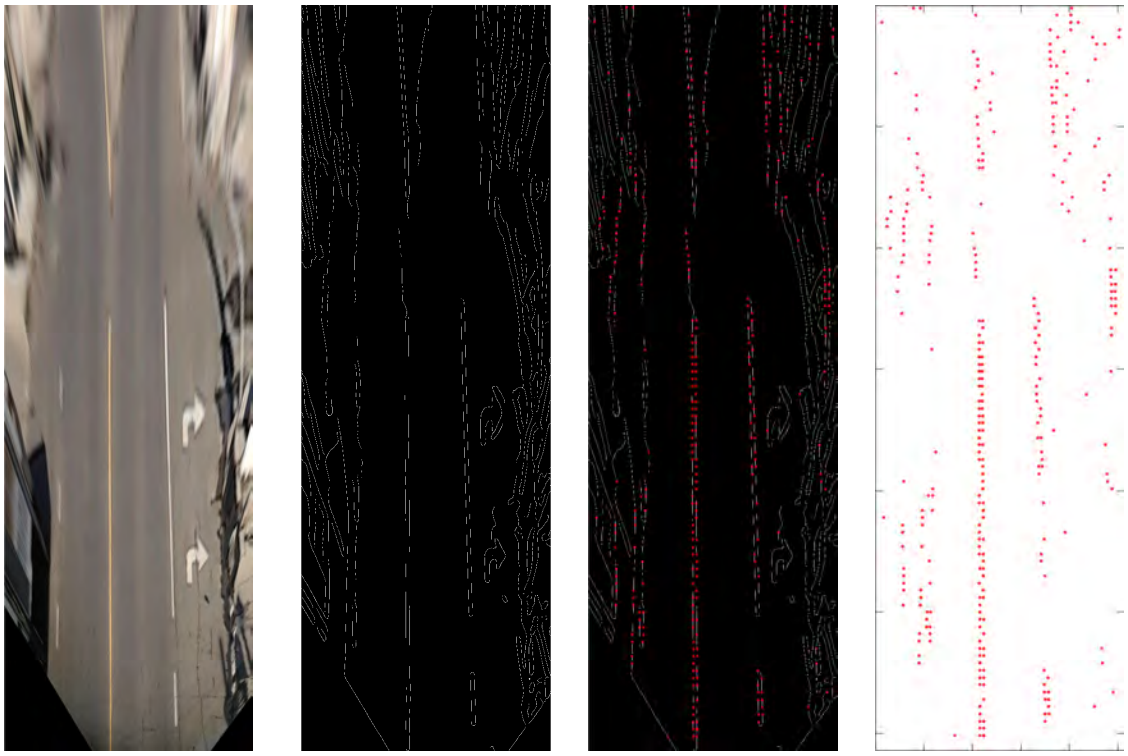


Figure 3.1: Sample of the histograms of edge pixels per column computed for 10 stripes in the images further analyzed in Figure 3.2. The selected peaks are shown in red.

overall *precision* (percentage of true positive selected), both expected and valuable features. However, analyzing other frames acquired in different road conditions (Figure 3.3), we notice that the same set of parameters leads to several misses in the detection of important lane points and to a decrease of the overall precision. Moreover, other alterations of such parameters do not seem to compensate enough for this issue without unreasonably compromising the previous result, and this behavior suggests an inherent lack of generalization of the algorithm to even slightly changes of road conditions. This issue could however be fixed with the design of an appropriate way to adapt the parameters to such different condition. Another cause for the

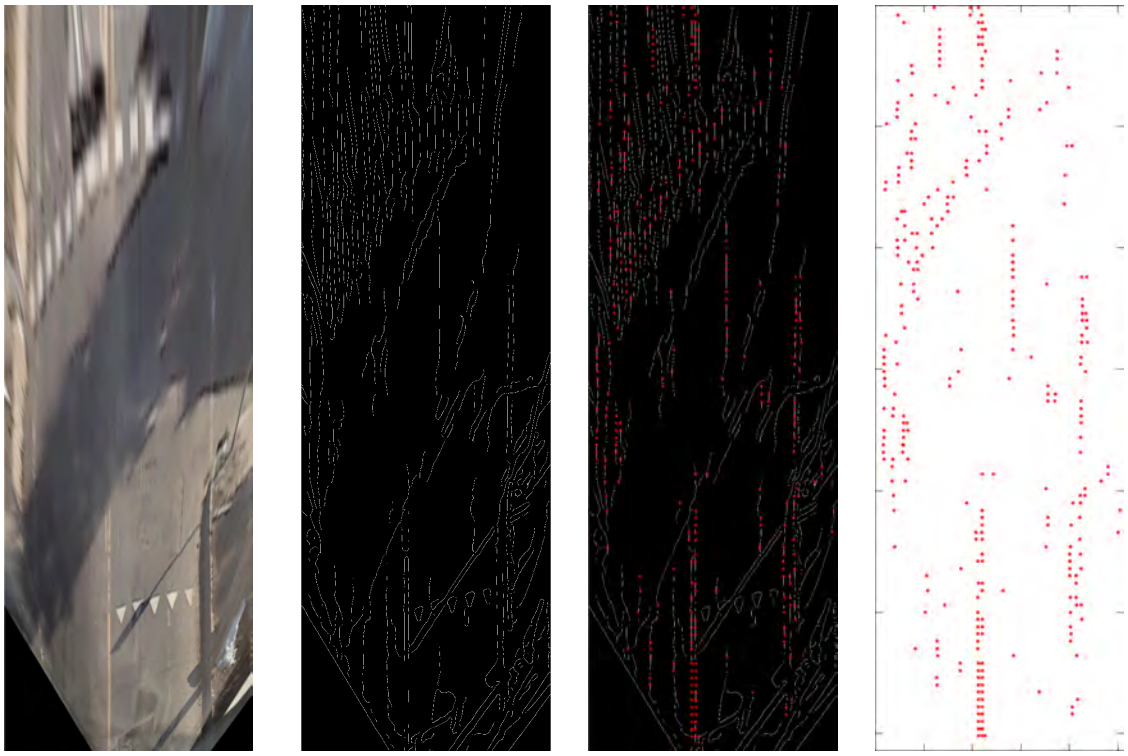


(a) (b) (c) (d)

Figure 3.2: Given the Bird's Eye View image (3.2a), an edge detector is applied (3.2b, Canny edge detector). The edge image is divided in horizontal stripes of 30 pixels ($Height\_stripe$) and for each stripe an histogram of the occurrences of edge pixel per column is computed. 3 points (N_points) are then selected as maximum histogram values for each stripe, constituting the point set displayed in (3.2c) and (3.2d).

significant presence of false detections could be due to the insufficient performances of the edge detection algorithm under particular road conditions. This aspect might be mitigated with the use of an edge detector more robust to noise, such as the Line Segment Detection algorithm (LSD).

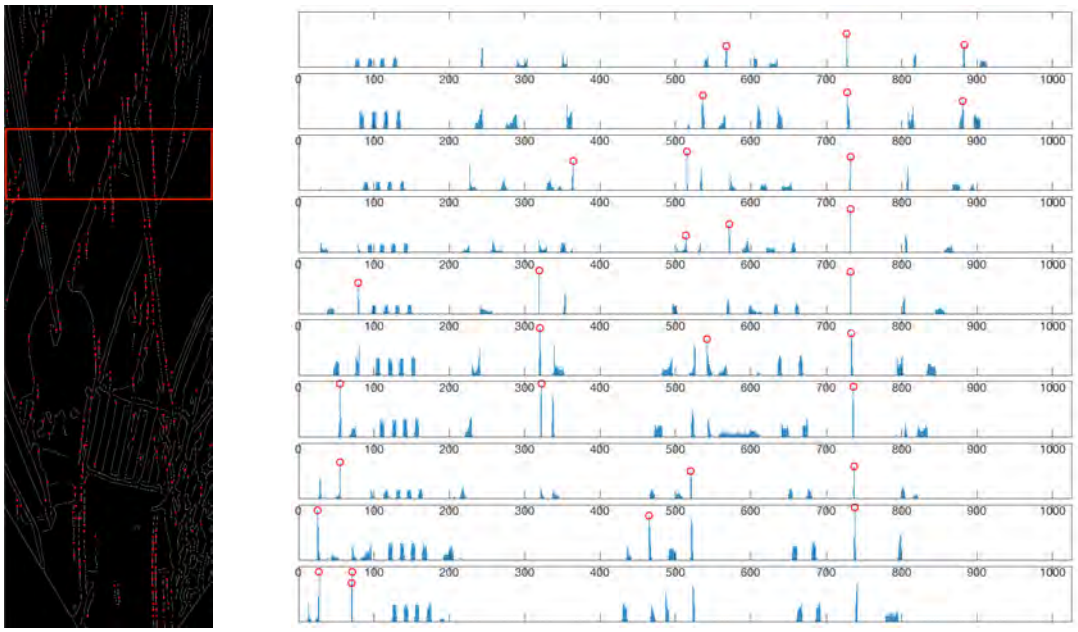
Nevertheless, the main issue observed lies in the assumptions that edge pixels from the lane markings would result vertically aligned (see Section 3.1.3). This is in fact true for straight lanes, but it doesn't hold for bends in the road, even when the stripes



(a) (b) (c) (d)

Figure 3.3: The pipeline stages described in Section 3.1.3 are applied to another frame. The images, and in particular 3.3c and 3.3d, highlight how the algorithm, tuned for working in Figure 3.2, is not able to adapt to the slightly more noisy scenario here represented, where the road is still straight but presents distracting lane markings and challenging illumination conditions.

height is very small. As a matter of fact, the profile of the corresponding histogram will be more dispersed around the lane marking (Figure 3.4a), potentially not accumulating enough points to become a global maximum. To recognize such marking we would need to analyze the histogram with a different approach, perhaps by means of its first derivatives after appropriate smoothing. However, since the algorithm lies on the hypothesis that true and false line marking pixels can be discriminated by their vertical alignment, it would be hard to tune such mathematical tools to operate such classification.



(a) (b)

Figure 3.4: Result of the algorithm applied to a turning road (a) and example of histograms (b) computed over the section highlighted in red of the edge image. The noticeable noisy component present in the image overcomes the lane markings, that are not selected because of their misalignment.

Fitting results

We test here the capabilities of the algorithm in enhancing the features to be exploited by the model fitting stage. To do so, we send its output to a simple line fitting procedure based on Random Sample Consensus (RANSAC). The procedure is not meant to be applied *as is* in our pipeline, but rather has been built for the mere purpose of assessing this preceding stage. Our fitting tests focus only on fully parametric model, that is to say a single, fixed-order, polynomial curve for each marking line.

As a first step, the points obtained are separated in two classes, *left* and *right*, according to their x-position. Then, on each class individually, RANSAC is used to identify and remove the outliers, and successively the model is fit with the remaining

points. The fitting is intended in the least-squared sense, and it is measured by means of the L_2 norm.

We apply this procedure to the frames seen in Figure 3.2, 3.3 and 3.4, in order to assess different road and edge image conditions. Figure 3.5 highlights the results.

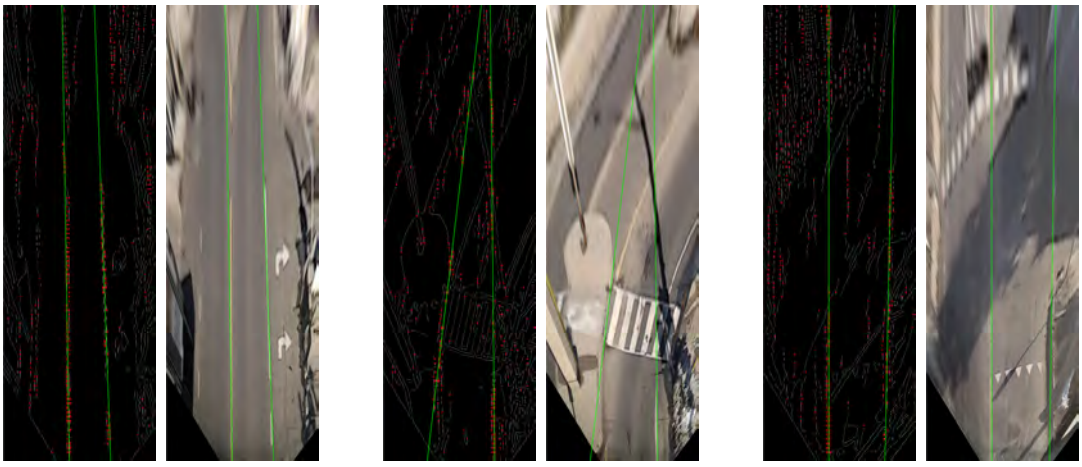
We notice that the algorithm results fairly effective on straight roads, even with a moderate amount of noise. On the other hand however, its performances drop drastically on bends of the road. This behavior can be explained, as mentioned previously in Section 3.1.4, by the fact that the assumption of lane markings being vertically aligned is not holding. Indeed, as illustrated in Figure 3.6, higher order curves are equally not capable of representing the exact lane markings from the same points, indicating even more the inappropriateness of the algorithm for non-straight road scenarios.

3.1.5 Conclusions

The algorithm can act as a starting point but we proved it to be not sufficient, *as is*, to achieve the desired goals.

Alternatives and improvements have been proposed in the previous sections, and can be summarized as follows:

- adaptive tuning of the method's parameters N_peaks and $Height\_stripe$;
- use of an edge detector less sensitive to noise, such as LSD;
- improvement of the peak selection method applied to the histograms, for example by avoiding to pick points too close to each other though non maximum suppression.



(a) (b) (c)

Figure 3.5: Evaluation of the impact of the algorithm in the final goal of the pipeline, i.e. detecting lane markings. A simple first order fitting procedure is applied to the selected edge points, displayed in red above. RANSAC is used to filter out the outliers, and subsequently a least-square line fitting is computed. Clearly, in favorable conditions as in 3.5a our procedure is precise enough to lead to a successful fitting, but as the image or road conditions move away from such reference many noise points are not filtered, impacting in the fitting results more (3.5b) or less (3.5c) significantly.

However, the current implementation proved to be not capable of dealing appropriately with road bends because of its underlying assumptions, and the said improvements may therefore not be enough to correct its flaws.

As a matter of fact, other approaches could be adopted to achieve a significant reduction in the number of considered points, although maybe not guaranteeing a large precision improvement. Among these is the application of random sampling. An appreciably narrower subset of points can be drawn from the set of edge points and used in succeeding stages such as the model fitting, at essentially no computational cost. In addition, since the samples can in theory be drawn by any distribution, one could embed prior knowledge in the process, thus achieving a relative increase in precision. A uniform distribution for instance would preserve the overall arrangement

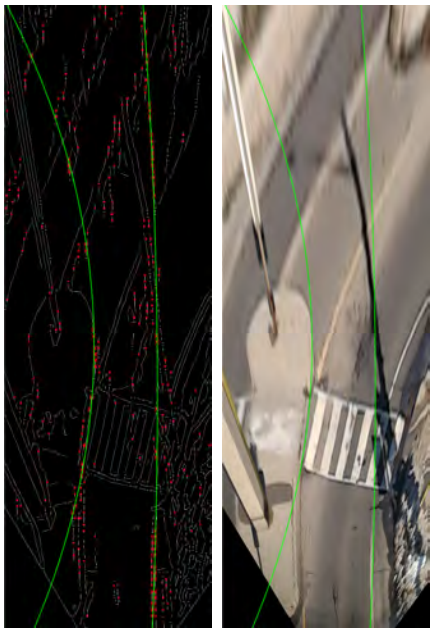


Figure 3.6: Evaluation of the algorithm performances if followed by a second order model fitting. This test is realized in response to the expected low precision observed in Figure 3.5 for bending road. As this figure highlights, even with higher order polynomial the fitting in such conditions is not satisfactory, and on the contrary is even worse than in the previous attempt. We conclude that most likely the influence of false positives in this case is too high for the fitting stage to produce acceptable results.

of points, including in the sample as much noise as the edge detector collected. On the other hand, weighting the samples based on their distance from the center of the image, or, better still, to the center of the lane estimated from previous frames, would most likely be beneficial for our purposes.

3.2 Vanishing point estimation based on short distance Hough transform

3.2.1 Introduction

We design an algorithm for the estimation of vanishing points of the scene observed by our monocular camera mounted on the vehicle. The system is assumed to be in motion on the road. The algorithm relies on the presence of lane markings on the pavement and is therefore not suited for unpaved or rural environment, which are outside the scope of our investigation. As potential part of our lane detection system, we expect the algorithm to be fast to achieve a real-time computation. The potential field of application for the algorithm is at first the dynamic adjustment of the camera calibration, and in particular of its pitch angle, significantly sensitive to bumps on the road and slope changes. Secondarily, the possibility to reduce subsequent computations through a properly scaled ROI and the employment of the horizon line as part of the model fitting algorithm are also considered valuable.

3.2.2 Previous work

The estimation of the vanishing points is an important tool for the application of Computer Vision to the fields of Robotics and Vehicle Perception. It provides a way of determining both the shape of the surrounding environment and the position of the camera - and thus of the attached system - in it.

As a versatile component of several vision systems, it can be adopted to save computational time by opportunely constraining the problem [51], or have a more complex role, from the characterization of the horizon under a non-flat ground assumption [9],

to its integration in an adaptive camera calibration algorithm [52]. Equally important, and closely related, are the applications that exploit vanishing points to compute and adjust inverse perspective mappings such as Bird's Eye View and scene rectification [46]. Finally, we must mention its role in several fitting algorithm for lane marking detection [1, 2, 53, 10].

[52] makes use of projective geometry and particularly of the estimation of vanishing points to demonstrate a simple calibration method for the extrinsic parameters of his camera, based only on a rectangular surface as reference. He then applies his founding to the real-time calibration of an unmanned aerial vehicle (UAV) in urban environments, with the vanishing points estimated from buildings and street edges. We find his work particularly useful for the computation of camera pose and orientation using simple objects, and interesting for the ability of detecting vanishing points on all three axis, characterizing the perspective geometry of the scene. However, we realize the line detector used must be precise enough to avoid the introduction of spurious lines, which can compromise the estimation, especially in a real-time scenario.

Nieto et al. [46] adopt a vanishing point estimation technique to adaptively compute their Bird's Eye View without relying on fixed extrinsic parameters, thus improving the stability of their road segmentation system. They apply a custom line detector designed to enhance lane markings and prevent noisy detections from other vehicles and external objects on the road. The Hough transform is then computed on the detected points to identify several candidate lines. This phase is combined with RANSAC to provide robustness to the algorithm with respect to outliers. Finally the vanishing points are computed by least square fitting using SVD (Singular Value Decomposition, [54, Chapter 3]) and tracked into a Kalman filter with a constant

velocity dynamic model.

The system strongly relies on the custom edge detector to separate noise from lanes, although it depends on a hand-tuned parameter strictly connected to the frame itself. In addition, no proper evaluation of its performance is shown. For this reasons, its possible lack of robustness might clearly represent a flaw, even though RANSAC helps cope with the risk. We must point out moreover that the test cases shown represent only straight, highway-like roads, and therefore a lack of precision in complex scenarios might be observed. Nevertheless, it is able to cope with obstructions from other vehicles, thanks to the segmentation assumptions on the aspect of lane markings and roads.

Vanishing point estimates are used as control points for the Catmull-Rom spline construction adopted by Wang et al. [10]. They claim that approaches based on direct Hough transform could lead to more desirable results than line clustering, statistical tools, an other more sophisticated variations. In their method, the orientation of the image gradient is processed to define lines of interest, which in turn vote for a vanishing line through a length-based Hough transform voting mechanism. Their results seem robust enough to changes in illumination and road conditions, and are applied indiscriminately to straight and bending roads. On the other hand, no estimation of the efficacy of this module, with respect to the whole system, is reported. We finally note that only few lines for each frame participate in the voting mechanism, and therefore more attention and responsibility is placed on the computation and elaboration of the gradient image they are extracted from.

Finally, in [9] a vanishing point estimate is applied to an adaptive ROI in order to

filter out unwanted information from the image without missing important components. Their algorithm extracts from a gradient image all the possible horizontal lines applying a Hough transform, computes then all of their intersections and through a voting scheme finally finds the vanishing line, which limits the Region Of Interest. Interesting for effective realizations, their algorithm is optimized for real-time embedded systems by opportunely constraining the Hough transform algorithm.

3.2.3 Methods

For clarity, we divide our description in four sections covering the four main stages for any algorithm of this kind, namely *preprocessing*, *feature extraction*, *model fitting* and *tracking*.

Preprocessing

As a preprocessing phase, a Region Of Interest is applied to the image in order to focus the analysis on a short segment of road in front of the car. In the short distance in fact, even when the vehicle is turning, lane markings can be approximated by straight lines. This property helps us reduce our computational time and, at the same time, increase our true detections ratio. Furthermore, in such short section it is easier to assume that only the road is captured, without obstructions from other vehicles or the misleading presence of pixels external to the road. This ROI computation is based on a previously performed extrinsic calibration with respect to the vehicle's world frame, and in our configuration is about 10 meters wide and extends for only 7–8 meters from the front of our car. Since this step is only finalized at focusing on the road in the short distance from the car, it is not required to be extremely precise. This is

why fixing the calibration matrix is acceptable for this step.

Features Extraction

On the extracted image section, after the detection of edges using the Canny algorithm, a constant number of lines is retrieved through Hough transform. In its standard version, the Hough transform is a majority voting algorithm used to detect lines in an image, although it can be adapted to detect other geometric objects as well. For computational issues, the parametrization of lines is usually obtained in polar coordinates (ρ, θ), as:

$$x \cos\theta + y \sin\theta = \rho \tag{3.1}$$

where ρ represents the normal distance of the line from the origin of the image, located in the upper-left corner, and θ the angle the normal forms with the x-axis (Figure 3.7).

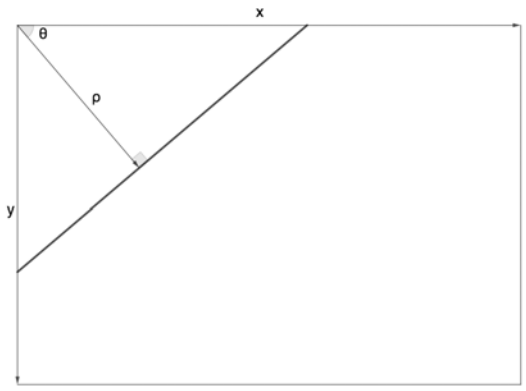


Figure 3.7: Polar representation of a line used by the Hough transform algorithm.

We distinguish the lane markings at the right of our vehicle from the ones at its left by the evaluation of their θ parameter, with simple geometric considerations.

This distinction is useful for reducing the detections of false vanishing point generated by noisy lines, but can also be used for further considerations on the position of the vehicle on the road. We point out however that this assumption is only justified on straight roads, and may be void for sharp road bends, in which case other mechanisms should be adopted.

Since vanishing points generate at the intersection of parallel lines at the horizon, for every pair of right and left line detected we consider their intersection as a candidate point. We have no guarantees that such lines are parallel to each other, but we know that, assuming the road is paved and marked, some will be corresponding to the lane markings, parallel by definition.

With all this in mind, we now need on one hand a method to filter out our candidates, and on the other a prediction-correction mechanism to stabilize the results in presence of noisy and missing detections. Since we are interested only in variation of the pitch of the vehicle, and thus only in the y coordinate of the points (assuming absence of significant roll), we furthermore discard the x coordinate and proceed with the tracking of the sole y component.

Model fitting

For the candidate selection, voting techniques are sometimes used, as seen in [9] and [46]. However, voting mechanisms such as RANSAC are known to significantly increase the computational costs, and if the presence of outliers is minimum, they may not be worth the effort. Moreover, since their result is still treated as a noisy measurement from the successive tracking step, the impact of noisy detection can remain contained. With this in mind, we opt for - and test - two less computationally

expensive criteria.

Despite its simplicity, as first methodology we adopt a Nearest Neighbor (NN) approach: among all the candidate points, we select only the closest to the vanishing point estimated at the previous time step. With this technique, spurious candidates are automatically discarded due to their distance to the tracked estimate. It is clear that this solution is reliable only for small variation of pitch over two consecutive frames. To increase the robustness of the method in the eventuality of detecting only outliers, the selected candidates are accepted only if within a threshold d from the previous measurement; a missing detection takes place otherwise.

