UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXI Ciclo

Study and development of road boundaries detection applications for embedded platform implementation

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Massimo Bertozzi

Dottorando: *Francesca Ghidini*

Anni 2015/2018

To my grandparents

Abstract

In the last three decades, we have assisted to an accelerated growth of intelligent systems for automated driving. These advancements are pushed by the idea of reducing traffic accidents and by critical economic impacts.

Advanced Driving Assistance Systems (ADAS) are nowadays readily available even in small and cheap car models. Autonomous lane keeping and adaptive cruise control are just some examples. Moreover, the automotive industry and many technology developers are working on highly-automated vehicles.

All these systems require complete knowledge of the vehicle surroundings. In this scenario, computer vision based on the use of cameras plays a fundamental role since cameras are cheap, easily integrable on a vehicle and they provide a considerable quantity of information. The main drawback of these sensors is the amount of processing required for extracting useful information from pixels. At the same time, space is one of the most exiguous resource on a car, and this put a challenge in implementing computer vision techniques on resource-constrained embedded platforms.

Highways and non-urban roads are well limited by lane markings. However, urban areas are more complex scenarios where the drivable area could be delimited by lane markings, parked cars, walls, and curbs. For these reasons, the use of ADAS is usually limited to highway scenarios where markings are visible and the geometry of the road is regular.

The purpose of this research is to develop robust perception systems for autonomous vehicle using computer vision methods able to run with real-time performance on an embedded platform. The works in this field are focused on the specific problem of road boundaries detection, in particular curbs and barriers. Both these structures represent an essential input for many high-level applications (for instance: path-planning, ego-lane estimation, obstacle avoidance, and localization) especially when lane markings are not present or when the complexity of the scene is high. The main contributions of this research are: two innovative approaches to curbs detection, one based on stereo vision and the other based on mono camera and deep learning, and a new approach to road barriers detection with a stereo camera.

The approach to curbs detection based on stereo vision takes inspiration from methods present in literature with the objective to increase detection quality and enabling the execution in real-time on embedded hardware. It makes extensively use of Digital Elevation Map (DEM) in order to search for specific height variations in the three-dimensional space. 3D models, representing the shape and position of curbs with respect to the vehicle, are extracted from the elevation map and a tracking system temporally integrates the measures. Quantitative and

qualitative analysis demonstrated the reliability of the system. Furthermore, this approach was successfully used as input for many high-level applications on a highly-automated vehicle.

The second approach to curbs detection is born from the need to overcome the limitations of stereo-based approaches. For this reason, a different method, based on deep learning and mono camera was chosen, not much work has been done in this field so far. Curbs detection is in this case addressed as a semantic segmentation problem, and a very light-weight model has been trained with promising results: the detection range is increased with respect to stereo vision methods and moreover, this algorithm provides robust results even in poor lighting conditions.

With respect of the approaches in literature, which are focused on specific structures, in this work barriers detection problem is addressed in its most generic meaning: detecting all the vertical structures defining road or driving corridor borders. Walls, guard-rails, fences and hedges are just some examples of the structures that fall into this category. In order to save computation time, this algorithm is designed to share its processing with the stereo-based curbs detection algorithm. The reliability of the proposed algorithm was proven with extensive tests in a real-world scenario and barriers were successfully used as input for path planning. This algorithm was able to run in real-time on an embedded platform together with the curbs detection algorithm, in this way a wide range of structures can be detected with a minimal computation impact.

During this research, emerged that several constraints accompany the process of developing computer vision solutions for the automotive industry: reliable and robust results are required as well as methods for handling adverse environmental conditions. Moreover, the software has to be designed in order to require as little as hardware as possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last thirty years advancements in sensors, perception algorithms and low-cost embedded processors have resulted in an accelerated growth of intelligent systems for vehicle technology.