In search of an alternative to this selection technique we notice that, assuming no outliers are present among the candidates, the average of the tracked coordinate (y) of each point represents the maximum likelihood estimate of such quantity, assuming each measurement is affected by Gaussian noise. As a reasonable way of removing possible outliers then, this seems a feasible approach. To maintain the computational cost low, under a rationale analogous to what mentioned for NN, we consider as inlier all and only the candidates within the threshold d mentioned above.

In both alternatives, d can be dynamically shrunk as the confidence over the vanishing point estimate increases.

Tracking

Finally, a prediction-correction procedure tracks the point and provides robustness and consistency in the time domain to the algorithm. Given the simplicity of the system, a linear model is adopted and a Kalman filter is applied. The role of the filter in this context is mostly to smooth out the prediction and cope with possible

missing measurements.

We represent our state as the single variable y, position of the horizon in the image. Clearly, y is fully observable, being also the only quantity object of our measurements. No control variables are considered. The system can therefore be described as:

$$\begin{cases} y_t = Ay_{t-1} + w_t \\ z_t = Hy_t + v_t \end{cases} \tag{3.2}$$

where z_t represents the measurement at time t of the state y_t, A is the state-transition matrix and w_t and v_t are respectively the process and observation noise, assumed to be drawn from zero-mean Gaussian distributions of covariance Q and R respectively.

To obtain a smooth estimate, and in absence of additional information, we define the state-transition matrix A as the identity matrix. This corresponds to assuming that the position of the horizon perceived from the vehicle, in absence of external input, remains unchanged, but our confidence on such estimation is decreased by w_t. This results in a simplistic but effective way of describing the system despite the lack of information about the vehicle relative motion.

Being z_t the measurement of y_t, also H is defined as identity matrix.

Finally, the initial covariances of the filter are hand-tuned to trade off precision and robustness.

3.2.4 Results

We apply the algorithm to data recorded in two different occasions. The two datasets partially cover the same roads, so we can contemporary test the performance of the algorithm on:

- same road, but different driving conditions; and
- same driving conditions, but different roads.

Our original aim was to apply the algorithm for the dynamic correction of the camera extrinsic parameters, and in particular of the pitch, to stabilize the extraction of the Bird's Eye View. For this reason, we would require the algorithm to be extremely precise for it being actually employed, and we perform our evaluation with this in mind.

The algorithm appears robust in highway driving, where the constant presence of lane markings and the simplicity of the environment provide a good amount of correct measurements to adjust and refine the estimate (Figure 3.8).

On urban routes however, the lack of constant detections combined with the substantial presence of noise in the edge images and, consequently, in the retrieved lines, constitute a barrier for the algorithm to achieve satisfactory results. Moreover, in urban environments, often complex lane topologies arise, with lanes not parallel to each other (Figure 3.9a) and at times not even marked (Figure 3.9b). In such cases, the additional inconveniences of false and missing detections arise. Despite these complications, characterizing the process noise with larger covariances stabilizes the results, as seen in Figure 3.10, although the precision required for our purposes cannot be achieved anymore.

An important note on the efficacy of this method is the implicit assumption that the camera is mounted with a null roll angle with respect to the vehicle. The estimation is in fact carried on only in the y coordinate, which is as assuming that the image x axis is perfectly parallel to the world horizon. If on the other hand the camera is subject to a roll angle, a correct estimation of the horizon position would

require the knowledge of at least two vanishing points on the ground plane. This could be achieved detecting parallel road markings in more than one direction, but it represents a condition hardly verifiable in normal driving situations. For this reason, in this case our algorithm would be inapplicable. Nevertheless we must mention that, if the roll angle of the camera is not null but remains constant in time, as it happens in most cases, the algorithm can be adapted to cope with this information. Notice finally that, to further improve the estimation, the roll angle could also be perceived by means of additional sensors, correcting the image at a preprocessing level.

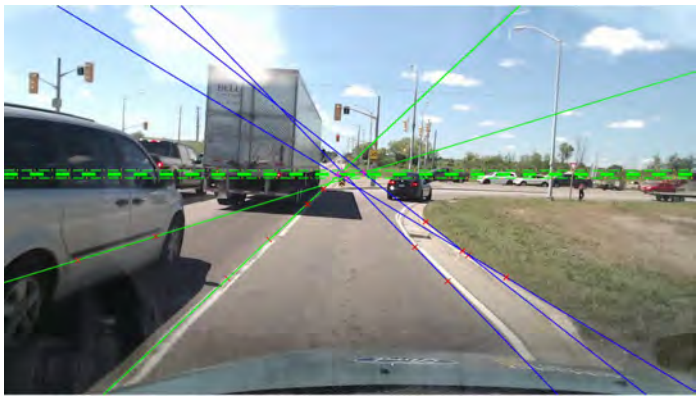
On the efficiency of the method, although we are not resorting on significantly complex operations, the computation of the edges and especially the application of Hough transform are known to be expensive.

A gradient-based edge detector such as the Sobel algorithm would require the convolution of the image with a gradient filter, and the subsequent thresholding of the gradient magnitude image [55]: the three operations are linear in the number of pixel, and can be defined as $O(nm)$, being $n \times m$ the size of the original image. Although it can be adapted for the use of this type of operators, our algorithm is instead designed for the application of the Canny edge detector [56], which requires additional convolutions and some subsequent linear scan of the image for the non-maximum suppression and thresholding operations. The algorithm remains overall linear in the number of pixels ($O(nm)$), but the computational time can in practice increase significantly by multiplicative factors.

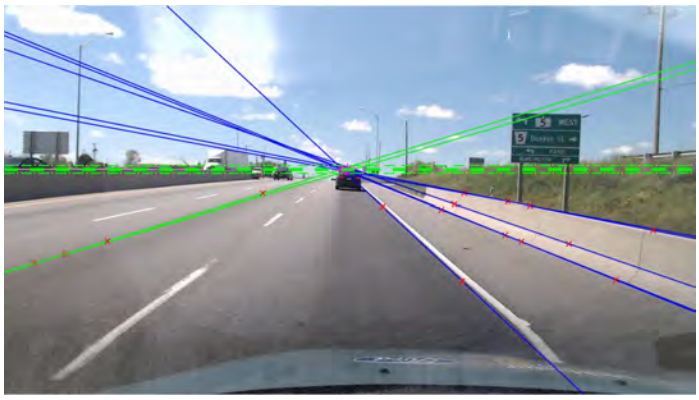
The computation of the Hough transform, given an edge image previously computed, has a cost dependent on the number of edge points (N_e) it is fed with, the number

of parameters to be estimated (p) and the size of the quantization grid in each dimension (k). In Big-O notation, it can be expressed as $O(N_e + p^k)$ [57], which can be approximated by $O(n + p^k)$ assuming $N_e \approx n \approx m$, with $n \times m$ dimension of the original image.

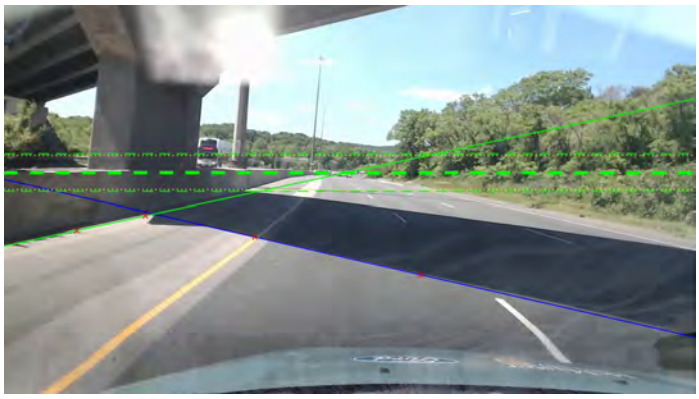
We must finally point out that the evaluation is performed by visual analysis of the results. This qualitative method inherently introduces subjectivity and imprecision in the assessment, but is justified by the lack of suitable benchmarks in the field of computer vision for autonomous driving in general, and of vanishing point estimation in particular. Some labeled datasets can actually be found on this specific task, as highlighted in [58], such as the York Urban Dataset (YUD) [59], the Eurasian Cities Dataset (ECD) [60] and the Horizon Lines in the Wild (HLW) dataset [61]. While the latter represents mostly natural and unpaved paths, the formers could in principle be used, being recorded in urban environments. However, they do not focus on driving, and most of the images they contain are taken from perspective external to the road. Our algorithm would by construction fail on such data, because built on the assumptions that images are taken from the frontal camera of a vehicle on the road.



(a)

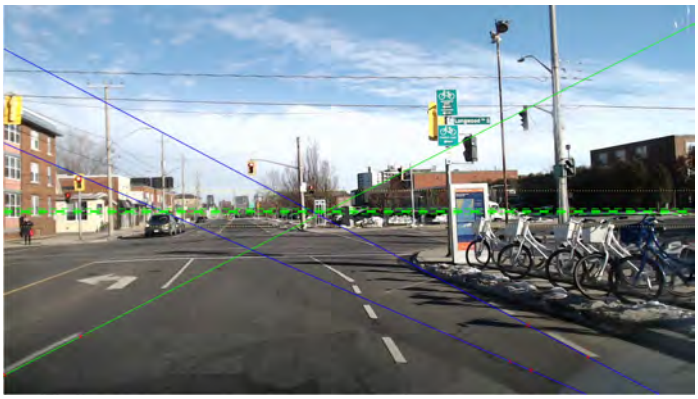


(b)



(c)

Figure 3.8: Results of our algorithm on highway roads. The averaging selection method (Section 3.2.3) is applied, and an example of situation in which multiple detections are found is provided in 3.8a. The algorithm proves itself capable of reacting to changes in slope and to track the horizon properly (3.8b). Moreover, the presence of noisy detection, especially due to shades (3.8c), does not impact its overall performance, thanks to the feature selection step based on threshold d (Section 3.2.3).

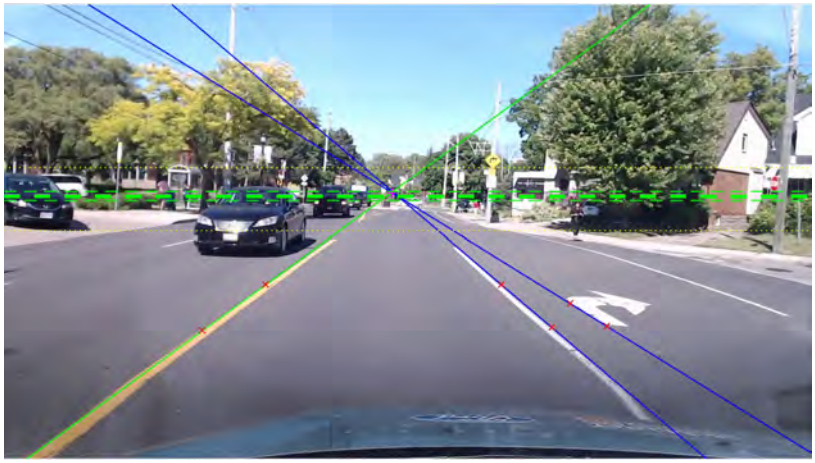


(a)

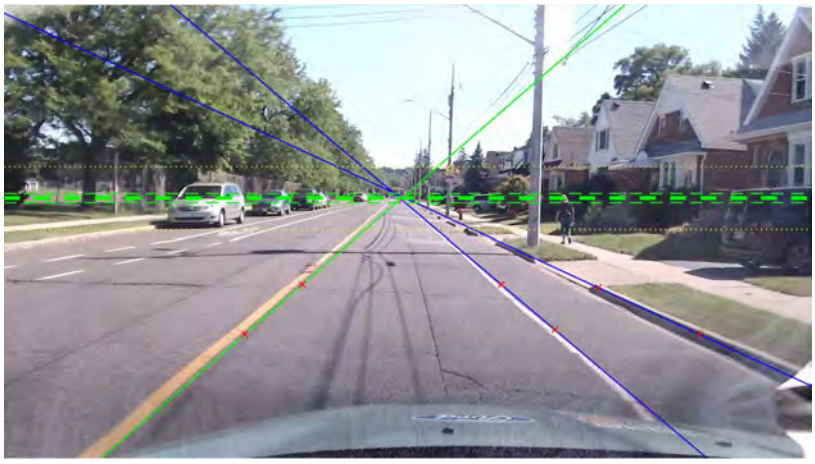
(b)

(c)

Figure 3.9: Examples of the issues registered on urban roads are depicted here. 3.9a shows a false vanishing point due to the detection of two non-parallel marking lines at an crossroad. In 3.9b instead the complete absence of markings is treated; although with a large uncertainty, in this case the estimate of the horizon remains substantially valid thanks to the tracking stage. When the lack of intersecting parallel lines persists however, the precision of the algorithm is severely impacted until the next detection, as shown in 3.9c.



(a)



(b)

Figure 3.10: Acceptable estimates in urban roads are presented here, recorded in presence of straight continuous lines. Some minor obstacles and potential noise source is present, although not particularly significant.

3.2.5 Conclusions

The algorithm is capable of performing the required estimation when strict constraints on the confidence interval are not necessary, while it does not meet the precision requirements for tasks as the dynamic calibration of the extrinsic parameters of the camera. This condition is particularly accentuated on urban roads, while highway recordings show an overall partial compliance. Nevertheless, we would still benefit from this computation as a bound for the definition of the ROI and as an additional information to consider during the estimation of the lane markings, particularly in the model fitting stage.

3.3 Line marking feature extraction in color space

3.3.1 Introduction

In the lane detection problem, the extraction of color features aims on one hand at enhancing the pavement markings to consolidate the detections from other extraction techniques, and on the other hand to capture the additional information conveyed solely by the color channels. The analysis of color features in fact can lead to the separation, in the image, of the pavement from the different types of lane markings. This can be important also because of the semantic associated to each color and shape of lane marking, usually prescribing different behaviors to the driver.

In this sense, the most interesting example for our purposes is the distinction between yellow and white lines. In North America, yellow lines are usually employed to separate traffic directions, while white lines are used elsewhere. Differently, in many European countries no such distinctions is made and white is used in both the situations. To complicate things however, in these countries yellow lines are often used to indicate temporary lane shifts due to construction projects, and coexist with the permanent white markings. For these and more reasons, their distinct identification can be crucial in the context of autonomous driving.

This leads us to our objectives. Our investigation is indeed conducted with two driving goals:

- primarily, we intend to study the possibility of detecting any line marking based solely on color features;
- secondarily, we try to evaluate the possibility of discriminating line markings by their color characteristics (i.e. yellow and white).

3.3.2 Background and Previous work

The extraction of features in computer vision often focuses only on monochromatic images, for example in the computation of gradients and in the detection of corners [5, Chapter 5]. However, almost any camera system can nowadays capture color images, and while flattening them into a single-channel would lead to a substantial information loss, the analysis of their color properties and, when needed, the redefinition of some of the algorithms involved, could be beneficial for the overall system performances [62]. To analyze the usage of color features in the literature and in our work, we first need to define some of the known color-spaces and discuss their properties.

RGB

It is known that while a monochromatic image can be represented as a matrix of pixels, a color image is usually encoded using few matrices, each containing the values registered by each camera sensors in the respective portions of light spectrum. The most common of such encodings is represented by the RGB color space (Figure 3.11). RGB is a linear color space, and it is defined considering as primary colors the ones corresponding to the wavelengths 645.16 nm, 526.32 nm and 444.44 nm (R, G and B respectively) [5, Chapter 3.3].

However, for the Dichromatic Reflection Model [62, Chapter 1], the values measured in an RGB space strictly depend on the position and orientation of the surface, of the viewer and of the light source [62, Chapter 1]. This means that the light coming from the same surface can be encoded in severely different ways if the surface is moved, or if it is moved from sunlight to shade. This dependency on environmental changes constitutes a dramatic drawback for our purposes, since the images captured by our

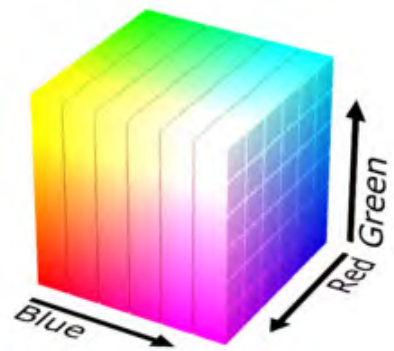


Figure 3.11: RGB color space, from [63].

car are often taken in presence of shaded areas and different illumination conditions due to changes in weather or to time of the day. Moreover, the car moves during time, thus changing the viewpoint of the scene and potentially the colors captured. It is for this reason that we explore different color spaces.

YCbCr

From a linear combination of the three channels R, G and B we can obtain the color space YCbCr (3.3), a luminance-chrominance space known mostly for its interesting image compression properties [64, Chapter 3]. YCbCr is built as a purely linear combination of the components of RGB, with coefficients slightly depending on the particular implementation used (as the discrepancies between [65, 66, 67] demonstrate), but not enough to affect its general properties. According to the ITU-R Recommendation BT.601 [65], and as also reported in [64], the YCbCr can be obtain

by the following transformation

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.29900 & 0.58700 & 0.11400 \\ -0.16874 & -0.33126 & 0.5000 \\ 0.50000 & -0.41869 & -0.08131 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} \tag{3.3}$$

As described in [9], this space can be used to retrieve an illumination-invariant operator for detecting white and yellow markings. The authors in fact explain that the Y (luma) component, computed as a combination of R, G and B with positive coefficients, will always represent white lane markings with the highest values in the image, regardless of the particular illumination conditions. In the same fashion, given the construction of the Cb component (blue-difference chrominance), yellow lane markings are necessarily mapped to the lowest values in this dimension. With this two properties, they demonstrate that it is easy to filter out at a satisfactory degree of precision the pixels not belonging to a lane marking, identifying also the color of the markings thus isolated.

HSV

Applying instead a non-linear transformation to the RGB components we can obtain a very popular space for its invariant properties: the HSV space. As shown in [62], it is obtained by the following transformations:

$$\begin{aligned} H &= arctan\left(\frac{\sqrt{3}\,(G-B)}{(R-G)+(R-B)}\right) \\ S &= 1 - \frac{\min\{R,G,B\}}{R+G+B} \\ V &= \max\{R,G,B\} \end{aligned} \tag{3.4}$$

where H stands for hue, S for saturation and V for value. Figure 3.12 illustrates the morphology of this color space and clarifies the terminology. [62] also proves,

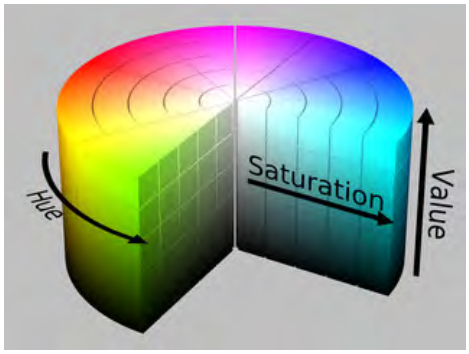


Figure 3.12: HSV color space, from [68].

using the mentioned Dichromatic Reflection Model, that the so constructed S and H dimensions are invariant to the surface orientation, illumination direction (viewpoint) and intensity. Moreover, H is also invariant to highlights due to specular reflections. We must additionally point out that a few variants of this color space are available with respect of the V dimension, at times substituted by the intensity (I) or the lightness (L), generating respectively HSI and HSL. For completeness, we report the definition of these dimensions, as they appear on [69]:

$$\begin{aligned} I &= \frac{1}{3}(R+G+B) \\ L &= \frac{1}{2}\left(\max\{R,G,B\} + \min\{R,G,B\}\right) \end{aligned} \tag{3.5}$$

L*a*b*

Another non-linear space is L*a*b*, designed to be compatible with human perception. Indeed, the channels of L*a*b* are obtained by a non-linear transformation on

the space XYZ [5, Chapter 3.3], a well-known space we decided not to analyze in our study. XYZ is one of the first color-spaces ever defined, and it is based on the concept of color matching functions [5, Chapter 3.3]. Each of its components X, Y and Z represent a possible primary color [5, Chapter 3.3]. While XYZ presents some interesting properties, that make it one of the most used color-spaces, we focus our attention on its non-linear derivation, L*a*b*. This is defined as:

$$\begin{aligned} L* &= 116 \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16 \\ a* &= 500 \left[\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}\right] \\ b* &= 200 \left[\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n}\right)^{\frac{1}{3}}\right] \end{aligned} \tag{3.6}$$

with X_n, Y_n and Z_n representing the XYZ coordinates of a reference white patch [5, Chapter 3.3].

The most interesting property associated to the space L*a*b* is that it is considered to be a uniform color space. This means that differences between two colors in L*a*b* represent good estimates of how different the two colors really are for the human eye.

Lane detection and illumination invariance

With an understanding of the different color spaces available and of their properties, we can finally analyze how to exploit them in our system. In the literature, [9] implemented a simple but effective illumination-invariant rule, based on the following considerations. Notice that we will use these results as a foundation for some of our

experiments.

They report that the RGB values of a pixel under two different illuminants are related by a linear transformation of the form

$$\begin{aligned} R' &= \alpha R \\ G' &= \beta G \\ B' &= \gamma B \end{aligned} \tag{3.7}$$

with α, β and γ linear coefficients. This means that the order relationship among pixel values is preserved, i.e. for any pixel i and j

$$\begin{aligned} R_i > R_j \implies \alpha R_i > \alpha R_j \implies R'_i > R'_j \qquad \forall i, j \\ G_i > G_j \implies \beta G_i > \beta G_j \implies G'_i > G'_j \qquad \forall i, j \\ B_i > B_j \implies \gamma B_i > \gamma B_j \implies B'_i > B'_j \qquad \forall i, j \end{aligned} \tag{3.8}$$

Since white pixels in RGB will have high values in the three components, we can thus expect that they will have higher values under any illumination condition.

Having said this, they then convert their images from the RGB space to YCbCr. At the basis of their considerations is the fact that the Y channel is computed, as we mentioned above, as a linear combination of positive coefficients of R, G and B:

$$Y = 0.29900\ R + 0.58700\ G + 0.11400\ B \tag{3.9}$$

Therefore, under any illumination condition, white pixels will present the highest Y values in the image. Thus, they show how, from the analysis of a histogram computed on the Y channel, it is possible to threshold and isolate the pixels likely to be part

of a white lane marking. We highlight how the threshold used is not fixed and it is instead computed considering the color distribution in each particular scene.

An analogous consideration is made for yellow markings in the Cb channel. Such channel is computed as a linear combination with negative coefficients for R and G and a positive one for B. The authors remark that the yellow color used in markings can be synthesized with high percentages of R and G, and none of B. For this reason, it is possible to set an adaptive threshold for the lower values in channel Cb and isolate the corresponding candidate pixels.

The use of color features is part also of the work discussed in [70]. They propose a probabilistic classifier that evaluates gradient-based and color-based features. The latter are computed starting from the HSV color space and taking into account its histogram distribution. The results are then used to estimate the likelihood of each configuration to be part of a lane marking.

Other works exploiting color features are often found on road detection, where more emphasis is put on highlighting the position of the pavement and the boundaries of unmarked roads. [71] presents a real-time shadow-removal technique for facilitating the detection of road segments. They show how a palette of colors distributes in particular linear patterns on a log–log chromaticity plot, and how this information can be exploited to re-project each color in a different feature space, invariant to shadow and illumination changes. They also point out how, to achieve this goal, an additional intrinsic parameter of the camera has to be estimated, but an off-line procedure is available.

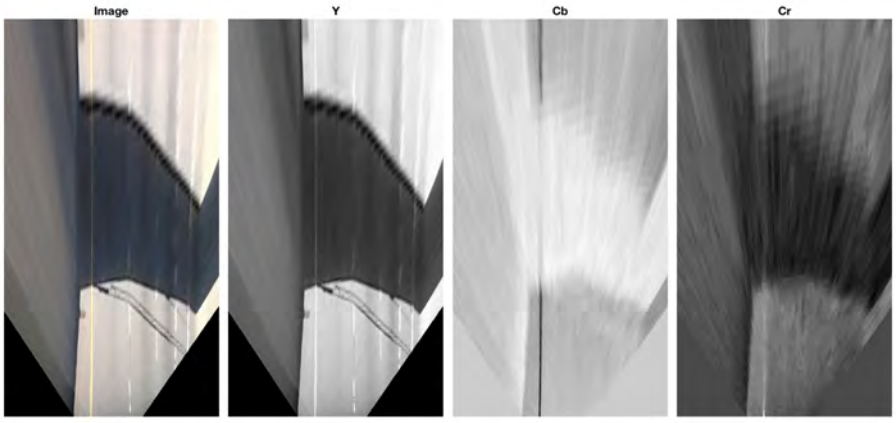
3.3.3 Methodology

Color spaces

As a first important step, we explore the color spaces described above on a set of test frames retrieved from our dataset. We analyze how the lines are represented in each color space in order to gain useful insights for the successive steps. Figure 3.13, 3.14, and 3.15 group a sample of the transformations analyzed, along with some considerations on the results of such experiments. We carried out our analysis on several image frames, but we report here our findings for only a single sample, in order to maintain this section contained.

We notice that, as expected given the considerations in 3.3.2, the channel Y and Cb of the color space YCbCr reveal interesting properties. Other minor features are detected, as explain in the captions, but are not taken into account in this work for their small potential implications in our research.

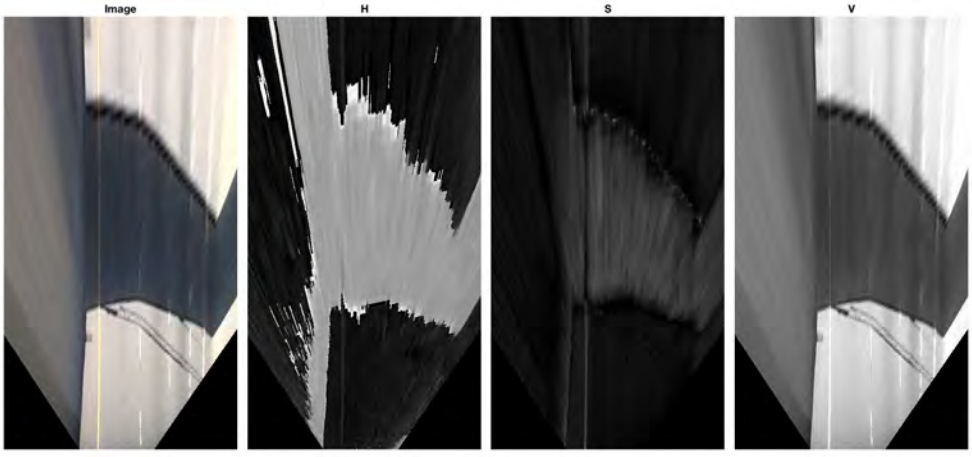
(a) Front view



(b) BEV

Figure 3.13: The space YCbCr is analyzed on an image presenting a strong illumination discontinuity. The image has been collected during our data collection drives and its analysis is performed on both its front view and BEV version. In both cases, the original RGB image is reported for reference. We can notice, as described in Section 3.3.2 how the white markings remain visible in the Y component, as well as the yellow line is clearly identifiable in the Cb image channel.

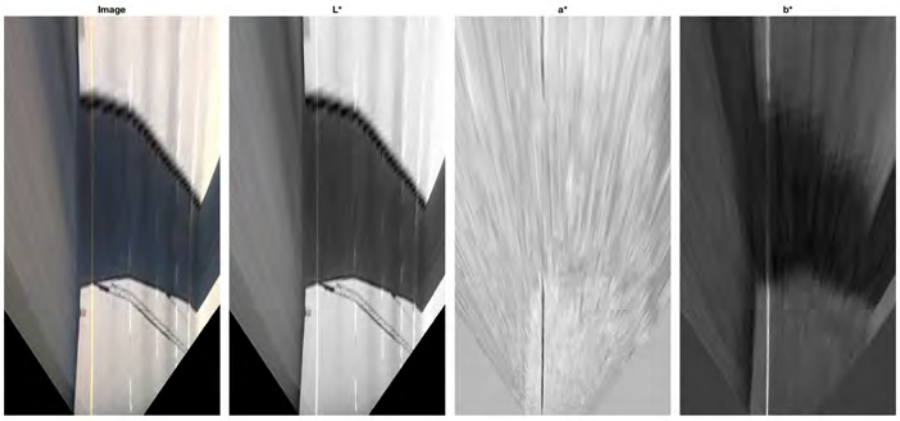
(a) Front view



(b) BEV

Figure 3.14: The space HSV is analyzed on the same image seen in Figure 3.13. As above, we report both the front view and BEV component, and for both cases the reference image in the RGB space. We don't notice any interesting feature from the analysis of the image, sign that a transformation to this color space will be less interesting for our purposes.

(a) Front view



(b) BEV

Figure 3.15: The linear space L*a*b* is analyzed on the same image seen in Figure 3.13. As above, we report both the front view and BEV component, and for both cases the reference image in the RGB space. We notice how, although not perfectly, the b* channel seems to preserve a good representation of the yellow markings despite the illumination changes. However, we find it still difficult to extract its precise position, and for this reason this feature will not be used in our current pipeline. It could be certainly considered for future works.

Hard threshold

Given the mentioned properties of the Y and Cb channel, we then experiment different mechanisms of employing them to properly detect the yellow and white lane markings.

As a first discrimination technique, we start from channel Y and apply a simple hard threshold. Formally:

$$\mathrm{Wh} = Y > \mathrm{th}_{\mathrm{wh}} \tag{3.10}$$

where we define as Wh the white mask, and as $\mathrm{th}_{\mathrm{wh}}$ the fixed threshold value applied. The $>$ operation is intuitively defined as a pixel-wise thresholding operation on the image. We select for this experiment different frames, each with different illumination characteristics. In the case here described, the images are collected on a highway road during a sunny day. The road is crossed by some overpasses, and we study the diverse illumination conditions generated by their shadows on the road. Our main analysis is carried out by setting several hard thresholds on each image and registering the different responses. Figure 3.16 illustrates the process and helps us draw some conclusions on the quality of this basic technique. The three frames compared capture respectively a clear road, the presence of a shaded area in the distance and the presence of a shaded area in proximity of the vehicle.

The most evident result is clearly that, for the three illumination conditions studied, the road markings are visible only for specific threshold values, very dissimilar from each other. This problem alone makes it already hard to apply this methodology in our final pipeline, since it would totally undermine its robustness to illumination changes. We notice also how sensitive the detection of the markings is with respect to changes in threshold value: as shown in the picture, a small variation from a working threshold can cause the system to miss all the lines.

Analogous results could be obtained from the Cb channel, in search of the yellow markings, as:

$$\mathrm{Yw} = \mathrm{Cb} < \mathrm{th}_{\mathrm{yw}} \tag{3.11}$$

where the yellow mask is represented as Yw.

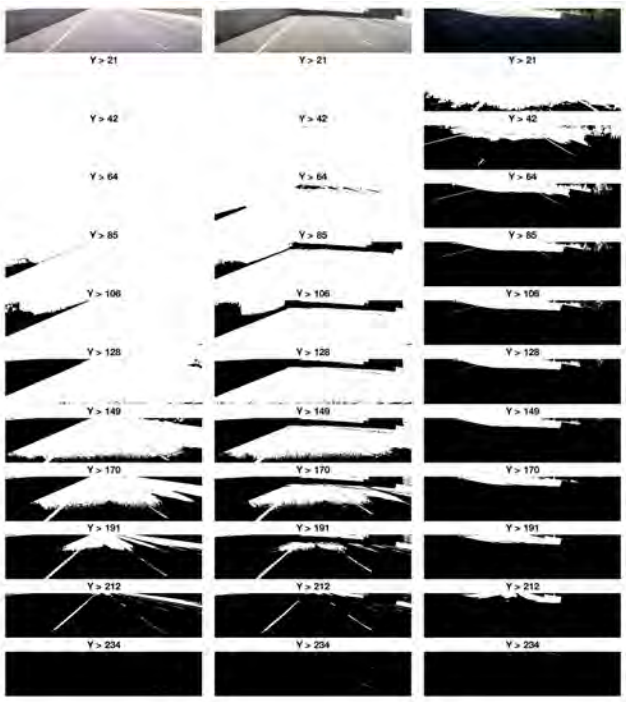


Figure 3.16: Results of the experiments on hard thresholding for the detection of white lane markings. The experiments are performed on three distinct frames (from left to right): in completely uniform illumination; in presence of a shaded region distant from the vehicle; in the middle of a shaded region, but with part of the pavement exposed to direct sunlight. For each frame, different hard thresholds are applied to the Y channel, with a linearly distributed pace over the entire range of pixel values (0–255). The resulting masks are displayed, along with the threshold value used to generate them. We can clearly notice how the first two frames, both captured in sunlight, have very large pixel values, and as a consequence respond only to high threshold values (i.e. th_{wh} above 170, more than half of the range). In addition, the road markings are clearly detectable only for thresholds in 191–212, and disappear soon after. This is a very small interval, showing how sensitive this operator could be. In an opposite fashion, the shaded image is characterized by low values of Y. As expected then, the markings are roughly visible only for low thresholds (i.e. th_{wh} in the range 42–64). This clearly highlights how hard thresholding is not suited to cope with illumination changes, unless proper preprocessing is carried out beforehand. Moreover, we already pointed out how the detection of markings is very sensible to small changes in the threshold value, and this could easily become an additional obstacle.

Adaptive threshold with cumulative histograms

It is thus clear that a more specific filtering procedure is needed. We resort, as a starting point, to the adaptive thresholding used in [9] and mentioned in our previous Section 3.3.2.

The image is first transformed into the YCbCr color space. Here, two analogous procedures are applied to the Y channel to extract white markings and to the Cb channel to extract yellow markings. We consider for the moment only the former. An histogram is constructed aggregating pixels according to their values. Cumulating the bins and normalizing the result between 0 and 1, we obtain the so called cumulative histogram. This representation can be also interpreted as the cumulative density function of the probability distribution of pixels with respect to different color intensities. Only at this point, a thresholding operation is performed, but on the histogram. Fixed a threshold percentage Tp, only the top Tp % of the probability mass in the cumulative histogram is kept, resulting in a strongly thresholded image. For the properties enunciated in Section 3.3.2, the mask thus obtained contains only pixels likely to be part of white lane markings. We notice, as mentioned by [9], that because of the indirect application of the threshold, this methodology results in significantly more accurate detections and in partial illumination invariance.

We performed the described computations on different test frames and evaluated the results with particular consideration to their response to different illumination conditions. We report here in Figure 3.17 the results of our analysis on white lane markings in scenarios with and without shaded pavement regions.

An analogous procedure is followed to separate the yellow markings and it is illustrated in Figure 3.18.

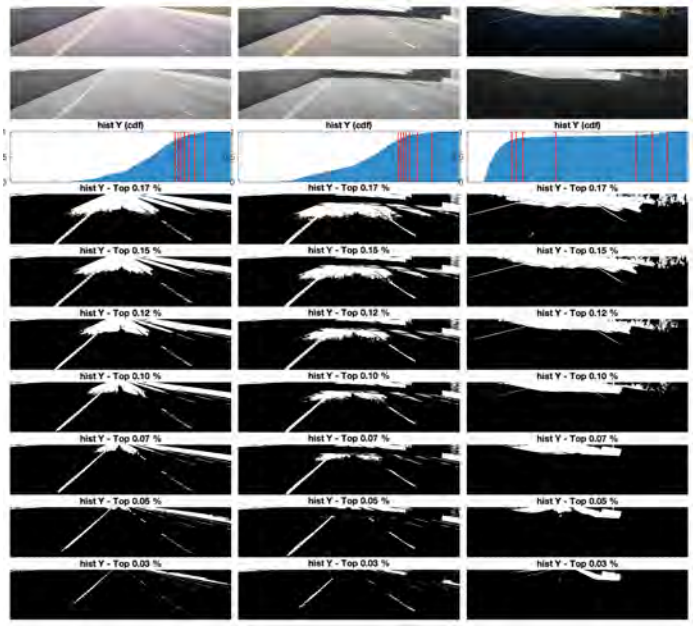


Figure 3.17: Results of the experiments on adaptive thresholding based on the cumulative histogram method. For coherence, the analysis is performed on the same three frames used in Figure 3.16. For each frame, we first display its Y component in the YCbCr color space. Then, the cumulative histogram is shown, with red markings indicating the applied thresholds. Below, finally, the masks obtained from each threshold are shown, each with its associated threshold value. We notice first of all, especially for the clear images, that there is no high sensitivity now, and almost any choice of the threshold is able to detect the markings. For low thresholds, clutter appears in the far away part of the image, while the near end is always fairly clear. Secondly, we move to analyze the distributions depicted by the histograms. The two clear images present a cumulative profile similar to the CDF (cumulative distribution function) of a Gaussian, indicating that pixel values are more or less normally spread. For the third image instead, a big peak in the lower bins is represented by the large initial slope, which flattens very soon. Nevertheless, we can still see a slight increase in the histogram corresponding to very high-valued bins. Those are most likely to represent white marking pixels. It is for this reason that, even under this uneven distribution, this adaptive threshold based on the histogram performs significantly better than a hard threshold. Indeed, we can see that for thresholds around $0.15 \div 0.17$ % most of the white markings are visible also in the shaded image. Since also the correspondent masks for the other two frames have satisfactory performances,, we realize that setting a threshold around such interval would represent a good trade-off of accuracy between unshaded and shaded scenes.

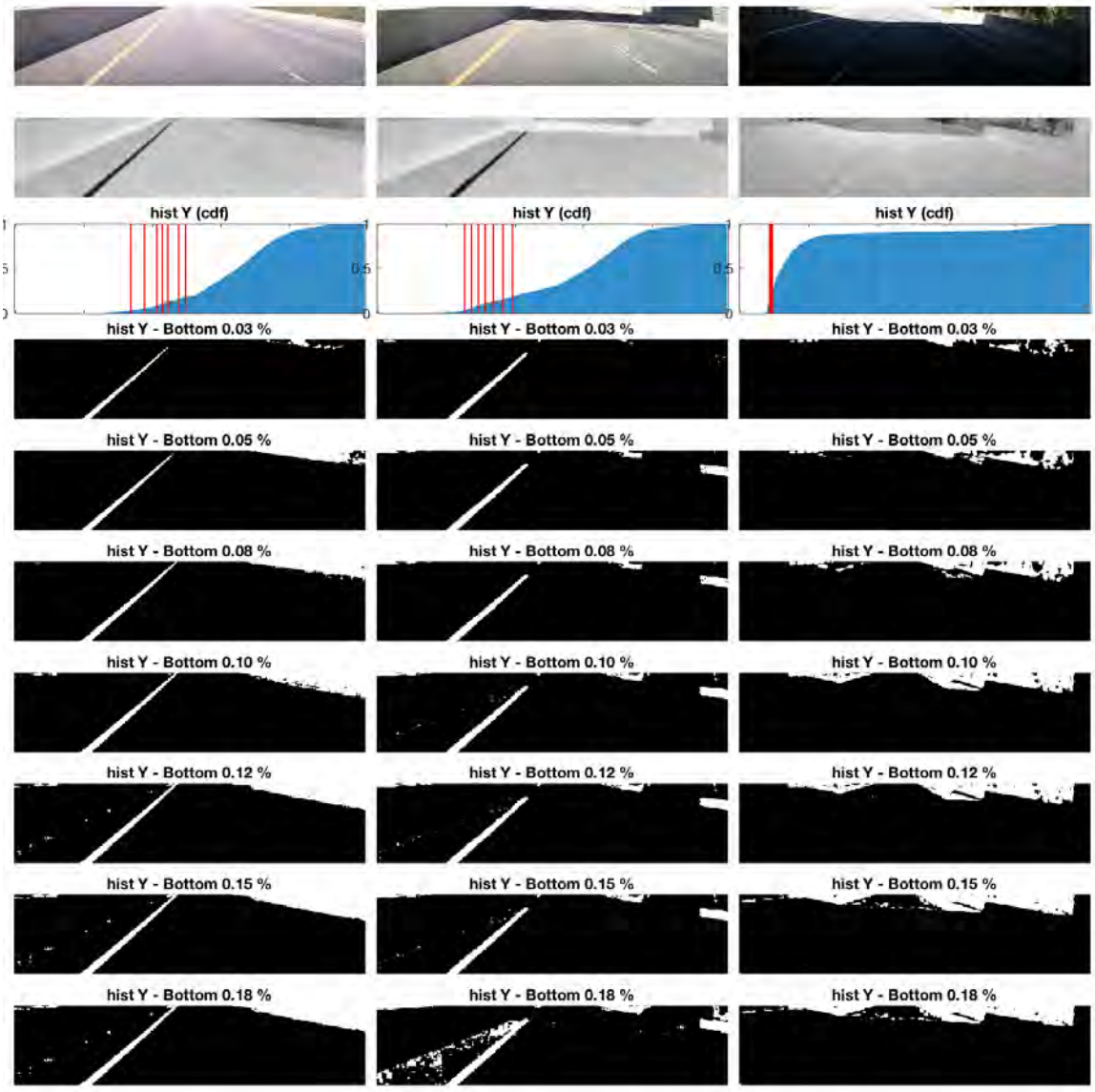


Figure 3.18: As in Figure 3.17, the results of the experiments on adaptive thresholding based on the cumulative histogram method are shown. The thresholds are set here to retrieve yellow markings. We notice how, although the robustness of the algorithm shown for the detection of white lines, here it is very hard to retrieve the yellow ones in the shaded image. It is not shown, but even further increases of the threshold (i.e. above 0.18%) does not achieve the expected results. The algorithm is still however very effective in sunlight conditions.

Morphological manipulation

In principle, the image masks obtained above could present clutter and spurious detections that have to be removed. For this reason, we experiment with a series of morphological operations, in order to find the best combination that fits our purposes.

At a closer analysis, the mask for the white markings (Figure 3.19) does not present significant false detections on the pavement and achieves an acceptable precision for this phase. The major noticeable flaw is instead the high rate of false detections in regions of the image outside the pavement. This is probably due to how close the reflectivity of the pavement at a far distance is to the one of the concrete surrounding the road. Nevertheless, we could in principle identify this artifacts by studying the single view geometry of the scene. Since our camera is calibrated, we could easily compute the world position of each of these points (assuming they are on the ground), and narrow or enlarge our ROI boundaries accordingly.

Figure 3.19: The raw white mask (Wh) obtained at the previous stage.

For what concerns the identification of yellow markings instead (Figure 3.20), the

detection of false positives is quite frequent and we thus proceed with further attempts to reduce their effect. In order to correct it, we devise three alternatives, which will be presented and analyzed in the following.

Figure 3.20: The raw yellow mask (Yw) obtained at the previous stage.

As a first idea, we notice that the white mask detects also most of the yellow lines. We don't consider these strictly as false positives, since we can always use the information in the yellow mask to easily filter them out. Moreover, it is more important for us to reliably localize all the line markings than to distinguish their color (and associated semantic interpretation). With this in mind then, we can actually think to rely on this detections to correct the yellow mask: we first create a dilated version of the white mask using a circular structuring element and then remove from the yellow mask any detection not belonging to it. Formally:

$$\mathrm{Yw}_{\mathrm{c1}} = \mathrm{Yw} \wedge \mathrm{dilation}\,(\mathrm{Wh}) \tag{3.12}$$

where $\wedge$ is defined as a pixel-wise logical AND. An overall idea of the process is given by Figure 3.21.

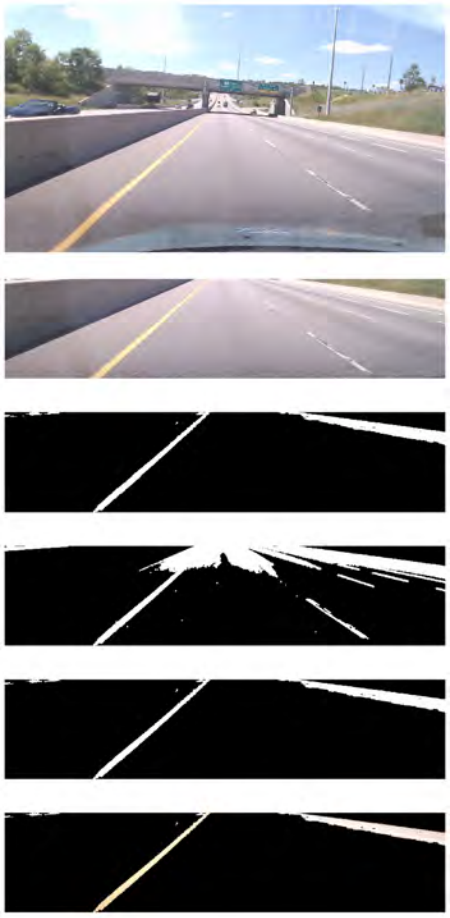


Figure 3.21: Pipeline for the correction of the yellow mask exploiting the properties of the white mask previously generated. In order, the following are shown: the original frame, our ROI, Wh, Yw, Yw_{c1} and finally the original frame filtered by Yw_{c1}. We can see how clear the left yellow markings appear. Some false detections are also present at the top, but they are located on region external to the road, and can therefore be filtered out in a successive step.

As a second option for correcting Yw, we try to focus on reducing the impact of isolated and nearly-isolated false detections. We do so in two ways. On one side, we apply a morphological opening to the image through a square structuring element,

operation that typically removes small and irregular components from the mask. Successively, to enhance even more the relevant information in it, we create an auxiliary mask to be applied to our Yw as a filter. This mask is the dilation of an opening of the original yellow mask, both using a circular structuring element. The rational behind this choice (which is equivalent to an erosion followed by two successive dilations), is to identify big blobs of noisy detections in the mask, exaggerate them and finally remove them. We must point out that to do so the size of the structuring element used is always chosen large enough to guarantee the filter never contains any lane marking line. The described steps are depicted in Figure 3.22. Formally, the computation can be expressed as

$$\begin{aligned} \mathrm{Yw}_{\mathrm{open}} &= \mathrm{opening}\,(\mathrm{Yw}) \\ \mathrm{Yw}_{\mathrm{filter}} &= \mathrm{dilation}\,(\mathrm{opening}\,(\mathrm{Yw})) \\ \mathrm{Yw}_{\mathrm{c2}} &= \mathrm{Yw}_{\mathrm{open}} \wedge \neg\, \mathrm{Yw}_{\mathrm{filter}} \end{aligned} \tag{3.13}$$

where $\wedge$ and $\neg$ are defined as a pixel-wise logical AND and NOT respectively.

Finally, the third method we devise relies on an attempt at identifying the pixels belonging to the road. With this information, it is possible to remove false yellow marking detections occurring in areas where no pavement is present. In this approach, first a road mask is created, exploiting properties of the S and V channels in the HSV color space. With different parameters, both channel undergo first a hard thresholding (lower values), and then a morphological opening. In the end, an approximated road mask is created as the logical OR of the two, i.e. where either of the channel S or V

have low values, and is used as an image mask (Figure 3.23). In our notation:

$$\begin{aligned} \mathrm{R_s} &= \mathrm{opening}\,(\mathrm{Yw} < \mathrm{th_S}) \\ \mathrm{R_v} &= \mathrm{opening}\,(\mathrm{Yw} < \mathrm{th_v}) \\ \mathrm{R} &= \mathrm{R_s} \vee \mathrm{R_v} \\ \mathrm{Yw_{c3}} &= \mathrm{Yw} \wedge \mathrm{R} \end{aligned} \tag{3.14}$$

A comparative analysis of the three correction methods is presented through some examples in Figure 3.24. As it is there highlighted, the first correction method, based on the white mask, is the most effective, and is indeed the one we decide to adopt. The second method strongly competes with it, but we disfavor it for its reliance on fixed morphological operations, which are more likely to not generalize in different scenarios.