These technologies are designed in order to save lives, reducing the number of traffic accidents, and to help the driver improving its confront. Since most of the automobile accidents are due to driver errors (such as aggressive driving, use of drugs and alcohol, inexperience), self-driving cars could drastically reduce the number of accidents helping to save thousands of lives.

Another source of saving is represented by the time spent in the car: with self-driving cars, people will be free to do other activities than driving, for instance: working, napping, reading or simply relax and enjoy the travel. In 2017, 3.1 trillions of vehicle miles were traveled in the U.S [4], that is more or less 10,000 miles per person every year. Assuming that the average travel speed is 60 miles per hour and that each vehicle holds only one person at a time, this translates to around 163 hours spent in the car per person and a total of 53 billion hours. Hypothesize that the value of the time spent on the car doing what we want instead of driving is about 2$, this would imply a saving of $100 billion every single year. Moreover, considering fewer conservatives assumptions, like 5$ hour value and a more realistic average speed of 30 miles per hour the saving can reach 500 billion dollars.

Self-driving cars will make driving more efficient also in terms of better traffic flow and less fuel consumption. Since cars will be automated there will be less possibility of accidents caused by human error and therefore fewer congestions and traffic jams. Other benefits are related to an increased energy efficiency: a significant saving in fuel expenses can be achieved by systems able to control vehicle speed, keeping it constant and, when it is necessary, optimizing acceleration and braking profiles.

Autonomous lane keeping and adaptive cruise control are just some of the most widespread systems that are now readily available even in small and cheap car models. They can either warn the driver in case of dangerous situations (e.g. lane departure warning and blind spot detection) or they can control the steering wheel and the brake in order to avoid collisions (e.g., adaptive cruise control and advanced emergency braking system). Moreover, many car makers, technology developers, and original equipment manufacturer (OEM) are working on automated vehicles requiring little or no human interaction.

All the systems mentioned above rely on a complete knowledge of the vehicle environment. Therefore the analysis and comprehension of the road scenario are one of the most critical and complex tasks for an advanced driving assistance system (ADAS) or an autonomous vehicle. This means to detect all the information related to the road (locate its boundaries and determinate the position and orientation of the ego-vehicle inside it) and to the other road users (e.g., cars and pedestrians). A robust and precise model of the driving space is the basis for many high-level applications such as path planning, lane keeping and departure warning, adaptive cruise control, and collision avoidance.

In this field, artificial vision plays a fundamental role because, with respect to other sensors such as RADAR and laser scanners, it brings many advantages: high information content, low power consumption, low cost and simple integration inside the vehicle. Furthermore, many road features, like lane markings and traffic signs, have only visual properties.

Unfortunately, the main drawback of camera sensors is the amount of processing required for extracting useful information from pixels. In the case of ADAS or self-driving cars prototypes, it is possible to use a standard general-purpose PC located

in the vehicle trunk. However, space is an exiguous resource in a car and, in a final product, the perception system has to be invisible, and it has to share the available space with other units. High accuracy requirements lead to the use of computationally-intensive computer vision techniques, and this puts a challenge in implementing them on resource-constrained embedded platforms. Therefore perception algorithms have to be designed in a way that requires as little hardware as possible.

The purpose of this research is to develop robust perception systems for autonomous vehicle using computer vision methods able to run with real-time performance on an embedded platform. The works in this field are focused on the specific problem of barriers and curbs detection.

1.1 Motivation and Problem Statement

Road environment reconstruction is one of the most relevant topics for self-driving car and ADAS. Lane departure warning (LDW) and lane keeping (LK) are just some examples of the technology with which many vehicles are equipped nowadays. LDW systems monitor the ego-vehicle position inside the lane and, in case of dangerous situations, they can warn the driver. LK systems instead are also able to take control of the car in order to reduce or avoid the damage to its occupants. Other systems, called highway pilot, can keep the safety distance from the vehicle ahead while driving in the lane. However, all these systems require a complete an accurate scene understanding and their use is limited to well-structured roads with visible lane markings, clear weather, and sufficient illumination conditions.