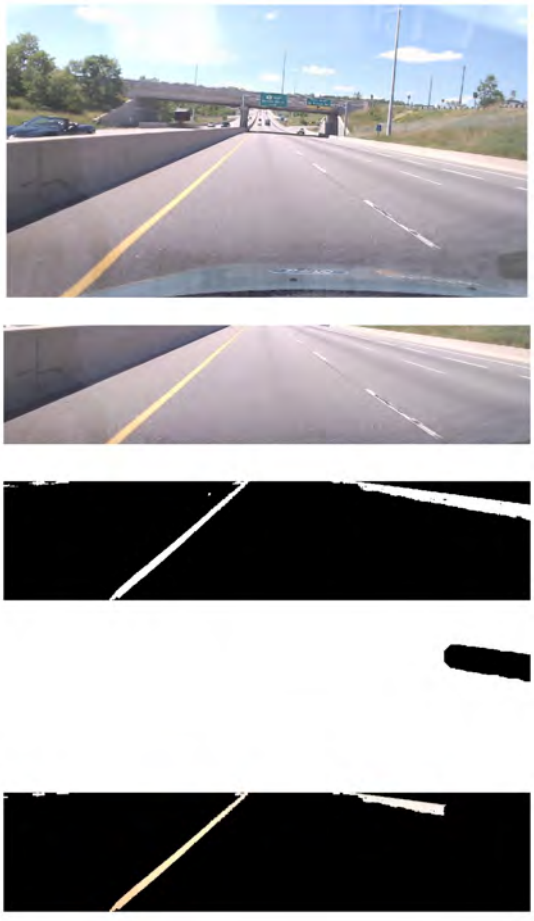


Figure 3.22: Pipeline for the correction of the yellow mask by morphological opening and masking. In order, the following are shown: the original frame, our ROI, Yw, the mask $\neg \mathrm{Yw}_{\mathrm{filter}}$ and finally the original frame filtered by $\mathrm{Yw}_{\mathrm{c2}}$. We can see that the left yellow marking appears clearly also here, and that some noise on the right, although not particularly relevant, has been removed.

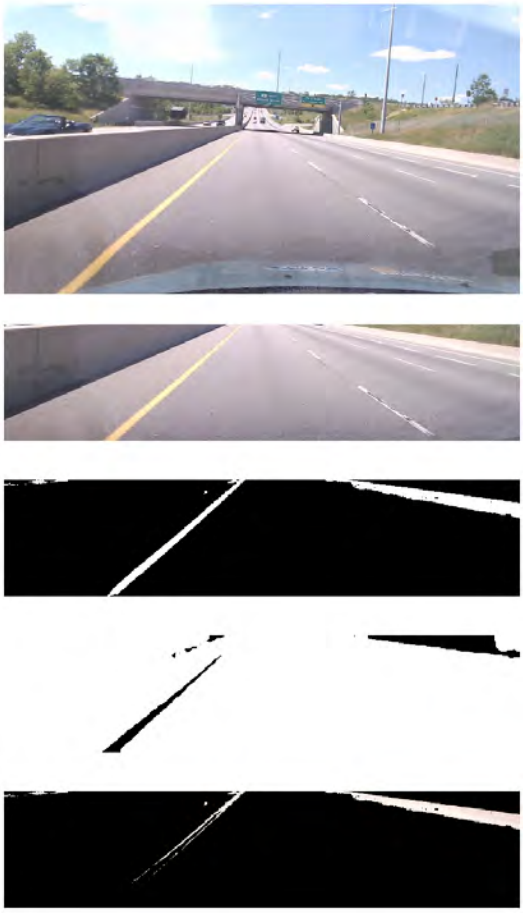


Figure 3.23: Pipeline for the correction of the yellow mask by morphological by usage of a road mask. In order, the following are shown: the original frame, our ROI, Yw, the road mask R and finally the original frame filtered by Yw_{c3}. We notice how this technique is not particularly effective due to the removal of the yellow markings in the road mask. It results very hard to cope with this issue without opening too much the road mask.

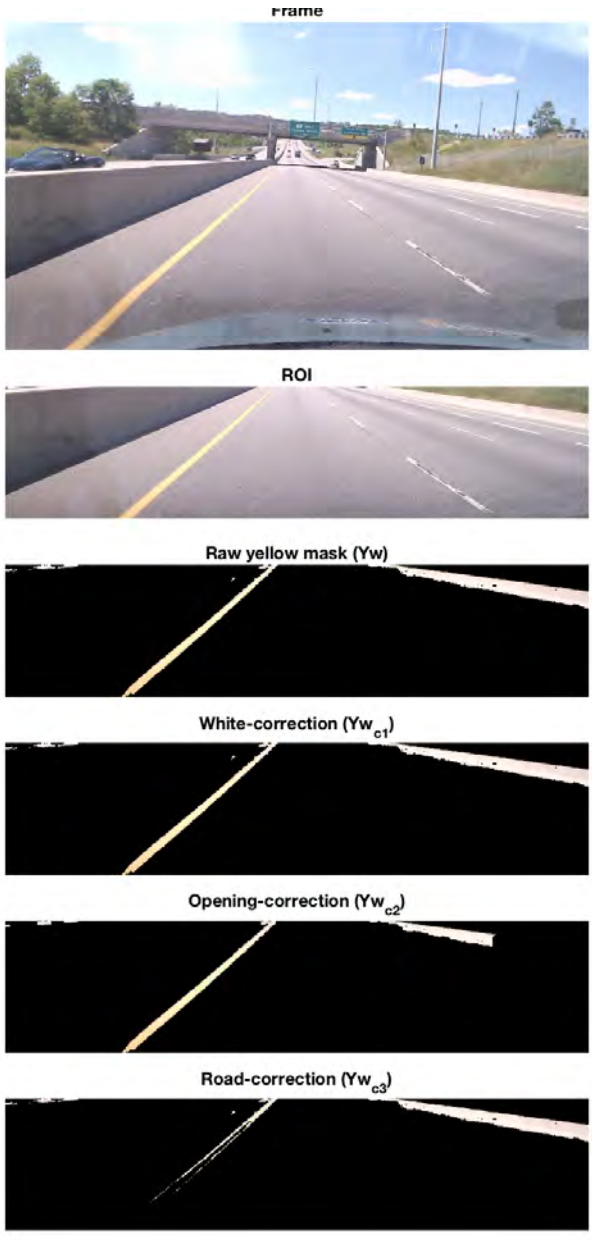


Figure 3.24: A comparison of the three correction approaches for the yellow component.

3.3.4 Results and evaluation

During the construction of our feature extraction technique, we tuned the involved parameters and selected the most appropriate operations by comparing the results on different road images. These had been previously collected with our equipped vehicle specifically for this purpose. Running the algorithm on few of such acquisitions, in Figure 3.25, 3.26 and 3.27 we display the results we obtained in order to perform a first evaluation.

Figure 3.25: Final results of our pipeline on highway roads. We notice how the pipeline doesn't have enough power to cope with abrupt changes in illumination, as highlighted by the third frame processed. Instead, appreciable results are obtained for frames under direct sunlight illumination. For them indeed, the amount of noise located on the pavement is significantly contained and concentrated in areas distant from our vehicle.

We can see how our technique is able to identify the lane markings with a reasonable accuracy when the road is brightly illuminated and doesn't present a large illumination excursion. The markings are almost perfect in the short distance, and

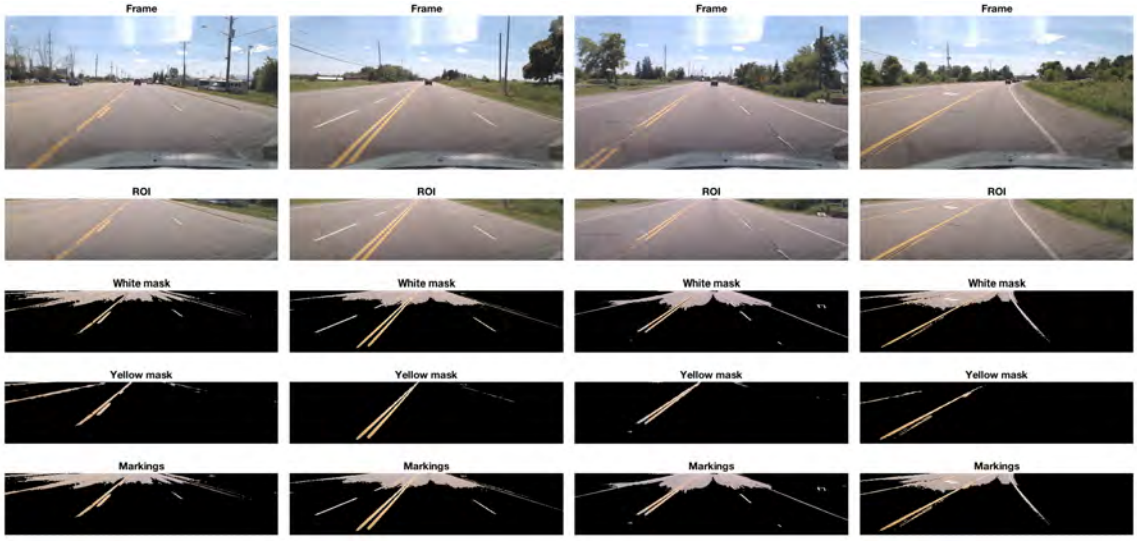


Figure 3.26: Final results of our pipeline on an interstate road. As in Figure 3.25, the pipeline is successful and noise is present only in remote areas of the scene.

become noisy as far as we move from the vehicle. This is due to the use of a relaxed threshold value (see Figure 3.17), in an attempt to confer robustness to the algorithm with respect to illumination differences. Unfortunately however, we are still not able to cope with abrupt illumination changes. These situations can occur, for instance, in the proximity of overpasses and similar road elements, due to their cast shadows. Nevertheless, we expect our algorithm to adapt and behave correctly if the variation in illumination conditions persists for a longer time, as it happens in road tunnels crossing a mountain, for example. Unfortunately however, the location in which all our tests have been performed didn't allow us to validate this claim, and we reserve it for a future work.

Important to realize at this point is that the evaluation presented up to now is performed only on frames available to us at the moment we devised our algorithm. Since we performed our tuning on such data, by assessing our methodology on them

Figure 3.27: Final results of our pipeline on urban roads. The algorithm detects most of the interesting features, although some noise is created by tree shadows and other vehicles. Notice that some noise is also present outside the pavement, but that such cases are not relevant for our analysis as they can be removed in subsequent steps. Indeed, geometric considerations can be used to estimate their position as external to the road.

only we would risk to introduce a strong bias in our analysis. We need instead to support our findings with further evidence. To do so, we apply our procedure to a new dataset, collected at a second time and with the sole purpose of testing our existing algorithms. The recordings contained in it were collected in an urban area never explored with our vehicle, and in a different day. It is clear that this cannot completely prevent the risk of overfitting and bad generalization, but we expect this consideration to significantly reduce their effect, if present. We use Figure 3.28 to show the road marking features we retrieved on them.

We can see that the algorithm behaves similarly on our test frames as it did on the images used for its tuning. The parameters chosen are then sufficiently general to produce good results also on unseen recordings, provided they are captured in

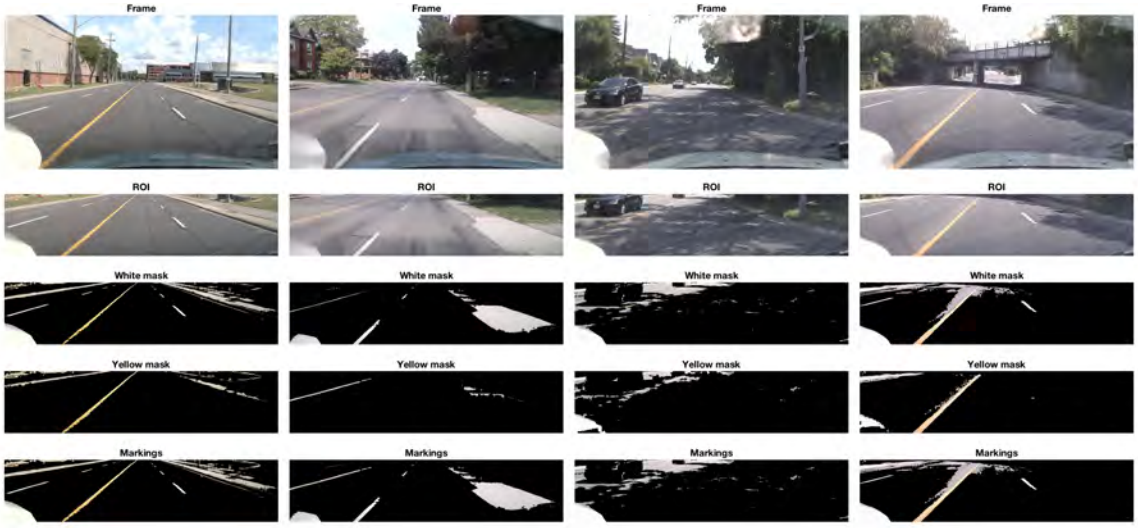


Figure 3.28: Samples of the results achieved by our pipeline on new images. The frames displayed are part of an evaluation set never used during tuning, and captured once the algorithm had already been built. We notice how the pipeline produces highly satisfactory results on the first, second and fourth frames, where the road is mainly exposed to sunlight. Unfortunately, we must confirm that, as expected, the algorithm doesn't generalize enough to cope with abrupt illumination differences, as shown in the third frame presented.

daylight. As we already knew from the above evaluation, noisy detection are present in remote areas of the scene, and the algorithm is not robust to cope with dramatic illumination changes. Interestingly enough, the second frame presented in Figure 3.28 shows a curious false detection: the sidewalk has been treated as a white marking because of its similar chroma and luminance in that particular frame. We could argue that such detection could be filtered by a careful analysis of its location in the world scene, but its proximity to the pavement would probably make it hard to find a general solution to this problem. Another argument in favor of its easy removal could be based on its disproportionate size, compared to the dimensions we expect road lines to have. Either way, what is important for us is only to be aware of the risks false detections introduce, and acknowledge their presence in order to design, in

the future, proper solutions or necessary countermeasures.

In conclusion to our evaluation then, the pipeline proposed is able to distinguish lane markings in a numerous amount of cases. Instances of noisy detection are of course present as well, and we acknowledge this fact and consider it for future improvements. Nevertheless, in the following stages of a lane detection system, these features will have to be fit to a particular model. With this in mind, it can be argued that, although precise features are often crucial to succeed, the presence of a robust model is, instead, always of extreme importance. Indeed, without its robustness, we wouldn't be able to achieve a good performance because of the very intrinsic nature of the problem, based on partial and noisy information. These issues are often due to missing frames, noise in the acquisition process and limited number of sensors for perceiving the environment, but certainly also include the incorrect estimates from the previous stages of the pipeline. We can say therefore that, within some boundaries, inaccuracies in the feature extraction phase should not doom the complete system to failure, if an appropriate model fitting algorithm is constructed.

For all these reasons, we are finally in favor of considering the methodology presented as overall successful.

3.3.5 Conclusions

In this section, we described our analysis of the feature extraction problem and how we can retrieve the position of the lane markings in each frame by exploiting the color information contained in it.

To do so, we first described the different representations of color images available and defined the properties of several color spaces. We discovered that some of them,

as the YCbCr space, suggest an illuminance invariance property, which could be exploited to discriminate the road markings from the rest of the scene.

As a second step, we proceeded at performing different experiments. At first, we tried to identify white and yellow lane markings by means of a hard threshold, but this approach didn't succeed, because of its low expressive power.

For this reason, we then studied an adaptive thresholding technique based on the YCbCr space, and highlighted how it displays a degree of stability with respect to its parameters in case of sunlight illumination. We also saw how this is not true under poor illumination conditions, and how in these cases there is only a very narrow and context-dependent range of parameter values for which the lane markings are detected.

As a further step, once tuned the algorithm to achieve a fair trade-off among different illumination scenarios, we tried to improve the quality of the detections. This was done by applying several morphological operations and comparing their results.

At last, we evaluated our pipeline. We first presented the detections obtained on known samples. To further support the evidence found, we then processed images never seen and recorded solely for testing purposes. The results are satisfactory in presence of uniform illumination or direct sunlight, but their quality becomes inadequate in case of abrupt illumination changes, due for example to overpasses. Nevertheless, this issue can be partially attenuated with a robust model fitting technique and possibly smoothed with a tracking stage.

For all these reasons, we can finally conclude that the algorithm has good overall performance, and we will indeed apply it in our lane detection system.

3.4 Feature selection for line markings fitting in Bird's Eye View

3.4.1 Introduction

When computer vision is adopted to discover some parametric structure in the images, a usual pipeline would go through two crucial steps: the extraction of relevant features and their usage to estimate the desired parameters (i.e. the model). Whichever the feature selection technique adopted is, the retrieved data points always present two types of impurity: *noise* and *outliers*. In order to obtain a robust fitting, these must be filtered out. In other words, it is necessary to select, among the maybe numerous features, only the most reliable ones.

This selection process is exactly what is treated in this document, with particular reference to one way of performing it devised for our specific domain. Starting from feature points close to the car, and moving forward in the Bird's Eye View image of the road, such method will try to engage and follow the points on the lane markings rather than guess their true position, by exploiting some assumptions on the smoothness of the lines and on the pose of the vehicle observing them.

3.4.2 Background and Previous work

Many computer vision applications rely on the notion of a model. A model is usually, in this context, a representation of some phenomena of interest in the scene, such as the presence of geometric elements and their characteristics, a description of the motion of an object or a perspective transformation underlying some visual effect. The model fitting problem is then, not surprisingly, the process through which these

characteristics of the scene are estimated, starting from some observed features (i.e. pixel intensity and color, presence of an edge or a corner, and so on).

Although the large variety of such problems, and despite different instances have been proven more compatible with particular algorithms, a common background unites them all.

A first shared feature regards their representation. As presented in [2, 3] indeed, some models can be described by a finite set of parameters, and in such case they are denoted as *parametric models*. Examples of this type of models are a polynomial curve, a conic or a homography to another image plane. At the other extreme, we find instead models represented by unknown functions or, equivalently, by an infinite number of parameters, and denoted for historical reasons as *non-parametric models*. This enables them to describe very complex or unpredictable scenarios with more freedom, at the cost of harder convergence. Furthermore, an additional third class recently defined constitutes the *semi-parametric models*, where part of the model can be described as parametric, and part as non-parametric.

With this in mind, the fitting problem reduces to an optimization procedure for the parameters $\mathbf{m}$ with respect to the features $\{\mathbf{x_i}\}_i$. For our purposes, we will consider only pure parametric models, i.e. $\mathbf{m} = (m_1 \ \dots \ m_k)^T$, but our discussion can be easily adapted to the case of semi- and non-parametric ones. Let's first define a function $f = f(\mathbf{m}, \mathbf{x_i})$ that characterizes the model, which is

$$f(\mathbf{m}, \mathbf{x_i}) = 0 \qquad \text{iff} \qquad \mathbf{x_i} \in \text{model}$$

We can think for example at a straight line in homogeneous coordinate, $r\colon ax + by + cw = 0$, where $\mathbf{m} = (a \ b \ c)^T$, $\mathbf{x_i} = (x_i \ y_i \ w_i)^T$ and $f = f(\mathbf{m}, \mathbf{x_i}) = ax_i + by_i + cw_i$.

We know that if a point $\mathbf{x_i}$ belongs to the line, then $f(\mathbf{m}, \mathbf{x_i}) = 0$. If the point instead does not belong to the line, but it is very close to it, as if perturbed by some noise, then $f(\mathbf{m}, \mathbf{x_i})$ will not be zero, but very small; on the contrary, if it is very far apart, as an outlier, $f(\mathbf{m}, \mathbf{x_i})$ will be significantly large. We can therefore treat f as a measure of *how much* a feature belongs the model (or equivalently, the model fits the feature).

With this result, we can assess how a model fits some data by computing an aggregate measure of this quantity with respect to all the data points. This is usually a p-norm, which is:

$$\|\mathbf{f}(\mathbf{m}, \mathbf{X})\|_p = \| \left(f(\mathbf{m}, \mathbf{x_1}) \ \dots \ f(\mathbf{m}, \mathbf{x_n}) \right)^T \|_p$$

with $p \in [1, +\infty)$. The best fit would lead of course to $\|\mathbf{f}(\mathbf{m}, \mathbf{X})\|_p \approx 0$.

Given this setting, important for understanding the following, we leave aside the determination of f and we concentrate on how to find some features points $\mathbf{x_i}$ that facilitates the model fitting procedure. This is crucial because, although a degree of robustness must be present in the fitting itself, without a proper restriction of the search space, its optimization would be intractable. In particular, what significantly increases the computation needed is the presence of many outliers. When the fitting algorithm doesn't know which features are inliers, it has to first determine it, and this problem has intrinsically a super-exponential complexity. Some probabilistic approaches can be devised, such as RANSAC, to reduce this complexity, but in order for them to succeed in a reasonable time the relative number of outliers must still be contained.

In our case, exactly to avoid such complexity, and baring moreover in mind that

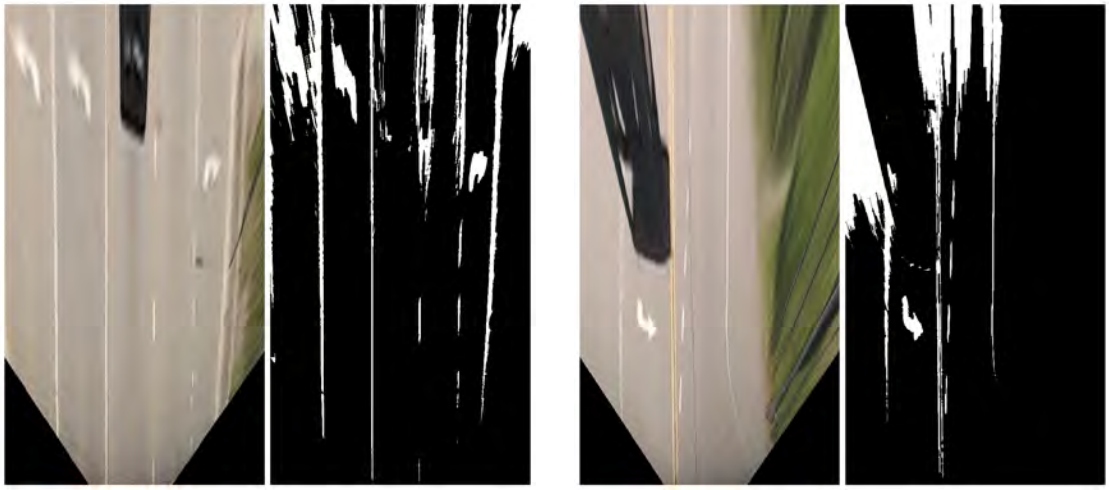
applications like ours are usually deployed on a real-time environment, we devise an algorithm that distills only few, highly reliable, feature points.

Our idea is based on the work found in [50], where the raw features are processed in the BEV and few points per lane marking are selected with an incremental approach. They start from the bottom of the BEV image (i.e. from the portion of the road closer to the vehicle) and search for the best line candidate point within a rectangular box positioned at a specified abscissa. Found such starting point, the box is moved upwards and centered with respect to it. The procedure then continues following the line on the pavement until an edge of the ROI image is met.

Other methods for filtering and refining the feature points before fitting the model can be observed in [33], where a line segment search is performed by connecting nearby feature points and subsequently applying non-maximum suppression. Others, as presented in [34, 72], employ a distance transform as an intermediate step to retrieve the mid-lane line and other points of interest for the fitting. Finally, experimental results have also been found by [73] applying clustering algorithms defined on fuzzy points.

3.4.3 Methodology

Inputs for our algorithm are the feature points extracted through adaptive color thresholding, as described in Section 3.3. We notice from Figures 3.29a and 3.29b that, although the feature extraction algorithm has a certain degree of robustness to noise, some outliers are still present and the markings are not perfectly outlined. For this reason, taking inspiration from [50], we search and retrieve here only some particularly interesting points among them, as mentioned in Section 3.4.2.



(a) occlusion (b) lane split and clutter

Figure 3.29: Examples of the issues faced with our feature extraction algorithm based on adaptive color thresholding. 3.29a shows the results when another vehicle is present on the road and occludes the sight of some markings. 3.29b instead depicts a situation with clutter in the distance and a lane split in the center of the image that could potentially mislead our line fitting.

We define $\mathbf{X_l}$ and $\mathbf{X_r}$ to be the starting points for our algorithm, referred respectively to the expected positions of the left and right lines in the proximity of the ego-vehicle. They are assumed to be expressed in the world coordinate frame and, if the camera is calibrated, can be successively converted into the corresponding image points $\mathbf{x_l}$ and $\mathbf{x_r}$ for the Bird's Eye View. In the following we will focus for simplicity on $\mathbf{x_l} = (r_l, c_l)^T$, but analogous results could be shown also for $\mathbf{x_r}$.

We consider a rectangular flat neighborhood of width w and height h, centered in $\mathbf{x_l}$, which we call *window* W (Figure 3.30a). Formally:

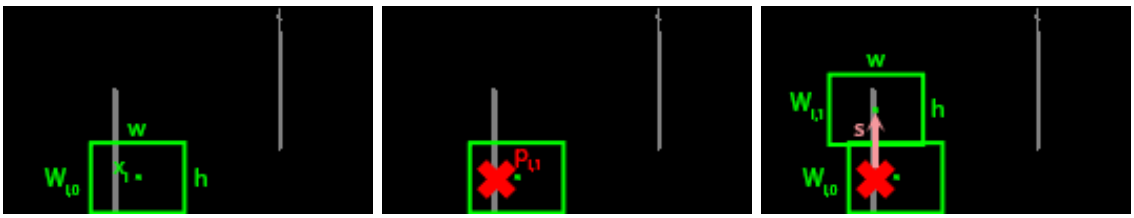
$$W_{l,0} = W(\mathbf{x_l}) = I\left[r_l - \frac{h}{2} : r_l + \frac{h}{2} \quad ; \quad c_l - \frac{w}{2} : c_l + \frac{w}{2}\right]$$

with I being the image under examination. Since this algorithm assumes to work

with BEV images only, it is implied that I is a BEV representation.

At this point, we determine the barycenter of the set of feature points within the neighborhood W (Figure 3.30b) and store its coordinates in a list of *interesting points* $P_l = \{\mathbf{p_{l,i}}\}_\mathbf{i}$, $\mathbf{p_{l,i}} = \left(x_{\mathbf{p_{l,i}}}, y_{\mathbf{p_{l,i}}}\right)$. The window is then re-centered with respect to $x_{\mathbf{p_{l,i}}}$ and moved forward, ready to repeat this operation and retrieve the successive *interesting point* $\mathbf{p_{l,i+1}}$. The step increment is kept fixed and named s (Figure 3.30c). We can formally represent this advancement as:

$$W_{l,1} = W(\mathbf{p_{l,1}}) = I\left[W_{l,0} + s\ ;\ W_{l,0}\right]$$



(a) (b) (c)

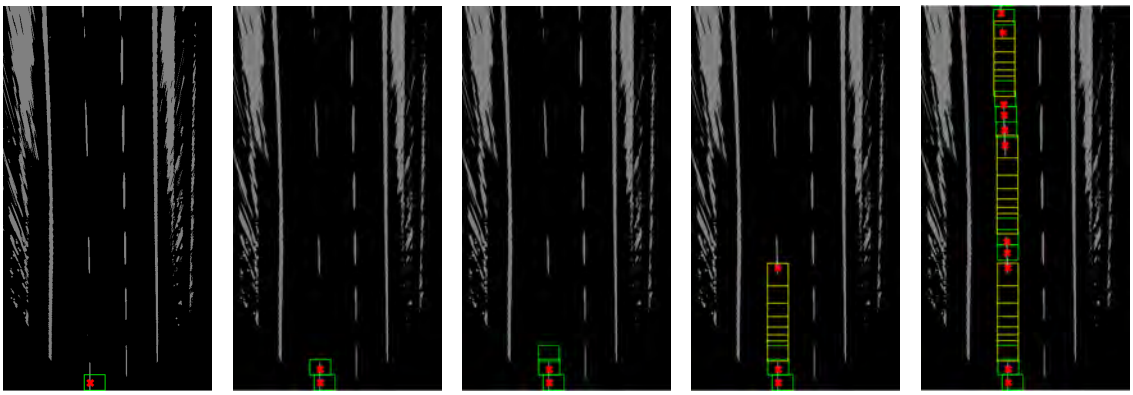
Figure 3.30: The definition of the neighborhood window and its step update are depicted. First, initial window $W_{l,1}$ is constructed around the initial point $\mathbf{x_l}$ (3.30a). Then, in 3.30b, the centroid of the cluster it contains is computed. Finally, the window is shifted forward of an increment s (3.30c).

The process continues selecting points $\mathbf{p_{l,i}}$ until one edge of the image is reached. If at some point not enough features are found within a window to build a reliable estimate of its centroid, a fall-back procedure starts. The dimensions of the neighborhood are increased, with particular emphasis on its height h, and the computation is performed again. The enlargement is reiterated until enough feature points are found inside the window and a centroid can be computed. Once an interesting point is finally determined, the original window size is restored and the procedure continues

from there.

At the end of each run, only the selected interesting points are stored and sent to the next stages of the pipeline. Additional information can also be collected, such as the number of times the fall-back procedure was called and on which areas of the image. This can be used for example to discriminate the type of road markings (i.e. solid or broken).

Figure 3.31 describes the steps illustrated above, while Figure 3.32 reports an example of the results achievable. We can finally see in Figure 3.33 the interesting points selected, as well as their projection into the Front View image.



(a) (b) (c) (d) (e)

Figure 3.31: A sequence recording the main steps of the described algorithm, including the fall-back procedure for temporary enlarging the window. From the left: a first centroid is detected inside the initial window (3.31a); the window is shifted forward and a new centroid is detected (3.31b); no features are present where the window is shifted (3.31c); the fall-back procedure is activated and the window is enlarged (in yellow) until enough feature points are found (3.31d); the algorithm terminates when it reaches either the top or the lateral edges of the BEV image (3.31e). Notice how the re-centering of the window at each step makes the algorithm flexible enough to follow the line in spite of the imperfections in the image rectification.

We must point out that the algorithm is run independently on each marking line (e.g. left and right), starting from $\mathbf{x_l}$ and $\mathbf{x_r}$ respectively. Indeed, this independence

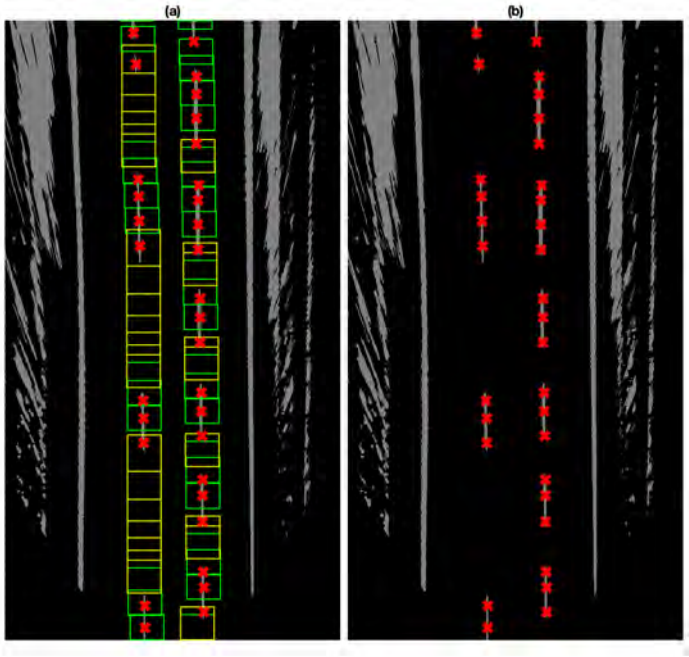


Figure 3.32: A complete picture of the results obtained with our algorithm on a straight highway road and the resulting interesting points selected. Notice how the algorithm performs well despite the large gaps in the broken line markings.

makes its computation extremely easy to parallelize.

Furthermore, this basic implementation can be extended to operate in more complex scenarios. Among the most interesting improvements is, for instance, the detection not only of the ego-lane, but also of other additional lanes on the road. If we think at the procedure just described indeed, due to the independence between different lines, initializing the algorithm in any point at the bottom end of the image would in principle allow it to perform its computation and find a candidate set of interesting points. Of course however, not all such initializations would lead to a correct detection. Nevertheless, as we did for the ego-lane, we can estimate the initial position of the unknown line markings given the calibration information of the camera. Therefore, we can not only initialize the algorithm on $\mathbf{x_l}$ and $\mathbf{x_r}$, but we can

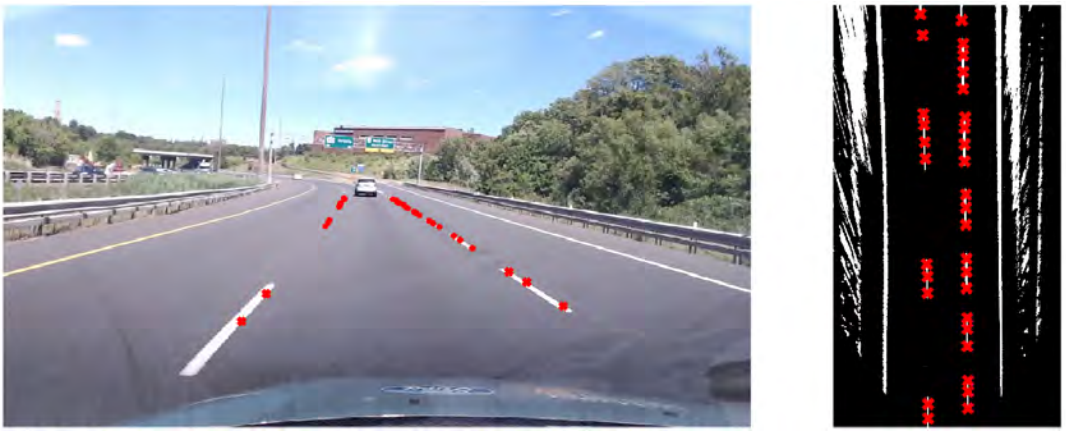


Figure 3.33: A projection on the front view image of the retrieved points seen also in Figure 3.32. Notice how well they adhere to the line markings of the ego-lane.

instead generate a whole set of points $\{\mathbf{x_1}, \dots \mathbf{x_n}\}$, each initialized where we expect to find a lane markings. If these estimates are precise enough and the window sizes are not too strict, each of them will detect a correct line point at the first step and then, thanks to the re-centering, end up with the correct computation. This is exactly what we do in Figure 3.34, where a multi-lane scenario is presented.

An additional development would work instead around the disruption of the independence condition. As mentioned before, treating each line separately achieves faster processing time, but doesn't allow us to capture the complexity of harder scenarios. For example, if no intersections are present, lane markings can be considered parallel to each other. Enforcing this information as a constraint on the feature selection process would severely restrict the search space, thus improving the ability of our algorithm to converge to the best solution. Several ways of inserting this idea in the algorithm are available. One possibility could be to run it on all the lines contemporary, shifting the windows forward in a synchronized fashion and enforcing some local constraint on the distance between interesting points selected on adjacent lines. This

would maintain the running time comparable and enforce a strict constraint on the lines, although it isn't always easy to implement when the lines are broken. Another alternative would be instead to maintain the same procedure described above, but reiterate it several times with different parameters and select, only at the end, the solution more supporting the required constraints. This allows for more flexibility on the formulation of the constraints at the cost however of an increased computational cost. Notice nonetheless that, before proceeding, we must ascertain that no intersection or lane merges and splits are present on the road, and this constitutes a research problem on its own.

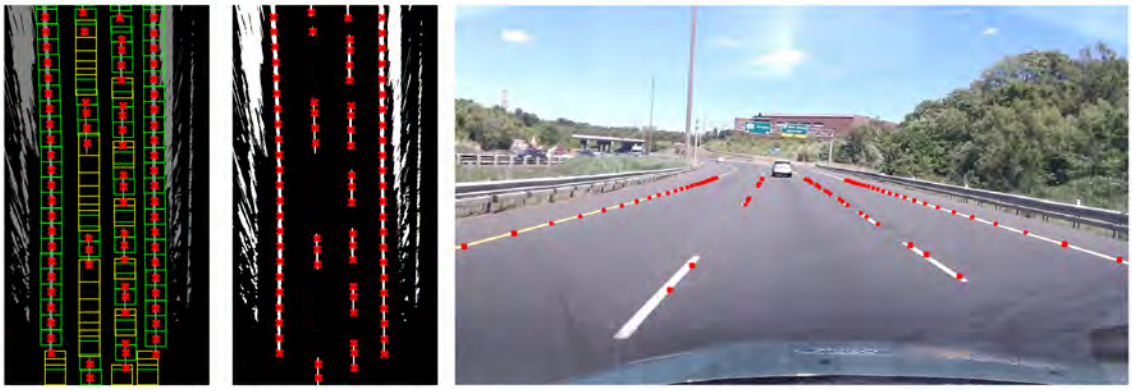


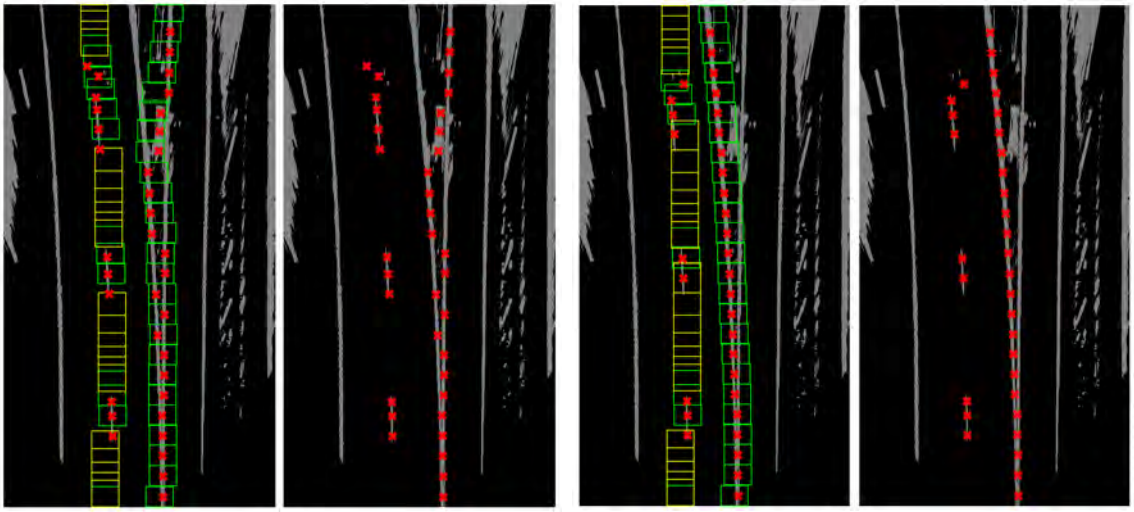
Figure 3.34: Extension of the described algorithm to the multi-lane scenario. The algorithm is run started from multiple initial points, carefully estimated in their world coordinates according to the type of road the vehicle is found in. For this instance, the initial windows are centered respectively at 6 m and 1.5 m to the right of the ego-vehicle and 1.5 m and 5 m to its left. We point out how the sequence of windows for each line is completely independent from the others.

In conclusion of this paragraph we must mention two improvements to the algorithms we had to introduce to improve its performances in particular reoccurring situations.

A first problem concerns the re-centering of the neighborhood window. We noticed

indeed that, for the presence of noise, resetting the center of the window with respect to the last centroid led to very oscillating results. Thus, to introduce a smoothing factor, we modified the algorithm imposing that the new window center would be computed considering an average of the previous k centroids detected. In our experiments, this new parameter was empirically fixed to $k = 3$. More complex filtering techniques could of course be introduced, but for this simple scenario it would have probably been unnecessary. Moreover, a moving average is more computationally efficient, property that can benefit a potential future deployment on a real-time environment.

In addition, a particular behavior was registered when a split in the road markings was encountered. In such situations, the centroid selection often faced the problem of deciding which line to follow, and the resulting output was not satisfactory (Figure 3.35a). In a fully autonomous scenario, the knowledge of a map and the output from the path planning and navigation system would be integrated and used to decide which line to follow. In lack of such information however, we forced our algorithm to remain focused on the lane currently occupied by our vehicle. To this end, we modified our centroid selection procedure to select, in case of a tie, the point closer to the ego-lane of our vehicle. Moreover, to avoid spurious detections, we added a threshold value to decide if a multiple selection was to be treated as such or was just due to the presence of noise. With this changes, the algorithm is capable to follow the inner lanes, including, most importantly, the ego-lane, without being misled by any lane split or merge (Figure 3.35b).



(a) (b)

Figure 3.35: The registered issue occurring with splits in the lanes is here analyzed. We can see how without proper measures the detections are completely useless in presence of the split, because of the alternation inside the window of features from both the generated lines (3.35a). With the trivial adjustment of giving priority to clusters closer to the ego-lane instead, the performances are substantially improved (3.35b).

3.4.4 Results and evaluation

Considering both the detection of the ego-lane and the extension of the problem to the adjacent lanes, in Figure 3.36 we can see some preliminary results obtained with our approach. We notice from the beginning that the algorithm detects very well the markings on straight roads, regardless of their type. This is particularly true for the ego-lane, which our method is usually able to accurately delineate. Some issues arise however when trying to detect also additional lanes, for reasons such as clutter in the feature image, at a certain distance from the vehicle.

Comparing these results with the output of the algorithm on road bends, we notice from Figure 3.37 that, as long as the curvature is not too sharp, the algorithm

(a) (b)

(c) (d)

Figure 3.36: Examples of correct detections on straight line markings in different road environment: a major (3.36a) and secondary (3.36b) highway, a rural road (3.36c) and an urban road (3.36d). We notice how it is impossible in some cases to retrieve all the lines in the scene because of the limitation of field of view used to compute the BEV image. We point out however that all the lines within the ROI are clearly detected without outliers, regardless of the conditions of the road or the type of line (solid or broken).

maintains its satisfactory performances. However, in exceptional cases, such as at a hairpin turn (Figure 3.38), its window-based mechanism looses the curved lines and fails. This cases however remain isolated to particular types of roads, and can therefore be predicted and tackled with other techniques. Moreover, enlarging the window in such circumstances, or introducing a level of tracking in the algorithm, such that it could exploit also its past computations, can certainly help preventing these situations. Indeed, if the algorithm is started long before the sharp bend, a progressively increasing curvature can be found in the previous frames, and their analysis can therefore be used to foresee this exception.

We proceed now to evaluate the presence of noise in the detected points, since it will be a crucial aspect for the success of the overall system in the subsequent

(a) (b)

Figure 3.37: The results produced by the algorithm on smooth road bends. The windows are large to follow the lines adequately but also small enough to avoid the introduction of outliers. A bit of noise is present however in the distance, but for an illumination effect completely uncorrelated to the road bend.

Figure 3.38: The poor performances of the algorithm in case of a sharp bend in the road, such as a hairpin turn. The windows are not large enough to detect a new candidate on the inner line of the curve and a completely wrong detection is returned. Results of this kind must be avoided or prevented also for the high risks that recognizing a turn as a straight lane might determine.

phases (actual model fitting). We notice that in most of the situation, when the detections are correct, no particular noise is present in the position of the points, and they adhere to the marking. The presence of noise is instead registered in the regions where the feature extraction has failed in the first place, such as when the illumination conditions in the region distant from the vehicle are adverse (Figure 3.39) or in presence of occlusive elements, such as another vehicle on the road. As expected indeed, occlusions severely disrupt the features enhanced at previous steps and therefore corrupt our detection (Figure 3.40).

Among our experiments, we noticed also a situation in which outliers were often generated: in presence of a lane split. Although the modifications described at the end of Section 3.4.3 do stabilize the algorithm in such occasions, when one of the branches is a broken line other, more complex, issues occur. Indeed, if within a single window we were to detect both the branches, then, as explained in precedence, the closest to the ego-vehicle would be selected. Instead however, if the window is positioned over an empty tract of the broken line, and nevertheless overlaps the farther branch, being this the only cluster available, its centroid is selected. This results firstly in an alternation of centroids taken from both the branches, and secondly in a possible deviations of the tracking to the wrong branch. Figure 3.41 clarifies the problem with an example.

Figure 3.39: Noisy detections and outliers are formed in presence of severe noise in the feature image itself. This is a sign that although the algorithm can improve many situations in which some impurities are present, it cannot cope with them where their level is too high. We notice however that this phenomena occur mostly on the far end of the lines, and that therefore could be avoided considering a shorter ROI.

As a last step of our analysis, we focus on the performance of the multi-lane detection, for which we have already reported some examples in Figure 3.34 and throughout this whole section. The main drawback of our algorithm is that it requires an initialization for the window of each line, and that its outcome is slightly sensitive to it. Indeed, although an imprecise initialization of some decimeters is not relevant,

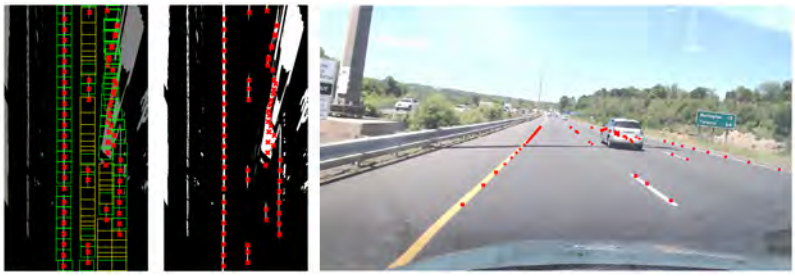


Figure 3.40: The presence of occlusion, represented by another vehicle on the road, determines numerous false positives and disrupts the detected shape of the underlying line in the feature image; our selection algorithm is not able to cope with this level of noise.

Figure 3.41: An unsolved issue in presence of lane splits delimited by broken lines is here depicted. Parts of the wrong branch are selected when the window overlaps only one of the branches at a time. As a result, outliers are produced despite the adjustment described in Section 3.4.3. We must point out also how in this example it is only for our luck that the tracking has continued on the correct line, but that in principle once this alternating behavior is triggered any of the two lines could be followed at last, producing potentially wrong results.

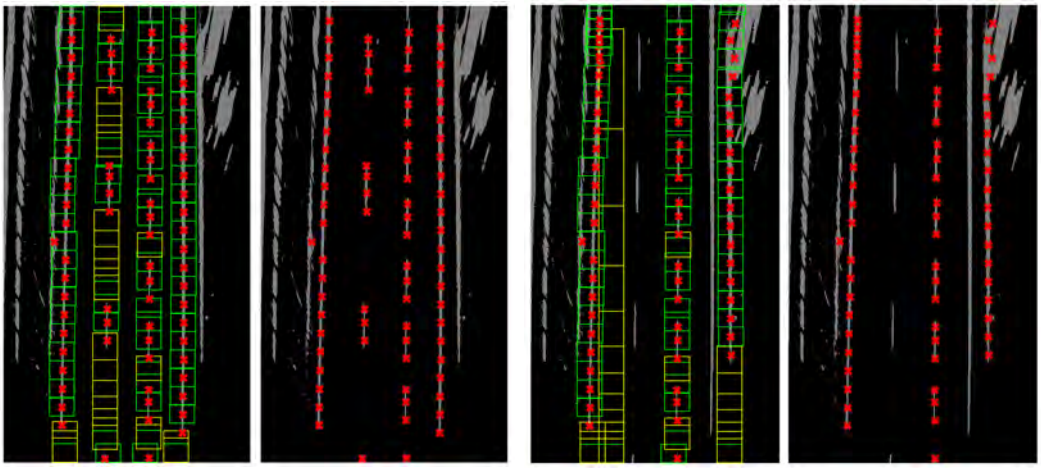
as the window will still be large enough to capture the line marking (Figure 3.42a), a deviation of 1 meter or more could severely invalidate our results (Figure 3.42b). This effect is amplified moreover by the presence of road bends or dashed markings, as the enlargement of the window in the forward direction might even lead it to a different line (as already shown in Figure 3.38).

There are however several options to fix this issue. On one hand, we are running

our algorithm without taking into account any temporal information. However, introducing some level of tracking, from simply storing a number of past detection to support the fitting, to the more articulated implementation of a complete Bayesian filter, it would be possible to construct a reliable estimate of where the lines will be in future frames. With such estimate, the initialization of the window, in terms of both position and size, becomes trivial.