Highways and non-urban roads are, in general, well limited by lane markings. However, urban areas are more complex and the drivable space could be delimited by lane markings, parked cars, walls, fences, hedges and curbs of quite different heights. The reliability and the accuracy of a perception system based on computer vision are also strongly influenced by illumination and weather conditions as well as the large variety of street configurations. In figure 1.1 are showed some examples of challenging conditions.

Since the level of vehicle autonomy is proportional to the robustness of the

(a) reflection (b) rain (c) traffic

(d) night (e) snow (f) unmarked road

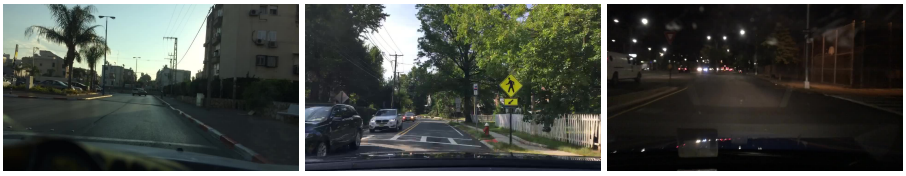
Figure 1.1: Some examples of challenging conditions taken from the Berkeley Deep Drive dataset [1].

perception system, it is necessary to develop algorithms for reliable detection of generic road edges able to deal with all the possible adversities.

1.1.1 Motivation for Curbs Detection

One of the most widespread structures that are usually present in urban scenarios is curb. They delimit road boundaries, for instance in the form of sidewalks, traffic isles or roundabouts, as shown in figure 1.2.

They can be used as input for center lane estimation and path planning. Curbs are also an essential feature for the localization task: some fully autonomous cars [5] use a pre-built map of the environment that they match against the detected curbs in order to estimate the vehicle pose. These techniques are able to obtain an accurate pose estimation even in the absence of a GPS signal, for instance in the presence of buildings and trees. Curbs are also a separator between road and sidewalks, so they can also give useful information for pedestrian detection.



(a) sidewalk (b) road delimiter (c) traffic isle

Figure 1.2: Examples of curbs representing road limits (from the Berkeley Deep Drive dataset [1]).

This kind of delimiters is usually not detected by the standard object detectors, because of their low height. Most of the obstacles detectors, in order to be robust to the noise, require to manage just obstacles with a minimum height, as in [6]. Also, a v-disparity approach [7] is not suitable for curbs detection since it has difficulty in identifying low objects since there is not sufficient data for the vertical histogram. Moreover, near intersections and pedestrian crossings, curbs tends to be fragmented in short segments which make the detection very difficult. Curbs detection is also a harder problem than lane detection since the contrast between curbs and road or sidewalks may not be very pronounced and can significantly vary due to different height profiles and colors along the curb. Specialized and sophisticated algorithms are therefore necessary.

1.1.2 Motivation for Barriers Detection

Among the various free space delimiters also the detection of road barriers is useful for a significant number of applications. The term barrier includes all those vertical structures that help to delimit the free space. Some examples of barriers are shown in figure 1.3. Guard-rails, wall, fences, and hedge are natural road delimiters while standing, or parked vehicles can be used to estimate a driving corridor for unmarked roads of when markers on the ground cannot be detected.

(a) fences (b) parked cars (c) walls

(d) hedge (e) standing vehicles (f) guard-rail and wall

Figure 1.3: Examples of barriers representing road limits (from the Berkeley Deep Drive dataset [1]).

In some autonomous driving vehicles, like the one described in [8], barriers are fused with lanes in order to obtain the ego lane estimation. These kind of boundaries are very challenging to detect because they exhibit a high variety in visual appearance, so there is the need for a system that can detect generic road edges.

1.2 Contribution and Thesis Outline

The aim of this work is the development of innovative algorithms for barriers and curbs detection both designed to run efficiently on low-cost embedded hardware.