Alternatively, the algorithm could be kept as it is, but initialized in many different points, each independently from the others. As a result, several sets of points corresponding to potential lines are detected and stored. After this first phase, we can analyze each set, enforcing some additional constraints, such as the mutual parallelism among all the candidate lines, or their non-intersection. As in a consensus technique then, the data most likely to represent the actual scene can be selected. In this way, no need for a precise initialization of each window is needed, and moreover no knowledge on the number of lines present is required a priori, at the price of a potentially expensive computation.

In conclusion, a consideration on the computational complexity of the algorithm must be added. Although all the operations are polynomial and easy to execute, the dynamic search of an adequate window size introduces some variability in the maximum computational time needed by the algorithm. This does not constitute a problem in our setting, as we are only conducting an experimental study. Nevertheless, it must be taken particularly into consideration once we decide to deploy the algorithm on a real-time system, as it is very important to maintain the computational time constrained in such environments.



(a) (b)

Figure 3.42: We show here the impact of an inexact initialization of the windows on the performances of the algorithm. In 3.42a, a scene is taken and the initial points for the algorithm are selected with a displacement of different decimeters with respect to the correct positions of the markings. Since the windows are large enough, the algorithm is able to hook the respective lines and follow them until the end of the image. However, if the displacement becomes more substantial, as 1 m, then issues arise, as depicted by 3.42b: a marking can be completely missed, and a line external to the road could be mistaken for an actual marking.

3.4.5 Conclusions

After introducing the problem of retrieving a limited and accurate set of points, in order to proceed with a reliable model fitting phase, in this section we described and evaluated a specific technique to perform this task.

Working on the BEV-space with the features previously selected, we have shown how it is possible to restrict our attention to a set of accurately chosen points by following the lines from our vehicle towards the horizon and employing a boxing technique based on the local continuity of such lines. With a simple adaptation, the algorithm is able to cope also with dashed lines, and with some additional adjustment its performance are significantly improved with respect to noise and outliers.

Furthermore, the same algorithm can be applied also in multi-lane scenarios.

However, some issues are present and must be considered for the planning of the subsequent steps. First, although the algorithm performs well on both straight and curve roads, sharp bends as a hairpin turn could not be detected, because the procedure is based on a fixed-size window. Second, it is not robust enough when severe clutter is present in the features it is based on, due for example to challenging illumination conditions or occlusion in the scene. Finally, some defects are related to the way the algorithm is initialized and the information it requires in this stage.

Moreover, for each of the issues discussed we presented at least one possible implementable solutions that should severely reduce or solve the related problem. For road bends and feature clutter, a model based on previous detection, through full tracking or just juxtaposition of previously found lines, can assure satisfactory performance in the short distance and acceptable performance elsewhere. The initialization problems instead could be solved by adopting multiple starting points and successively selecting the most likely results.

Finally, with all this in mind, we consider the algorithm presented and its results very promising, for its surprising simplicity and at the same time its notable performances. Moreover, we acknowledge the presence of the demonstrated issues but, with them, we registered also their potential solutions, judging them at the end of the day as a reasonable price to pay for its qualities and, at the same time, large potential for improvements.

3.5 Fitting line markings using preselected features

3.5.1 Introduction

In Section 3.4 we introduced the concept of model fitting and observed how the presence of noise and outliers in the features adopted could severely compromise the final outcome. We then explored a way of carefully selecting feature points to facilitate the subsequent tasks, exploiting the continuity of the line markings and adjusting the underlying algorithm to cope with broken lines and multi-lane scenarios.

In this section we will study different models to fit the selected points and the consequences they entail in terms of computational demand and overall performance.

3.5.2 Background and Previous work

As presented in details in Section 3.4, the model fitting problem can be reduced to the optimization of some parameters $\mathbf{m}$ with respect to the features $\{\mathbf{x_i}\}_i$, as determined by the function $f = f(\mathbf{m}, \mathbf{x_i})$. Analyzing this function, we can measure how a model fits some data by computing:

$$\|\mathbf{f}(\mathbf{m}, \mathbf{X})\|_p = \| \left(\; f(\mathbf{m}, \mathbf{x_1}) \; \ldots \; f(\mathbf{m}, \mathbf{x_n}) \; \right)^T \|_p \tag{3.15}$$

The best fit would lead to $\|\mathbf{f}(\mathbf{m}, \mathbf{X})\|_p \approx 0$ and would best represent the distribution from which our features have actually been sampled. Finding such fit is exactly the purpose of this section.

First step of every model fitting procedure is the definition of a function f, that represents the family of models to employ. A vast variety of models is available and

has been described in the literature for fitting road marking lines. The majority of the reviewed systems revolves around *polynomial lines* (linear [74], quadratic [34, 75], linear-quadratic [76, 16], cubic [77]) and *splines* (Cubic [33], Cubic Hermite [34], Catmull-Rom [10], B-Snake [12]); some more convoluted classes however exist, such as clothoids ([78, 79]), often used in road design ([80]), and non-parametric models ([29]).

Once the model family has been chosen, we need to define a score measure to evaluate the coherence of each model instance with the observed feature points. As explained above, this is equivalent to selecting the norm to be applied to f, following Equation 3.15. At times, this choice is constrained by the form of f; in general however several options are available. The most known method is certainly least squares (LS) [81] and its variations, such as total least squares (TLS), for their simplicity as well as their robustness to random noise. They are very sensitive however to outliers.

To cope with this issue often RANSAC [34, 82, 83, 32, 33, 84, 85, 44] is introduced, in combination with any other type of scoring mechanism. Notice however that its introduction in the pipeline impacts significantly on the time complexity of the algorithm. It is also for this reason that a good feature selection algorithm is crucial for the model fitting stage, especially in applications with the potential to be deployed on real-time environments. To conclude this brief panoramic, we point out that at times also custom LS-based scores could be designed in order to impose particular constraints on the results. Although adopting such variants requires some effort, in both their implementation and execution, in certain situations they allow to significantly restrict the search space, thus leading to faster global convergence and better output quality.

Moving back again to the fitting problem, our third and last step is the actual model fitting. As mentioned, this is nothing but a multi-variable—free or constrained—optimization of the chosen f with respect to its parameters $\mathbf{m}$ and according to the scoring method selected. As we can imagine, different model types and scoring functions entail different optimization algorithms. Most of the times however general purpose techniques could be employed.

3.5.3 Methodology

For our experiments, we take as input a set of interesting points P_k believed to belong to the same line k. Formally, we define $P_k = \{\mathbf{p_{k,i}} = \left(x_{\mathbf{p_{k,i}}}, y_{\mathbf{p_{k,i}}}\right)\}_{\mathbf{i}}$. These points will be here considered as being retrieved using the algorithm described in Section 3.4, but any other comparable selection technique could in principle be adopted. For further clarity, an example of such points is provided in Figure 3.43.

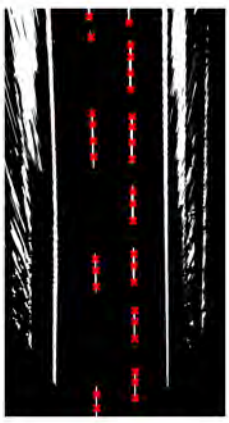


Figure 3.43: Example of the feature points used to find a fitting model.

Following the steps outlined in Section 3.5.2, our first goal is to determine a model family f. We decide to analyze both polynomial models and some splines, reporting

below a descriptions of their features and defects. For what concerns the optimization score, we apply to all of the analyzed models an ordinary least square methodology. We consider the abscissae to be vertical in the pictures.

Polynomial models

A generic polynomial line of degree α in coordinates $(x, y) \in \mathbb{R}^2$ is parametrized as:

$$P(x, y) = \sum_{i+j=\alpha} a_{i,j} x^i y^j \tag{3.16}$$

For road markings, usually $\alpha = 1, 2, 3$ are used:

$\alpha = 1$**.** Linear models are often employed when only the portion of the road closest to the ego-vehicle is taken into consideration. Indeed, for very short distances, any line marking can be approximated with a straight line obtaining satisfactory results.

$\alpha = 2$**.** Quadratic models aim at describing the road lines with parabolic or hyperbolic curves. This is the simplest choice to model also bends in the road, although it often does not provide enough expressive power to describe them precisely.

$\alpha = 3$**.** Cubic models bring with them a potentially large expressive power, but if the number of feature points is not elevated, they risk to dramatically overfit the data, and this is particularly true in presence of some outliers. Indeed, while a quadratic model can at most fit a single road bend, a cubic function is able to model a wide range of lines, even for example strongly S-shaped curves, and the optimization procedure could thus end up selecting even completely implausible models in presence of strong outliers.

When fitting a polynomial curve, we are assuming that our feature points are completely free from the presence of outliers. This is plausible in our circumstances because we perform the model fitting only after a feature selection step. Also for this reason we employ just a simple least square optimization instead of exploring more sophisticated (and time-consuming) ones.

Figure 3.44 shows a comparison of different polynomial fitting applied to the same image.

Spline models

A spline S is a piecewise polynomial function defined to fit some data points while respecting some characteristic constraints. Formally, we divide the domain $[a, b] \in \mathbb{R}$ in k intervals delimited by $x_0, x_1, \ldots, x_k$ (break points). The intervals need not to be equally sized. Each of the points x_i is associated with the respective function value y_i, to be approximated by our spline line S. Then, a different polynomial function of a fixed order P_i is associated to each interval, and each polynomial is connected to the adjacent ones by means of some some additional constraints on the continuity and differentiability of the overall function. Formally, we have:

$$\begin{aligned} S(x) &= P_0(x) \quad \text{for } x \in [x_0, x_1) \\ S(x) &= P_1(x) \quad \text{for } x \in [x_1, x_2) \\ &\ldots \\ S(x) &= P_{k-1}(x) \quad \text{for } x \in [x_{k-1}, x_k] \end{aligned} \tag{3.17}$$

such that

$$P_i(x_{i+1}) = P_{i+1}(x_{i+1}) \quad \forall i \in \{0, \dots, k-2\} \qquad \left(S \in C^0\right)$$
$$P'_i(x_{i+1}) = P'_{i+1}(x_{i+1}) \quad \forall i \in \{0, \dots, k-2\} \qquad \left(S \in C^1\right)$$

These are the standard constraints usually adopted. Additional constraints are then needed to fully characterize the spline curve, and it is exactly on the nature of such constraints that different families of splines are generated. (For more details on spline curves we reference [86].)

In the remainder we will focus only on not-a-knot cubic splines. These splines are additionally twice continuously differentiable and obey to the not-a-knot boundary condition, i.e.:

$$\begin{aligned} P''_i(x_{i+1}) &= P''_{i+1}(x_{i+1}) \quad \forall i \in \{0, \dots, k-2\} \qquad \left(S \in C^2\right) \\ P'''_0(x_1) &= P'''_1(x_1) \qquad \text{(not-a-knot boundary condition)} \\ P'''_{k-2}(x_{k-1}) &= P'''_{k-1}(x_{k-1}) \quad \text{(not-a-knot boundary condition)} \end{aligned} \tag{3.18}$$

For more information on the process of approximating these functions by least squares we refer the reader to [87].

Some examples of this model are shown in Figure 3.45, applied for comparison to the same image represented in Figure 3.44.

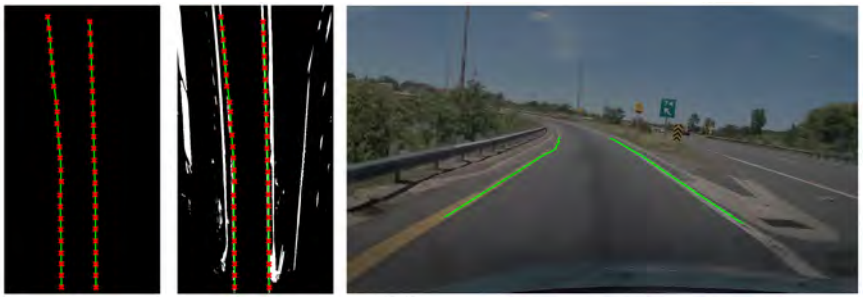
(a) $\alpha = 1$

(b) $\alpha = 2$

(c) $\alpha = 3$

Figure 3.44: Example of a polynomial line fitting applied to the image of a bend in a highway scenario. We point out that, being in a highway, the bend is not too sharp, although some curvature is present. Moreover, notice that no outliers are present in the picture, as assumed before, and that very little noise can be seen. With this in mind, we can see that in this scenario a linear approximation is not accurate enough to describe the road markings, although it still manages to give us a sense of the direction of the road. For quadratic lines instead, we achieve very good performance, highlighting the road curvature perfectly. Finally, we can see how adding more expressive power, with the cubic lines, does not result in any increase of performance in this particular instance, being the road structure rather simple (a single road bend).

(a) Least squares with 5 break points



(b) Least squares with 10 break points

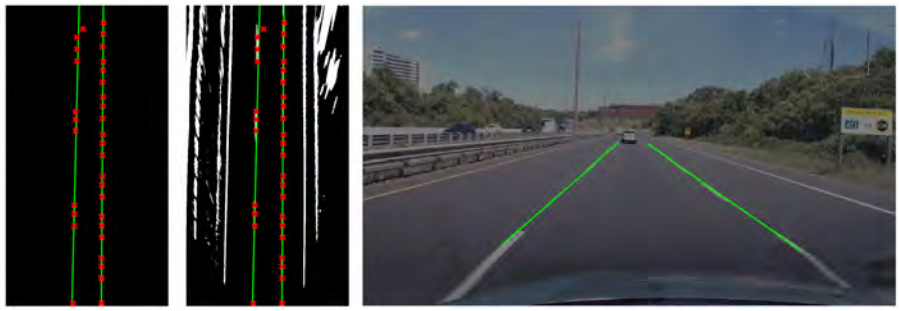
(c) Pure interpolation

Figure 3.45: Not-a-knot cubic spline fitting is here added to the scenario presented in Figure 3.44. In particular, we fit our models with 5 break points for 3.45a, 10 break points for 3.45b and all the data points as break points (pure interpolation) in 3.45c. We can see that although the main shape of the markings is captured, the pure interpolation clearly fails due to the presence of noise (even if in a small amount). Using few break points instead (and thus forcing more approximation), the curve is able follow the feature points smoothly. It does not perform however greatly in the distance, where even a small amount of noise is significantly amplified by the perspective projection, and the lack of a strong constraint on the shape of the line makes the output of our algorithm potentially less reliable.

3.5.4 Results and evaluation

We proceed to evaluate the four types of models fitting analyzed, i.e. linear, quadratic and cubic polynomial lines, and cubic splines.

We already know that the model complexity of linear lines can't match with the shape of a general road marking, due to its inability to model curves in the road, from the simple highway turns to the more complex urban scenarios. According to our study, a linear polynomial can still be used however as a first approximation in two situations: where dealing with straight roads, such as in most highways, and when fitting only lines in the very short proximity of the vehicle. Figure 3.46 proves with an example these two cases.



(a) Straight roads

(b) Short distance

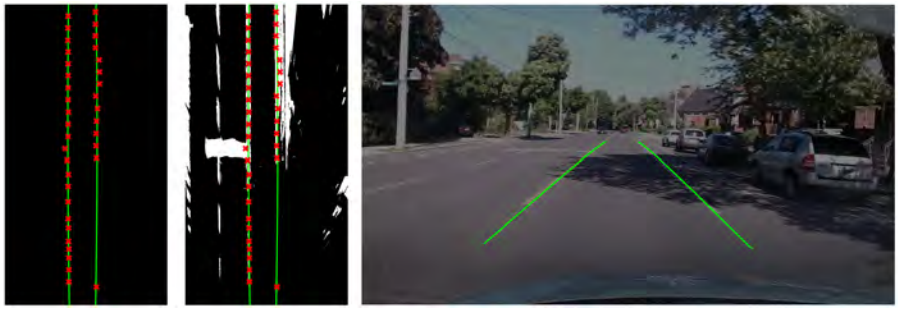
Figure 3.46: Two examples of situations where a linear model can be employed successfully as a first approximation: when the road is known to be straight (3.46a) or when only markings in the short distance are considered (3.46b).

Quadratic lines are the model that best generalizes the description of the road markings in the tested scenarios. Indeed, it has the strong ability to well describe curves (Figure 3.47), while at the same time its bias for representing only one single bend in the road enhances its robustness to noise. This is clear from Figure 3.48, where both quadratic and cubic lines are compared on an image damaged by the presence of a few noisy detections. We must notice however that this same lack of expressive power is at time too strict and doesn't allow it to capture more complex situations, where a cubic model could in principle perform better.

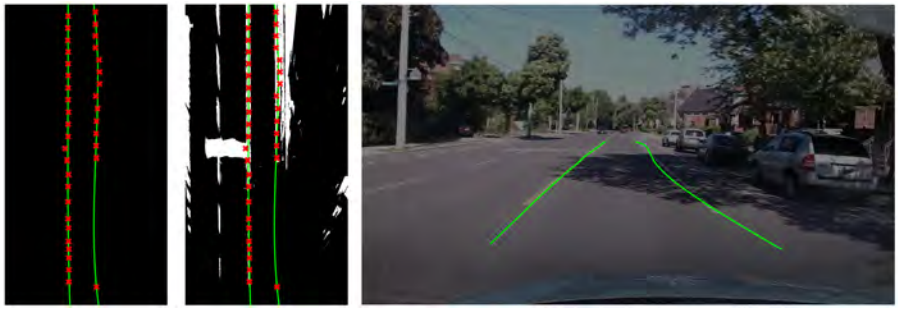
Figure 3.47: Example of the satisfactory performances of a quadratic model in presence of a road bend.

We must point out that all these polynomial methods are, however, unable to cope well when some outliers escape the feature selection process, as seen for example Figure 3.49. This is a limitation imposed by the fact of using a single model for the whole length of the road markings, made more severe by our choice of employing only least squares techniques.

Finally, we notice instead that cubic splines present a degree of flexibility we can use to our advantage, as in the situation depicted in Figure 3.50. However, the selection of the number of break points (and thus the degree of approximation desired) is not simple and finding a fixed trade-off severely reduces the potential of



(a) Quadratic model



(b) Cubic model

Figure 3.48: Comparison of the robustness to noise of quadratic and cubic models. It is clear how the larger expressive power of cubic models is, in this case, overfitting the data.

this method. Indeed, while we showed in Figure 3.45 how a limited number of break points (5 in the picture) would benefit the generalization of the algorithm, it is clear from Figure 3.50 that this is not always the case. When indeed some outliers are localized in a distinct region of the image, a larger number of break points seems to allow the algorithm to maintain good performances in the remaining part of the road, isolating the portion with issues. In this way, it would be easier to post-process the corrupted section, for example by means of tracking. However, although this difficulty, we must point out that this property still represents an important feature, as it is not seen in any of the polynomial line studied before, where any outlier always contributes to the fitting in an equal measure.

Figure 3.49: The behavior of polynomial lines in presence of outliers (false detections on the right side of the road): the model is not able to distinguish in- and out-lier points and fits them all.

3.5.5 Conclusions

In this report, we introduced the problem of model fitting and some of the model families used in the literature. We then implemented polynomial and spline models to gain understanding into their performance in different situations. We have not mentioned it explicitly, but a strong consideration emerges clearly from the results obtained: the set of possible scenarios to be treated when dealing with line detection and its model fitting stage is extremely large, and this is indeed the main limit our research has to face. As a result, it is very hard to find a well-balanced compromise between one or another family, although every model tried, with a specific set of parameters, is able to perform satisfactorily under specific scene conditions.

(a) Spline with 5 break points

(b) Spline with 10 break points

(c) Quadratic line

Figure 3.50: Three models are applied in a situation with some outliers in the distance; approximately the first 30 meters present instead perfect feature points detection. We see how a cubic spline with a low number of break points (3.50a) is not able to separate the well-detected area from the noisy one, and the resulting fit is impacted by this lack of flexibility. Raising instead the number of break points (3.50b), more freedom is allowed and it results indeed in a perfect fit of the well-detected area. Finally, we show for comparison the performances of a quadratic line in the same situation (3.50c), in demonstration of how the fitting function considers every points equally when finding the perfect match.

Chapter 4

Overall system

Given the experiments performed and described in Chapter 3, several options have been analyzed on how a lane detection system could be constructed. After such analysis, we face here the problem of actually selecting which components will constitute our final system. Our description will follow step by step the general pipeline, introduced in Chapter 2, Figure 2.1, and will be finally summarized by Figure 4.1.

The *acquisition* process is performed by means of a single camera, opportunely mounted on a vehicle and facing forward. Along with the images thus acquired, we assume to possess also the full calibration information of the camera, with respect to a world reference system with origin at the center of the front side of our vehicle, at ground level.

The *preprocessing* transformations applied to these input images, processed frame by frame, are mainly two. First, the image is projected into the Bird's Eye View plane, where the subsequent stages will act with more ease. Then, a reduction of the Region Of Interest is carried out, in order to decrease the large amount of information to be processed and at the same time enhance the section of the image of true interest

for us: the road.

Subsequently, the *feature extraction* is performed. We adopt the custom color-based technique designed to retrieve white and yellow line markings, as described in Section 3.3. Notice that, although the methodology applied is in principle interchangeable for the purpose of the next stages, this particular choice provides a solid support for them, which are indeed fine-tuned around it.

Finally then, it's the time of the *model fitting* stage. Even though we described it in precedence as a single atomic step, we realized that, to achieve satisfactory performances, this phase has to be in turn split in two parts. First, our experiments enhanced the clear need for what we call a *feature selection* step, where only few feature points are meticulously chosen among the vast collection previously extracted. These small set of points helps to significantly reduce the computation needed in the traditional fitting step, while also purifies the features from potential outliers. In our final system, we adopt the window-based algorithm proposed in Section 3.4. As a final remark notice that, since this algorithm has to work in strict connection with the previous *feature extraction* phase, the two are fairly coupled. However, any other feature extraction methodology with the same input-output relation and a similar feature distribution could in principle be used before it.

After selecting the most important features, a more classic *line fitting* is applied. This optimization is performed in the Bird's Eye View plane and, for our system, adopts a least square method considering the abscissae to be directed upwards in the image. In particular, a quadratic polynomial model is selected for this task. We noticed indeed that it is very hard to chose only a single model, because of the strong variability of environmental conditions our system has to face. However,

quadratic curves appeared to be the best compromise, providing good performances in the majority of the examined cases and displaying a certain degree of robustness in adverse scenarios.

In the light of the above, for each acquired frame the *output* of the overall system is constituted of three components: the parametric function describing the road markings in the BEV image, the BEV and front-view images acquired and, finally, the calibration information available at the beginning. By means of these components, it is then possible to project the detected lines into the front-view image and, finally, into the world reference frame, retrieving the estimated position of each lane on the road, ultimate objective of our work.

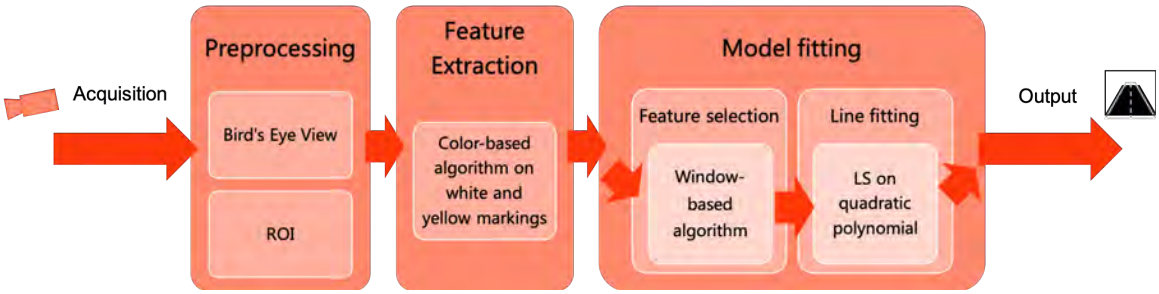


Figure 4.1: Pipeline of the overall final system.

4.1 Technical setup and implementation

The practical nature of our research lead us to make use of several technical components throughout our study, both on a hardware and software level. While some of them have been acquired, others, when possible, were realized in our facilities. Regardless however of their nature and origin, they are presented in the following.

4.1.1 Hardware

We acquired the data used for all of our experiments and results evaluation with our instrumentation, and in particular using our Logitech HD Pro C920 camera mounted on a Ford CMAX SEL-Energi 2013.

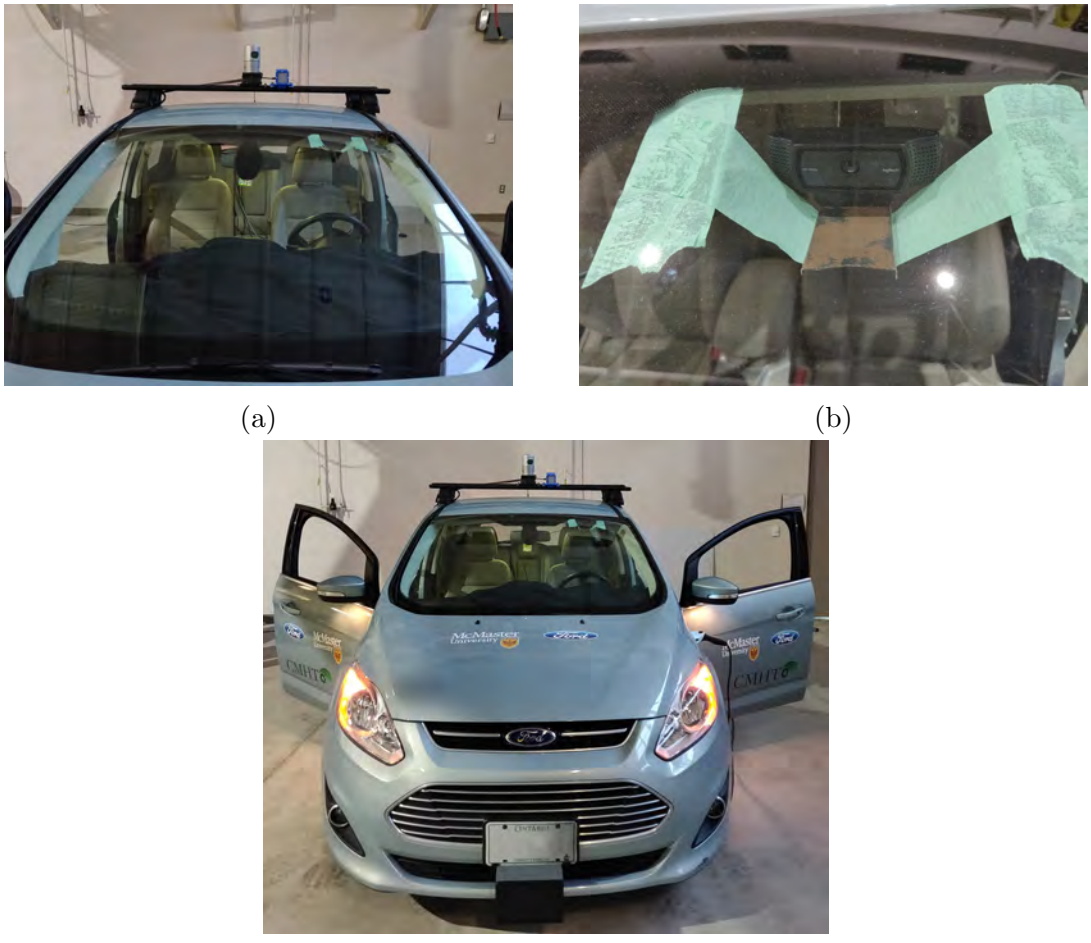
The camera supports a maximum resolution of 1080p at 30 fps or 720p at 60 fps, and provides several low-level hardware features. Moreover, its nominal focal length of 3.67 mm is low enough to provide a wide field of view (nominally 78°). Table 4.1 reports further details on its technical specifications.

Logitech® HD Pro C920	
Focal Length	3.67 mm
Diagonal Field of View (FOV)	78°
Frame Rate (max)	720p@60fps 1080p@30fps
Optics	Carl Zeiss® lens with 20-step autofocus
Video formats	Uncompressed, H.264
Others	- Automatic low-light correction ($^1/_2$ resolution) - 1/4-20 tripod mounting

Table 4.1: Technical specifications for our camera

The camera was mounted inside of our vehicle, close to the top end of the windshield. This position confers to it full protection from adverse weather conditions

and at the same time provides a discrete height for capturing the road underneath. A custom mounting plate has been prototyped in-house to secure the camera in the right position, avoiding oscillations that could introduce severe noisy conditions in the recordings.



(a) (b)

(c)

Figure 4.2: The camera mounted behind the windshield of our Ford CMAX. Other sensors, not used in our work, can also be seen.

Finally, the computational power needed was obtained from a MacBook Pro (13-inch, 2016), with a 2 GHz Intel Core i5 CPU and a 16 GB 1867 MHz LPDDR3 RAM.

This choice is clearly feasible for our work because of its experimental nature, but different considerations have to be made when deploying the system on a real-time environment. In particular, the operating system on such laptop does not provide support for real-time applications, and more specific options, as a real-time embedded platform, would be required.

4.1.2 Software

For the execution of our experiments and the final development of our system, our data acquisition procedure had to be detached from the actual processing pipeline. This choice enabled us to proceed with the acquisition before the overall system was constructed, and to save parts of the recordings for a purely testing phase.

Additionally, moving forward with our development, the capabilities required by each of the two parts slowly diverged, and with them also their implementation details. The acquisition indeed was, on one hand, expected to be fast enough to capture frames at the required frequency and with no data loss. The processing pipeline, on the other hand, required very fast prototyping, while not necessarily efficiency. As a consequence, two different environments were adopted. Within the second moreover, a specific experimental framework was devised to facilitate common research operations.

Data acquisition

For the acquisition and storage of the data, a near real-time application had to be developed. This software interfaces with the camera, connected via USB, and contemporary acquires the data and saves them on the hard-drive. The parallelism is achieved through multi-threading, where a synchronized buffer is shared between this

two software components. The communication protocol for the acquisition of the images as well as their saving on disk is implemented with the use of the open source library OpenCV on C++.

Because of computational power limitations, the search for a trade-off between image resolution and sampling frequency was for us required. Our system is capable of sampling and saving the data in real-time at 15 fps at a resolution of 720p; to sustain a higher sampling frequency, such as 30 fps, with the same image resolution, the saving process would need to be optimized, possibly on the hardware side. If the algorithm is run in the current conditions, the saving loop would not be able to keep up with the sampling frequency, and a constant increase of the buffer size would be required to avoid data loss. Needless to say that this would further overload the system.

To conclude, we must mention that, for optimizing storage and time complexity, the data are compressed according to the standard H.264 and saved in file format M4V.

Processing

Once the data has been securely stored, the research of a processing algorithm is performed by implementing several experiments and analyzing their results. The main requirement for this phase is to obtain, in the shortest possible time, a working version of each candidate algorithm. This way, even if it is not optimized, we can rapidly proceed with its testing. We acknowledge of course that its lack of optimization is likely to limit its time performances. Nevertheless, their fast implementation strongly benefits the research character of our work. Furthermore, this evaluation still gives us

enough information to assess each algorithm, and if the experiments are run within the same platform, their performances can still at first be estimated on a relative scale, by direct comparison. Only in a second time, and if required, a precise analysis can be performed.

With this in mind, our development choice fell onto MATLAB, which provides a fast prototyping environment, enriched by an extensive library of utilities and domain specific toolboxes. MATLAB is also one of the first choices for developers in the automotive field, for its simplicity and deep coverage of algorithms designed for—and known in—such sector. For our purposes, the Image Processing Toolbox and the Computer Vision Toolbox were preciously employed for their implementations of several well known algorithms. A small contribution was given also by the Automated Driving System Toolbox and the Robotics System Toolbox, clearly closely related to our field of study.

Notice finally that, even under these choices, each of the algorithms implemented can always be optimized in the future, moving towards lower level techniques (as C++ with OpenCV) and being finally deployed on a real-time platform.

The experimental framework

In order to experiment but at the same time keep track of our achievements, a Git repository has been maintained. Its nature indeed would make it easy, at a later time, to retrieve past results and investigate, for instance, the nature of some code changes. This choice has been beneficial for two different reasons. On one hand, it satisfies several software engineering principles, helping maintaining the code and enabling its versioning. On the other, it contributes to the scientific research process itself, as

a scientist should always be able to trace back his own ideas and hypothesis when evaluating the outcomes of his experiences.

Moreover, we designed the internal structure of the repository to further facilitate the scientific work. We made different attempts to find a code structure sufficiently constrained to enforce a productive organization, but at the same time free enough to avoid limiting the research itself. At the end, the arrangement currently adopted has been considered best and has indeed given its results. Focusing on our MATLAB workspace, its root folder is divided, as shown in Figure 4.3, into:

`/experiments`. It contains in turn a folder for each experiment we ran, identified by a number and a short description. A special one, *exp000 - template*, represents a template for the creation of new experiments. It contains a previously set `config` file along with a simple `main`. Once instantiated, any custom function needed by the new experiments can be implemented and placed in its folder, if not already present in `/include` or `/libs`.

`/include`. It contains all the libraries and utility functions we designed and implemented for the project, to be shared among the experiments.

`/libs`. Here are instead all the external libraries we imported.

`/results`. With a sub-folder structure analogous to `/experiments`, here all the results are divided and collected. The code in *experiments/exp000 - template*, presented above, is already set up to create and store all the results into the correct experiment folder, and provides adequate APIs to perform this task.

This organization helps us maintaining each experiment separated, while at the same time leaves us the possibility of easily sharing routines among them. In addition,

the automatic creation of a result folder for each experiment helps us in the collection and, most importantly, in the storage of the results, ready to be analyzed at any time and protected from being mismatched or overwritten.

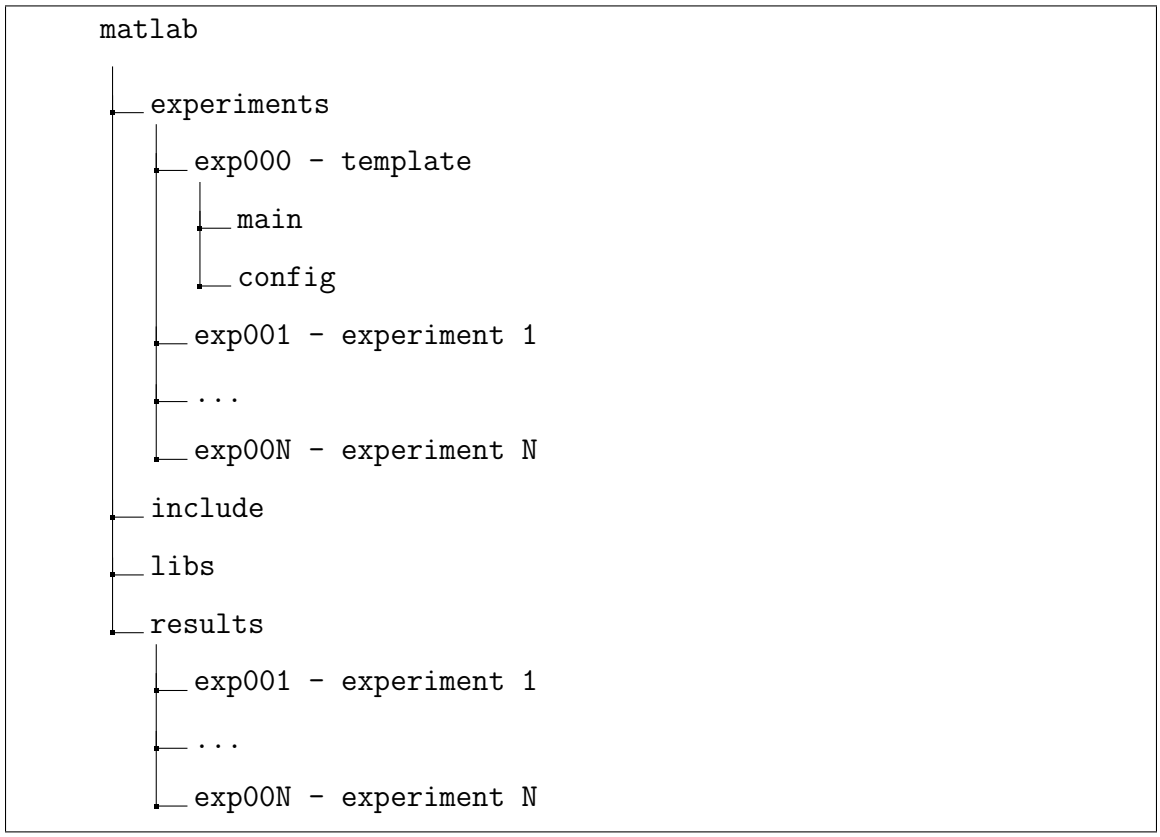


Figure 4.3: The folder structure of the MATLAB scientific framework we designed.

4.2 Results and evaluation

We conclude the description of our system with an overview of the results obtained and a conclusive evaluation.

As mentioned throughout the reports in Chapter 3, to develop and assess the performances of our work we acquired a substantial amount of data by driving our

equipped vehicle through the streets of Hamilton, ON (Canada) and the neighboring towns. We included in our dataset recordings of urban scenarios as well as data taken on a highway, an intercity road and some rural roads (Figure 4.4).

Figure 4.4: Sample frames extracted from the data we acquired, recorded in highways, intercity roads, rural roads and urban scenarios.

The acquisitions were performed in four different occasions, although the first two runs were carried out only for testing purposes and presented some imperfections in the camera mount and its acquisition channel. Thus, the data from the third and longest drive has been mainly used, after splitting it between a train and test set as

much homogeneously as possible. The fourth recording was instead kept for testing only and was acquired in an urban area never visited before.

Although the first test runs were performed partly during night-time, no available data for after dark-driving was reliable enough to be used. Instead, the main runs were conducted during sunlight and featured mostly clear or cloudy weather conditions.

Analyzing now the results obtained on the mentioned test set, we can draw some conclusions. The system results capable of detecting the main lane on the road (the lane where the vehicle is located) in favorable illumination and weather conditions (Figure 4.5). In particular, it is highly successful in highway scenarios (top figure in Figure 4.5). When however the shade of trees and overpasses cover the line markings, the illumination of the scene generates artifacts on the asphalt (such as bright spots or reflections) or occlusion prevents us to retrieve a complete information, the algorithm suffers some deficiencies (Figure 4.6). We must remember however how also the internal conditions of the system could influence this result. First, the acquisition process is performed with a web-cam, simple and effective sensor for an experimental environment, but obviously lacking some of the quality expensive professional systems have. Moreover, and most importantly, our pipeline works on a frame-by-frame basis, and no temporal tracking is currently implemented. Its introduction would indeed be able, probably, to improve significantly these lacks, exploiting a large slice of information (the temporal one) not used yet up to now.

On the pros side for our system is furthermore its ability to correctly detect also adjacent lanes, when present and visible, as demonstrated by Figure 4.7. This ability is of course not immune from the issues previously discussed, and it's best appreciated when the road markings are in clear environmental conditions. As a disadvantage,

no mechanism is presently implemented to determine or estimate beforehand if the additional markings are present and in what position it is more likely to find them. It is also for this reason that a future extension of the system to make use of the temporal information could be largely beneficial for many of its components.

Minor feature, yet noteworthy, is the built-in mechanism for recognizing, along with the position of a road marking, also its nature, that is to say whether it is solid or dashed line. Because of the important semantic connotation these two types of lines usually carry, this could be an important strong point for our methodology, as no mention of similar features was indeed present in the literature we reviewed.

Finally, we can notice how the computational performances of this system are rather limited, despite the noticeable simplicity of its building blocks. However, we know that it has purposefully been built within a scientific-oriented environment, which is MATLAB, where more emphasis is put on fast prototyping and efficient development of experiments rather then on the deployability of the outcome on an actual real-time platform. It is true indeed that the algorithm could technically be implemented using more universal programming languages, as C++, and possibly adopting non-proprietary frameworks, as OpenCV. And it is also true that several optimization techniques could be applied to adapt it for a real-time environment and maybe even for running on a vehicle. Nevertheless, this has never been our objective, which was instead to perform a scientific study, and thus we can serenely say that these points only represent valid future extension to the work here presented.

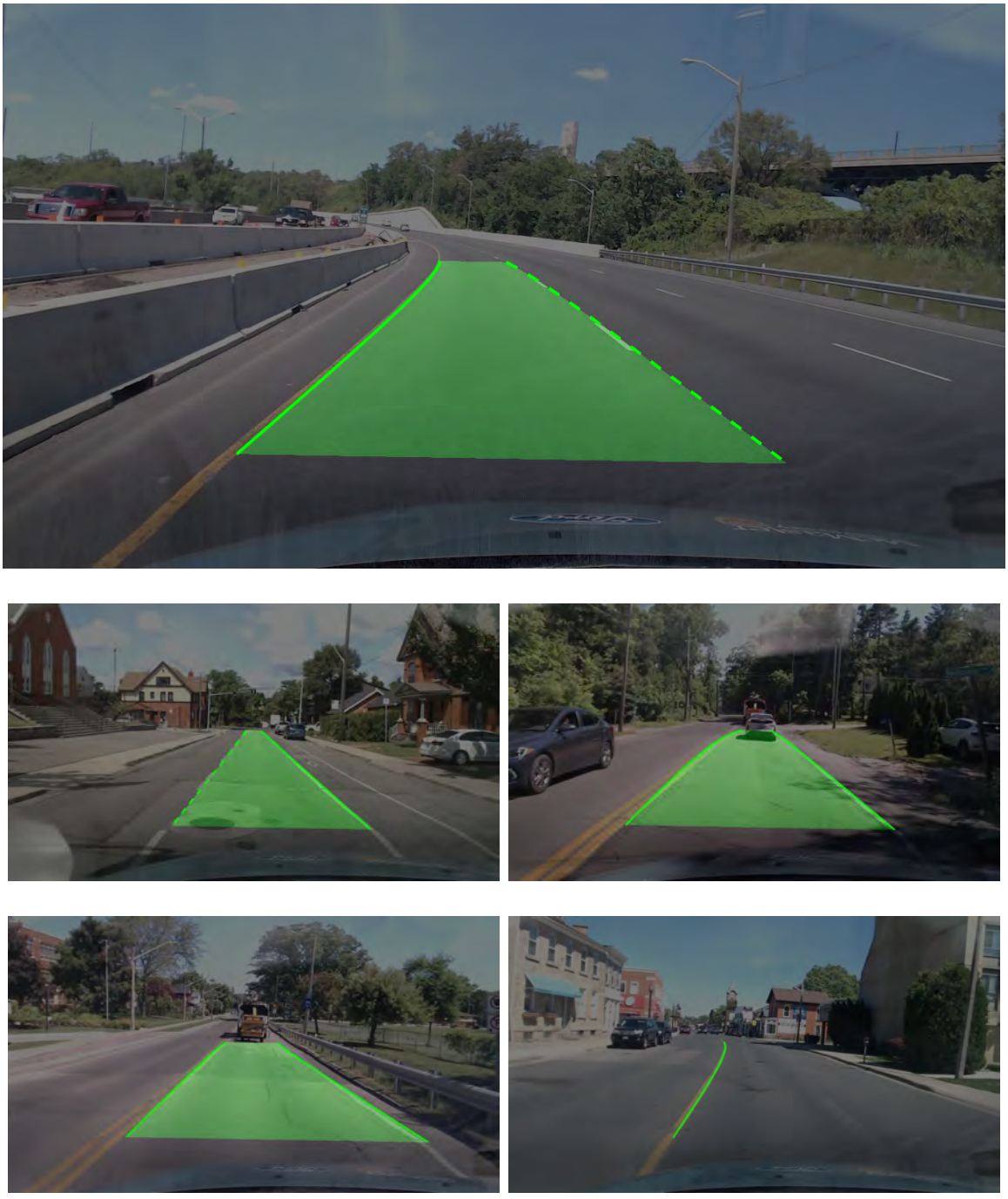
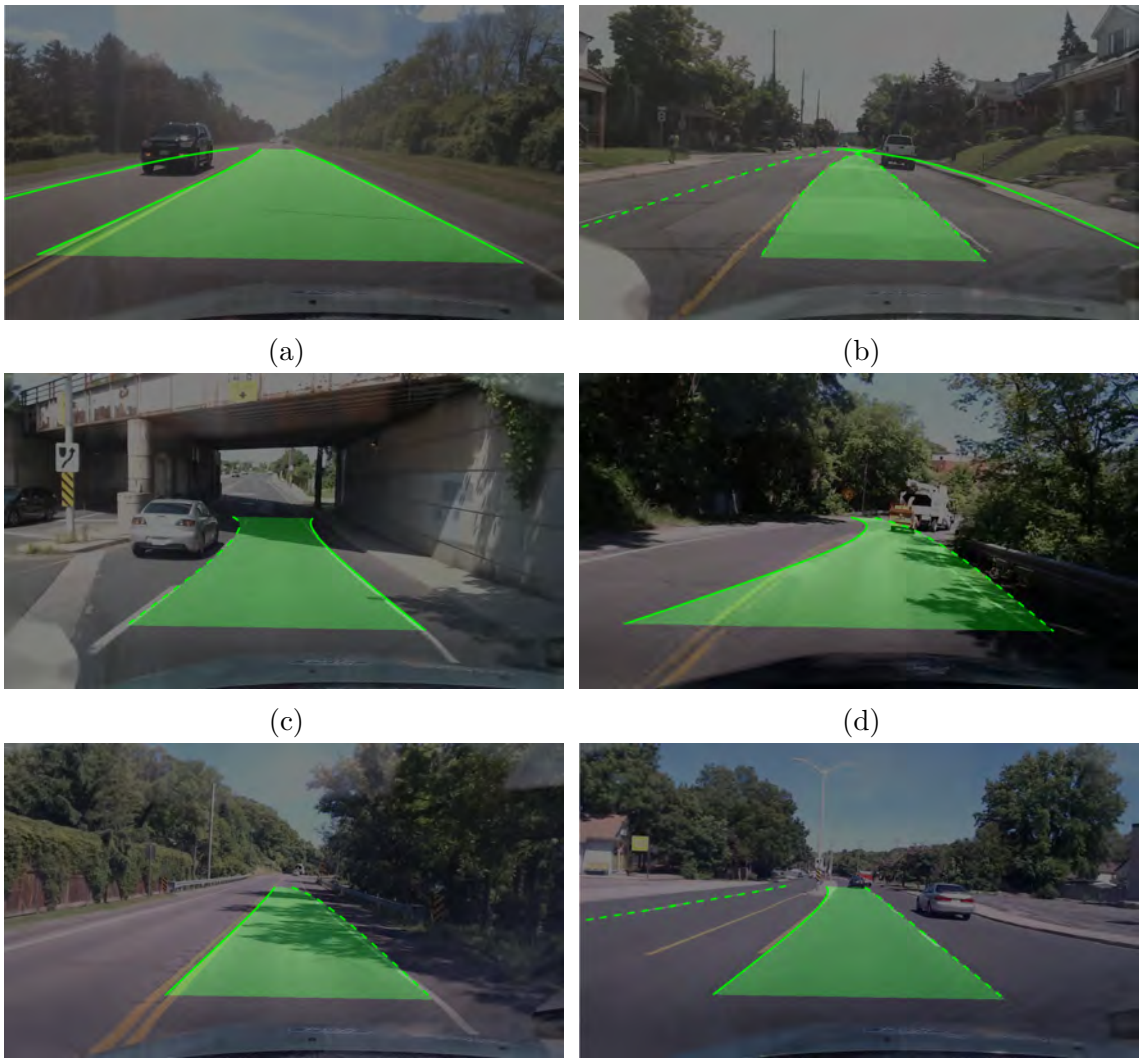


Figure 4.5: Correct detections of the main lane in several favorable environmental and road conditions.



(a) (b) (c) (d) (e) (f)

Figure 4.6: Examples of the failed detection observed from our experiments on the test dataset. From further analysis, we determined them to be caused mainly by occlusion from other vehicles (4.6a) and irregular illumination conditions, such as direct sunlight (4.6b) or presence of shades from overpasses (4.6c) and trees (4.6e). Secondly, also sharp bends (4.6d) and presence of occluding road elements (4.6f) have been registered.