The main contributions of this work are:

- development of a stereo vision based curbs detection algorithm which has the objective of increasing reliability and range of detection with respect to state of the art methods. Moreover, this method is developed in order to be executed on an embedded platform;

- stereo vision based technique for the detection of road barriers. With respect to the state of the art methods that are focused on specific structures, in this research, the problem is addressed in order to detect generic structures. This algorithm is developed specifically to be executed on an embedded platform with a minimum computational impact;
- development of a monocular curbs detection algorithm based on deep learning techniques with the objective of overcoming the limitation of the stereo visions methods. Curbs detection is addressed as a semantic segmentation problem, this kind of approach has not yet studied.

The rest of this thesis is structured as follows:

- Chapter 2: provides background concepts and literature overview of previous works in these fields;
- Chapter 3: describes the proposed stereovision-based approach for curbs detection;
- Chapter 4: a presentation of the algorithm for barriers detection;
- Chapter 5: describes the monocular curbs detection algorithm based on deep learning;
- Chapter 6: concludes this thesis with outlines and future work.

Chapter 2

Background

In this chapter related works on curbs and barriers detection are reviewed.

Since the proposed stereovision-based curbs and barriers detection algorithms make extensively use of Digital Elevation Map (DEM), the literature regarding this kind of 3D environment representation structure will be reviewed.

Deep learning-based state of the art solutions for image segmentation will be presented as well since they represent the basic concepts for the development of the monocular curbs detection algorithm.

2.1 Digital Elevation Map

Grid-based approaches play a fundamental role in the representation of vehicle 3D environment which includes road boundaries, free space, both static and dynamic obstacles.

DEM and occupancy grids are the most famous representation techniques in the literature. These structures represent a portion of the three-dimensional environment by mean of a horizontal grid, divided into cells, where each one stores information about its portion of the 3D space.

Occupancy grids were originally proposed by Elfes [9]: in this representation, every cell in the grid contains a random variable representing the likelihood of the

cell to be free or occupied.

DEMs are, instead, a height-based representation of the environment: every cell can store a set of height-related statistics like minimum, maximum or average height. DEMs are sometimes also described as having 2.5 dimensions since the description is not complete: for instance tunnels, tree branch and bridges cannot be completely represented.

One of the major contributions for the DEM based approach was proposed by Oniga [10], where a system for road surface and obstacle detection is described. Since DEM can be large data structures representing a vast portion of the space, they can be used for terrain mapping [11] even in unstructured environment [12] and also in the case of planetary surface exploration [13]. In [14] 3D grid mapping is used to find traversable and non-traversable regions. DEMs are also suitable for low obstacles detection, like curbs or traffic isles [15, 16, 10]. The use of DEM for curbs detection will be described in details further on. DEMs can also be used for indoor pedestrian tracking as described in [17].

One of the major benefits introduced by the DEM-based approach is the versatility: DEM can be used for a wide variety of applications, and they can be built in real-time using the vast majority of 3D sensors. Moreover, maps, in general, allow to consider the sensor specific noise model, so they are also suitable for applications of multi-sensors data fusion [18].

It is possible to define DEM over different coordinates system. The most common approach is to describe the DEM in a Cartesian space, assigning a constant length and width to every cell. These maps are easy to handle; however, they have the drawback that image projection of distant cells is too small and they tend to be empty since only a few disparities measures are associated with them. As an alternative, grids can be defined in the column disparity space *(u,d)*, with this representation the effect of empty cells is intrinsically handled since, the image, all the cells have a constant dimension. Moreover, any cell does not lie outside the camera field of view. These two different methods are shown in figure 2.1.

In [19] it is proposed a stochastic approach to evaluate generic discrete representations of the three-dimensional world consisting on a comparison between a local

(a) Cartesian DEM (b) column disparity DEM

Figure 2.1: a Cartesian DEM (left) and a column disparity DEM (right) from the bird-eye-view. The red triangle represents the field of view (FOV) of the camera. It is possible to notice how all the cells of the (u, d)-space DEM are inside the FOV and how the problem of empty cells is intrinsically handled.

perceived representation with the corresponding previously computed ground truth. Accurate depth sensing and precise localization are used to generate the ground truth. DEMs and stereo vision data are used as a test case.