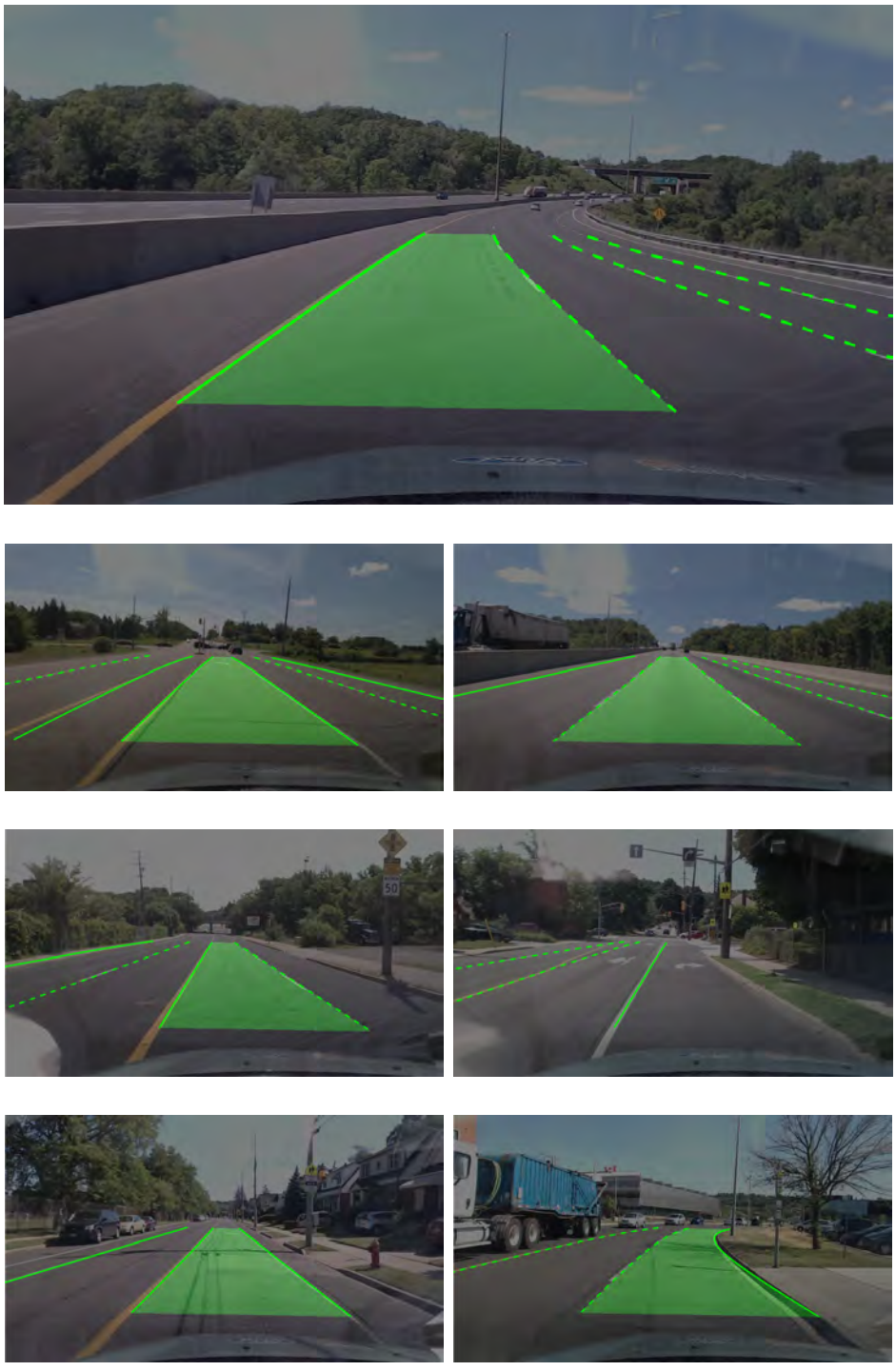


Figure 4.7: Correct multi-lane detections in several favorable environmental and road conditions.

Chapter 5

Conclusions and future work

As with many problems, there is a long path form theory to practice. What can be easy for a human, is often difficult to be done by a computer. Moreover, the sheer amount of different situations, environmental conditions and technical factors have to be considered.

We set out to better understand and present to the reader approaches and compromises related to lane detection systems with single camera. We analyzed the existing techniques and their operating conditions, in order to understand their weaknesses. By so doing, we obtained the necessary insights to setup the hardware tools and build the software framework needed to perform experiments on our own system. With it, we were finally able to further investigate this task and possibly, in the future, other connected issues.

We can consider our work as for both its theoretical and technical implications.

For what concerns the backgrounds and the most theoretical parts, we first identified the key techniques used in lane detection, as reported in Chapter 3. We studied typical preprocessing methods, feature extraction and selection techniques and finally

line fitting algorithms. While doing so, our aim was to collect all the needed background information in one single place, ready to be used as a good starting point by anybody growing interests in this area. It is also for this reason that we present each technique in a self-standing report, easy to be read individually and out of context. Notice moreover that, with this chapter organization, we indirectly categorized our literature review by topic, favoring also its rapid consultation in the future.

At last, through further experimental analysis of each different approach, we were finally able to outline their weaknesses. It is also to this end that we concretely implemented all the studied methods, as reported below.

Equally important, on the technical level we achieved instead the following objectives.

1. We mounted our camera on a car, calibrated it and recorded our own driving dataset (Chapters 4.1.1 and 4.1.2).

2. We implemented and studied different key techniques adopted in lane detection: histogram-based edge feature extraction and selection, vanishing point estimation, color-based feature extraction, window-based feature selection, line fitting using preselected features (Chapters 3 and 4.1.2).

3. We constructed a simple but effective software framework for organizing the resources shared across multiple experiments. Thanks to it, we can guarantee the reproducibility of each result and prevent their accidental loss. Moreover, the general character of such framework makes it easy to be adapted for any other situation where variations of an algorithm have to be run on the same data (Chapter 4.1.2).

4. We implemented an overall system assembling some of the pieces from point 2 (Chapter 4).

Even more importantly, thanks to our work in its entirety we deepened our understanding on how environmental conditions affect lane detection systems and, as the literature partly anticipated, we experienced the difficulties in fine-tuning our system to cope with each different scenario. We noticed that although many algorithms exist, and even though they all produce some results, each of them seems to be characterized by its own operating conditions. Some are best only on straight roads, while others perform better on bends. Some prefer sunny weather, while others work only at night. The list goes on and on, as the number of possible situations grows, and this is in our opinion the most important observation to make. Dozens of tentatives have been made over three decades to provide a system with good generalization properties, thus enabling it to perform correctly in most environmental condition; but no one has succeeded yet. The question to ask ourselves is then, probably: why? What we are inclined to conjecture is that the space of all the possible environmental conditions might be ways too large to be captured by over-specific fine-tuned algorithms. By the same token, if no such systems succeeded over all these years, then it could be that plain computer vision methods—like the ones we considered—have not enough expressive power to achieve such objective.

We can nevertheless still point out the importance of our algorithm, which can for example be adopted as a safety check within a larger lane detection system. Since it doesn't require complex operations and its behavior is deterministic, it can be used to compare the output coming from a higher level component and verify the plausibility of its results. This is an important functionality within complex autonomous driving

systems, as they are usually highly based on probabilistic reasoning.

Even after these considerations however, it seems clear how this system should be coupled with a learning component. For this reason we propose, as a future improvement of our work, the introduction of learning techniques into the pipeline. Deep learning models could indeed be able to learn how to react to weather and illumination changes only from samples, thus losing the strong bias introduced by manual tuning and possibly improving the adaptability of the system.

To this extent, some additional components would be needed. First, a quantitative evaluation of the detections has to be introduced. Among the others, a common performance measure consists in evaluating the number of points correctly detected by the system, with respect to the amount of false positives and negatives. Another possibility would be, instead, to consider the distance between the estimated position of each line and the ground truth. This distance can be expressed either in a form of mean error or, sometimes, as an area measurement.

Furthermore, a key requirement for learning techniques is the presence of labeled data. Several ways of obtaining them are available. On one hand, we could employ our current system to generate them, possibly altering some of the parameters on the fly according to the environmental conditions. In this fashion we could obtain a higher accuracy, despite a loss of generalization, which wouldn't be significant in this scenario. On the other hand, several dataset are nowadays publicly available. The information they contain is often of various nature, but in general they all include labeled images and, in some cases, their camera calibration parameters. This information would be enough for our purposes. However, the usage of such sources would entail some complications, as the data used for training the system would be acquired

with a different camera. In such case then, particular attention would have to be put into assuring a high degree of generalization also towards the characteristics of the sensors used.

Aside from machine learning techniques, we moreover think that the addition of a tracking stage to the pipeline would introduce a significant improvement. In fact, exploiting the now ignored temporal information, the system could potentially solve many of the misdetections we registered, as most of them were due to noise and outliers, which influence is usually episodic and often doesn't last for many consecutive frames.

Finally, we must mention that important developments could also be achieved in the future through the introduction of additional sensors, such as a GPS, an IMU or even a Lidar, although this would pose non-trivial data fusion issues. Likewise, continuations of our work could also lead to the deployment of the system, after the necessary optimizations, on a real-time platform.

Bibliography

[1] S. P. Narote, P. N. Bhujbal, A. S. Narote, and D. M. Dhane, "A review of recent advances in lane detection and departure warning system," *Pattern Recognition*, vol. 73, pp. 216–234, 2018.

[2] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, pp. 727–745, 2014.

[3] H. Zhou and H. Wang, "Vision-based lane detection and tracking for driver assistance systems: A survey," in *Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), 2017 IEEE International Conference on*, pp. 660–665, IEEE, 2017.

[4] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[5] D. A. Forsyth and J. Ponce, "A modern approach," *Computer vision: a modern approach*, pp. 88–101, 2003.

[6] D. Liebowitz and A. Zisserman, "Metric rectification for perspective images of planes," in *cvpr*, p. 482, IEEE, 1998.

[7] R. Laganiere, "Compositing a birds eye view mosaic," in *Proceedings of the Vision Interface Conference, Montreal, Canada*, pp. 382–387, Citeseer, 2000.

[8] I. S. Kholopov, "Bird's eye view transformation technique in photogrammetric problem of object size measuring at low-altitude photography," in *Proceedings of the International Conference "Actual Issues of Mechanical Engineering" 2017 (AIME 2017)*, Atlantis Press, 2017.

[9] J. Son, H. Yoo, S. Kim, and K. Sohn, "Real-time illumination invariant lane detection for lane departure warning system," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1816–1824, 2015.

[10] Y. Wang, D. Shen, and E. K. Teoh, "Lane detection using spline model," *Pattern Recognition Letters*, vol. 21, no. 8, pp. 677–689, 2000.

[11] S.-N. Kang, S. Lee, J. Hur, and S.-W. Seo, "Multi-lane detection based on accurate geometric lane estimation in highway scenarios," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014.

[12] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision computing*, vol. 22, no. 4, pp. 269–280, 2004.

[13] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen, "A novel lane detection based on geometrical model and gabor filter," in *Intelligent vehicles symposium (IV), 2010 IEEE*, pp. 59–64, IEEE, 2010.

[14] T. Veit, J.-P. Tarel, P. Nicolle, and P. Charbonnier, "Evaluation of road marking feature extraction," in *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pp. 174–181, IEEE, 2008.

[15] B. Southall and C. J. Taylor, "Stochastic road shape estimation," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, pp. 205–212 vol.1, July 2001.

[16] K. H. Lim, K. P. Seng, L.-M. Ang, and S. W. Chin, "Lane detection and kalman-based linear-parabolic lane tracking," in *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC'09. International Conference on*, vol. 2, pp. 351–354, IEEE, 2009.

[17] J. M. Alvarez, T. Gevers, Y. LeCun, and A. M. Lopez, "Road scene segmentation from a single image," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 376–389, Springer Berlin Heidelberg, 2012.

[18] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *IEEE transactions on intelligent transportation systems*, vol. 7, no. 1, pp. 20–37, 2006.

[19] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: a line segment detector," *Image Processing On Line*, vol. 2, pp. 35–55, 2012.

[20] United States Department of Transportation - Federal Highway Administration, *Manual on uniform traffic control devices for streets and highways, 23 CFR 655 F.* 2009.

[21] Ontario Ministry of Transportation, *Highway Traffic Act, R.S.O. 1990, c. H.8.* Queens Printer for Ontario, 1990.

[22] S. Beucher and M. Bilodeau, "Road segmentation and obstacle detection by a fast watershed transformation," in *Proceedings of the Intelligent Vehicles '94 Symposium*, pp. 296–301, Oct 1994.

[23] J. F. Khan, S. M. Bhuiyan, and R. R. Adhami, "Image segmentation and shape analysis for road-sign detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 83–96, 2011.

[24] N. Soquet, D. Aubert, and N. Hautiere, "Road segmentation supervised by an extended v-disparity algorithm for autonomous navigation," in *IEEE Intelligent Vehicles Symposium, Istanbul, Turkey*, pp. 160–165, 2007.

[25] J. P. Gonzalez and U. Ozguner, "Lane detection using histogram-based segmentation and decision trees," in *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pp. 346–351, IEEE, 2000.

[26] H.-Y. Cheng, C.-C. Yu, C.-C. Tseng, K.-C. Fan, J.-N. Hwang, and B.-S. Jeng, "Hierarchical lane detection for different types of roads," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 1349–1352, IEEE, 2008.

[27] D. Levi, N. Garnett, E. Fetaya, and I. Herzlyia, "Stixelnet: A deep convolutional network for obstacle detection and road segmentation.," in *BMVC*, pp. 109–1, 2015.

[28] G. L. Oliveira, W. Burgard, and T. Brox, "Efficient deep models for monocular road segmentation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4885–4891, Oct 2016.

[29] J. Ruyi, K. Reinhard, V. Tobi, and W. Shigang, "Lane detection and tracking using a new lane model and distance transform," *Machine vision and applications*, vol. 22, no. 4, pp. 721–737, 2011.

[30] C.-C. Wang, S.-S. Huang, and L.-C. Fu, "Driver assistance system for lane detection and vehicle recognition with night vision," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3530–3535, IEEE, 2005.

[31] J. G. Kuk, J. H. An, H. Ki, and N. I. Cho, "Fast lane detection & tracking based on hough transform with reduced memory requirement," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 1344–1349, IEEE, 2010.

[32] J. Kim and M. Lee, "Robust lane detection based on convolutional neural network and random sample consensus," in *International Conference on Neural Information Processing*, pp. 454–461, Springer, 2014.

[33] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, 2008.

[34] A. S. Huang, D. Moore, M. Antone, E. Olson, and S. Teller, "Finding multiple lanes in urban road networks with vision and lidar," *Autonomous Robots*, vol. 26, no. 2-3, pp. 103–122, 2009.

[35] J. Deng and Y. Han, "A real-time system of lane detection and tracking based

on optimized ransac b-spline fitting," in *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, pp. 157–164, ACM, 2013.

[36] H. Xu, X. Wang, H. Huang, K. Wu, and Q. Fang, "A fast and stable lane detection method based on b-spline curve," in *2009 IEEE 10th International Conference on Computer-Aided Industrial Design Conceptual Design*, pp. 1036–1040, Nov 2009.

[37] H. Marzbani, R. N. Jazar, and M. Fard, "Better road design using clothoids," in *Sustainable Automotive Technologies 2014* (I. Denbratt, A. Subic, and J. Wellnitz, eds.), (Cham), pp. 25–40, Springer International Publishing, 2015.

[38] Y. Wang, D. Shen, and E. K. Teoh, "Lane detection using catmull-rom spline," in *IEEE International Conference on Intelligent Vehicles*, pp. 51–57, 1998.

[39] E. W. Weisstein, "Least squares fitting," 2002.

[40] P. V. Hough, "Method and means for recognizing complex patterns," Dec. 18 1962. US Patent 3,069,654.

[41] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer vision, graphics, and image processing*, vol. 44, no. 1, pp. 87–116, 1988.

[42] Q. Li, N. Zheng, and H. Cheng, "Springrobot: A prototype autonomous vehicle and its algorithms for lane detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 300–308, 2004.

[43] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[44] Y. Ding, Z. Xu, Y. Zhang, and K. Sun, "Fast lane detection based on birds eye view and improved random sample consensus algorithm," *Multimedia Tools and Applications*, vol. 76, no. 21, pp. 22979–22998, 2017.

[45] E. Maggio and A. Cavallaro, *Video tracking: theory and practice*. John Wiley & Sons, 2011.

[46] M. Nieto, J. A. Laborda, and L. Salgado, "Road environment modeling using robust perspective analysis and recursive bayesian segmentation," *Machine Vision and Applications*, vol. 22, no. 6, pp. 927–945, 2011.

[47] K. Zhao, M. Meuter, C. Nunn, D. Müller, S. Müller-Schneiders, and J. Pauli, "A novel multi-lane detection and tracking system," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 1084–1089, IEEE, 2012.

[48] Y. Zhou, R. Xu, X. Hu, and Q. Ye, "A robust lane detection and tracking method based on computer vision," *Measurement science and technology*, vol. 17, no. 4, p. 736, 2006.

[49] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[50] R. Zuccolo, "Self-driving cars - advanced computer vision with opencv, finding lane lines." `https://chatbotslife.com/self-driving-cars-advanced-computer-vision-with-opencv-finding-lane-lines-488` Apr 2017. [Online; accessed 21-June-2018].

[51] P. Parodi and G. Piccioli, "3d shape reconstruction by using vanishing points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 211–217, Feb 1996.

[52] Z. Kim, "Geometry of vanishing points and its application to external calibration and realtime pose estimation," 2006.

[53] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection using b-snake," in *Information Intelligence and Systems, 1999. Proceedings. 1999 International Conference on*, pp. 438–443, IEEE, 1999.

[54] J. W. Demmel, *Applied numerical linear algebra*, vol. 56. Siam, 1997.

[55] W. K. Pratt, "Digital imaging processing: Piks inside, isbns: 0-471-37407-5 (hardback)," tech. rep., 0-471-22132-5 (Electronic).

[56] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 679–698, Nov 1986.

[57] M. G. Albanesi, "Time complexity evaluation of algorithms for the hough transform on mesh connected computers," in *CompEuro'91. Advanced Computer Technology, Reliable Systems and Applications. 5th Annual European Computer Conference. Proceedings.*, pp. 253–257, IEEE, 1991.

[58] F. Kluger, H. Ackermann, M. Y. Yang, and B. Rosenhahn, "Deep learning for vanishing point detection using an inverse gnomonic projection," in *German Conference on Pattern Recognition*, pp. 17–28, Springer, 2017.

[59] P. Denis, J. H. Elder, and F. J. Estrada, "Efficient edge-based methods for estimating manhattan frames in urban imagery," in *European conference on computer vision*, pp. 197–210, Springer, 2008.

[60] O. Barinova, V. Lempitsky, E. Tretiak, and P. Kohli, "Geometric image parsing in man-made environments," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 57–70, Springer Berlin Heidelberg, 2010.

[61] S. Workman, M. Zhai, and N. Jacobs, "Horizon lines in the wild," *CoRR*, vol. abs/1604.02129, 2016.

[62] T. Gevers, J. Van De Weijer, and H. Stokman, "Color feature detection," in *Color image processing: methods and applications* (R. Lukac and K. N. Plataniotis, eds.), vol. 9, pp. 203–226, CRC press, Oct. 2006.

[63] Wikimedia Commons, "Hsv_color_solid_cylinder.png: Sharkd," 2010. File: "RGB Cube Show lowgamma cutout a.png".

[64] T. Acharya and P.-S. Tsai, *JPEG2000 standard for image compression: concepts, algorithms and VLSI architectures*. John Wiley & Sons, 2005.

[65] Radiocommunication Sector - International Telecommunication Union (ITU), *Recommendation ITU-R BT.601-7 (03/2011) - Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*. 2011.

[66] Radiocommunication Sector - International Telecommunication Union (ITU),

Recommendation ITU-R BT.2020-2 (10/2015) - Parameter values for ultra-high definition television systems for production and international programme exchange. 2015.

[67] E. Hamilton, "Jpeg file interchange format," 1992.

[68] Wikimedia Commons, "Rgb-cube the rgb color model mapped to a cube. pov-ray source is available from the pov-ray object collection. sharkd," 2010. File: HSV_color_solid_cylinder.png.

[69] G. Saravanan, G. Yamuna, and S. Nandhini, "Real time implementation of rgb to hsv/hsi/hsl and its reverse color space models," in *Communication and Signal Processing (ICCSP), 2016 International Conference on*, pp. 0462–0466, IEEE, 2016.

[70] J.-W. Lee and J.-S. Cho, "Effective lane detection and tracking method using statistical modeling of color and lane edge-orientation," in *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, pp. 1586–1591, IEEE, 2009.

[71] J. M. Á. Alvarez and A. M. Lopez, "Road detection based on illuminant invariance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 184–193, 2011.

[72] R. Jiang, R. Klette, T. Vaudrey, and S. Wang, "New lane model and distance transform for lane detection and tracking," in *International Conference on Computer Analysis of Images and Patterns*, pp. 1044–1052, Springer, 2009.

[73] D. Obradović, Z. Konjović, E. Pap, and I. J. Rudas, "Linear fuzzy space based road lane model and detection," *Knowledge-Based Systems*, vol. 38, pp. 37–47, 2013.

[74] V. Gaikwad and S. Lokhande, "Lane departure identification for advanced driver assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 910–918, 2015.

[75] J. C. McCall and M. M. Trivedi, "An integrated, robust approach to lane marking detection and lane tracking," in *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 533–537, IEEE, 2004.

[76] P.-C. Wu, C.-Y. Chang, and C. H. Lin, "Lane-mark extraction for automobiles under complex conditions," *Pattern Recognition*, vol. 47, no. 8, pp. 2756–2767, 2014.

[77] S. Jung, J. Youn, and S. Sull, "Efficient lane detection based on spatiotemporal images," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 289–295, 2016.

[78] J. Goldbeck and B. Huertgen, "Lane detection and tracking by video sensors," in *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, pp. 74–79, IEEE, 1999.

[79] S. Nedevschi, R. Schmidt, T. Graf, R. Danescu, D. Frentiu, T. Marita, F. Oniga, and C. Pocol, "3d lane detection system based on stereovision," in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pp. 161–166, IEEE, 2004.

[80] H. Marzbani, R. N. Jazar, and M. Fard, "Better road design using clothoids," in *Sustainable Automotive Technologies 2014*, pp. 25–40, Springer, 2015.

[81] Q. Chen and H. Wang, "A real-time lane detection algorithm based on a hyperbola-pair model," in *Intelligent Vehicles Symposium, 2006 IEEE*, pp. 510–515, IEEE, 2006.

[82] M. Nieto, L. Salgado, F. Jaureguizar, and J. Arróspide, "Robust multiple lane road modeling based on perspective analysis," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pp. 2396–2399, IEEE, 2008.

[83] C. Lipski, B. Scholz, K. Berger, C. Linz, T. Stich, and M. Magnor, "A fast and robust approach to lane marking detection and lane tracking," in *Image Analysis and Interpretation, 2008. SSIAI 2008. IEEE Southwest Symposium on*, pp. 57–60, IEEE, 2008.

[84] J. Guo, Z. Wei, and D. Miao, "Lane detection method based on improved ransac algorithm," in *Autonomous Decentralized Systems (ISADS), 2015 IEEE Twelfth International Symposium on*, pp. 285–288, IEEE, 2015.

[85] A. Lopez, C. Canero, J. Serrat, J. Saludes, F. Lumbreras, and T. Graf, "Detection of lane markings based on ridgeness and ransac," in *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pp. 254–259, IEEE, 2005.

[86] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, and C. De Boor, *A practical guide to splines*, vol. 27. Springer-Verlag New York, 1978.

[87] C. De Boor and J. R. Rice, "Least squares cubic spline approximation i-fixed knots," 1968.