One of the major drawbacks of DEM representation is that, when they are built using minimum, maximum or average height of the points inside the cell, structures such as tunnels and bridges cannot be appropriately modeled. In [20] is presented a building algorithm which looks for gaps in the vertical structures. Another extension is proposed by Triebel et al [21] wherein each cell a set of surface patches are stored. In order to achieve a robust classification of traversable space, some approaches [22, 23] use a combination of occupancy grids and DEM.

In [24], Danescu et al. propose a particle-based dynamic elevation map in which every cell of the map stores its height and also a probability distribution of the speed in order to classify dynamic obstacles correctly. A set of particles is used to represent the map; every particle maintains height, position and speed measures. Particles can move from one cell to another on the basis of their speed vectors. They can also be created, destroyed and multiplied employing an importance-resampling technique

driven by stereo vision sensor measurement data.

2.2 Related Work

In this section, related works on curbs and barriers detection will be reviewed.

2.2.1 State of the Art on Curbs Detection

In unmarked and urban roads, the detection of curbs is a fundamental element for both ADAS and self-driving car perception systems since they define road borders in most of the driving scenarios. Curbs are also essential features for vehicle localization task since they can provide, for instance, the location of sidewalks, intersections or roundabouts when the presence of trees or buildings occludes the GPS signal. In [25] a localization method based on curbs detection is proposed: curbs are detected using a laser scanner, which provides a 3D point cloud, and are then used as input of a Monte Carlo localization algorithm [26] able to produce an accurate estimate of the vehicle position even in the absence of GPS signal. In [5] is described a robotic architecture of a self-driving car: its localization module uses a pre-built map of the vehicle environment, measured curbs are matched against the map in order to get an estimation the vehicle position.

Curbs can be on both sides of the road, in this case, they are usually a long continuous structure parallel to lane markings. However, at intersections, roundabouts or pedestrian crossing, they can be made of different shorts segments, these situations make curbs detection a challenging task. Moreover, because of their low height, it is not possible to use standard object detectors, such as Stixels [6] or V-disparity [7], since they need to model object with certain a minimum height in order to be robust to noise artifact. Curbs detection is also a harder problem with respect to lane perception, since the contrast between curbs and road or sidewalks may not be very definite and can significantly vary due to different materials, height profiles and color patterns along the curb.

For these reasons, several approaches have been developed to address this task explicitly.

The algorithm proposed by Se and Brady in [27] is probably one of the first methods addressing curbs detection task. This algorithm detects curbs from clusters of parallel straight lines in the 2D image by mean of Hough transformation. Edge points that serve as input to the Hough transformation are detected using the Canny edge detector [28]. In [29] curbs are extracted with a texture based filter: the algorithm uses local binary patterns because of their accuracy on segmentation and for their suitability to be ported on an embedded platform, and a neural network as a classifier. Prinet et al. [30] propose a curbs detection algorithm based on a mono camera with fish-eye lens. Curbs are modeled using both visual features and geometric characteristic: an SVM classifier is firstly used to locate curbs on the image space, the 3D geometry is then estimated by making assumptions about the scene and the camera. Temporal integration is used to filter out false positives. With respect to other methods, mono camera based algorithms allow detections at more considerable distances, but they are susceptible to false positives typically caused by edges in the image, such as lane markings, shadows or cracks on the road.

For these reasons, the work in [27] was extended in [31] by Turchetto and Manduchi adding 3D information obtained from stereo vision. The Hough accumulator is now weighted by a function proportional to the scalar product of brightness gradient and 3D elevation gradient. In [32] the weighted function is further extended with the introduction of a more sophisticated weight function considering the estimated surface curvature as well. Enzweiler et al. [33] present a curb classifier that operates on both 3D data from stereo vision and 2D intensity image where a classifier based local receptive field features is used in conjunction with a multi-layer neural network. Classifier results are then used as an additional input on a lane detection system. In [34], a method for low obstacles detection (including curbs) based on the merging of 3D and 2D information is presented. A combination of intensity and 3D features is used in order to extract candidate obstacles. Candidates are then tracked, clustered and used for spline fitting. In [35] stereo vision is used to detect a flat area with v-disparity. In that flat area, a 16-dimensional descriptor, based appearance, geometry, and 3D information, is used to describe curbs that are classified with SVM. Curbs are then used for road and sidewalk detection. Approaches which use 3D information from

stereo vision in conjunction with visual features are less sensitive to false positives and allow detections to larger distances, however, they suffer from vast computing time. Nonetheless, stereo vision systems are cost-efficient and can provide 2D and 3D information.

In [36] is presented a purely stereovision based method for step detection in the field of mobility aids for visually impaired people. This method estimates a piecewise planar model of the scene and a tensor voting technique is applied for finding globally consistent normals, and plane segments are then extracted with clustering. Fernandez et al. [37] extract curbs measures from a 3D point cloud gained with a stereo camera using curvature features and Conditional Random Fields (CRF). Purely stereo vision based algorithm suffer for a limited detection distance, but they are less sensitive to false positives. Most of the purely stereo vision methods are based on the use of DEM, since the curbs detection method presented in this work make extensively use of the DEM these methods will be presented in detail in the next subsection.

Apart from stereo vision, input from other 3D sensors, such as laser scanners, are possible. Since these sensors are more accurate than stereo cameras having low noise levels and since they generally require less computational resources, extensive research has focused on these sensors as well. Pollard et al. [38] use three LIDAR for a Personal Mobility Vehicle (PMV). Curbs and steps are detected with the analysis of altitude first derivative. In [39] curbs are obtained from successive readings of a 2D laser scanner measurement system. A system based on the extended Kalman filter system is used to filter and segment input data simultaneously. In [40] an omnidirectional LIDAR with 64 beams is used: dense height images are built on laser measurements, with this compact representation, curbs are detected by edge filters. An expected road model is then used to match the extracted curbs measurements. Kellner et al. [41] present and compare different segmentation and classification methods for the detection of curbs features from a 3D laser scanner. Furthermore, it is described a grid architecture where each cell of the map contains linear model parameters which locally describes the detected curb.

Different frameworks where LIDAR are fused with mono cameras are proposed. The approach described in [39] is extended by Kodagoda et al. [42] by adding a

mono camera. In [43] 3D information from the laser is used to detect curbs searching for a specific height variation, camera image is then used to expand the identified curb along the brightness edge defined by it. Then curbs are tracked on the basis of vehicle ego-motion. Tan et al. [44] proposed a curb detection algorithm based on curbs geometric property, a color camera and Velodyne, which can detect curbs up to 30 meters.

Time of flight camera is used in [45], where it is proposed a modified version of RANSAC, called Connected Component RANSAC (CC-RANSAC), where the fitting score is the largest number of model inliers that are connected in the same set. The improved CC-RanSaC is used to compute scene most relevant planes, and curbs are defined as the boundary between planes.

There are different kind of models for representing curbs shape: linear models [27, 31], polynomial [46, 47, 16], polylines [48] or splines [15]. Also non-parametric models are possible [40, 43].

2.2.1.1 Digital Elevation Map for Curbs Detection

DEMs are used on a great number of works in the literature.

Maye et al. [49] present a curbs detection algorithm for pedestrian robot navigating on urban environment: a piecewise planar model of the scene is constructed, and curbs are detected as plane boundaries segments. In the first step, a DEM is built using 3D measurements from a laser, in each cell is stored an error model. Planes are estimated with a mixture of linear regression models on the DEM.

Siegemund et al. [46] propose a curb detection algorithm where curbs horizontal and vertical geometry are extracted from a column-disparity DEM. Height measures are assigned to a Conditional Random Field (CRF) which is aligned to the DEM. The underlying idea is that even at great distances the average height levels of the curb adjacent surfaces (for instance street and sidewalk) still differ. The curb is defined as a third order polynomial dividing those adjacent surfaces. The approach is divided into two steps. In the initial step a 3D point cloud, gained from the stereo camera, is assigned to the curbs adjacent surfaces with belief propagation on the CRF. Surfaces are then reconstructed in the second step. The work in [46] is extended in [16] where

the 3D points from the stereo camera are classified as road or sidewalk using a temporally integrated CRF based on the use of Kalman filter.

Michalke et al. [50] use a DEM built using stereo camera 3D measurements and temporally integrated in order to reduce the noise. On the filtered DEM are then applied odd Gabor filters in order to extract edges due to elevation change.

Oniga et al. conducted one of the most extensive works in the field of curbs detection with DEM. In his first work [48] a DEM is built from the 3D stereo data. Height variations are detected applying edge detection filters on the DEM. The 3D noise is reduced employing of a multi-frame persistence map: edges are filtered with vehicle ego-motion and only persistent points are kept. Edges are then used as input for the Hough transformation, and segments and chain of segments are extracted from the accumulator. The work was then extended in [47] where curbs are modeled as cubic polynomials computed with a RANSAC approach. Vertical profile and height from the ground are then extracted. In a later work [15] a cubic-spline model is used instead. In [10] the DEM is used for detecting simultaneously road surface, obstacles and sidewalks improving the detection using a density map.

2.2.2 State of the Art on Barriers Detection

The problem of road boundaries detection for autonomous vehicles and driver assistance technologies has been studied for decades. Extensive work has been done for lane markings and curbs detection. However, there are other kinds of objects that are of particular interest for road boundaries detection: barriers. The term barrier includes all those vertical structures, such as guard-rails, walls, fences, lines of standing vehicles that help to delimit the free space or the driving corridor or that can act as an additional input for lane keeping

In the literature, only a few publications have explicitly focused on the detection of barriers so far.

Aufrere et al. [43] present an approach for detecting continuous structures (curbs, barriers, and walls) alongside and in front of the vehicle. As for curbs, the 3D information gained from a laser scanner is used to detect barriers searching for specific height variations. The laser scanner is also used in [51] where it is presented an approach for

the detection and classification of various types of safety barriers (for instance Jersey or steel safety barriers).

Radar and mono camera are fused in [52] for the specific task of guard-rail detection. The approach is divided into two steps. In the first step, edge detection is applied to the input image, only the edges with an orientation comparable to a guard rail are kept. During the second step, the algorithm analyzes the edges and, according to their length, determines if the object is a guard rail or not. Only objects with speed lower than 5m/s are considered for guard-rail detection.

Stereovision is used in [53] for lane and guard-rail detection. The lane surface is firstly estimated, 3D points are then classified as belonging or not to the road surface on the basis of the previously estimated lane model. Points lying above of the lane surface are used for guard-rail detection. The points are analyzed through lateral distance histograms: their maximum locate the guard-rail, results are then temporally integrated to increase algorithm strength. Scharwachter et al.[54] propose a guard-rail detection system for the specific case of highway scenario using a stereo camera. A feature encoding method is combined with geometry information: candidate guard-rails are localized in the image with Hough transformation, for each potential guard-rail linearity constraints for depth and height are applied. Guard-rail appearance information is encoded with a bag-of-feature representation.

Monocamera is used in [29] for guard-rail detection. This algorithm, in a similar way to a structure from motion approach, tracks Harris features with the method of Lucas and Kanade [55] extracting 3D information from them.

2.3 Deep Learning for Semantic Segmentation

The problem of semantic segmentation consists on assigning a meaningful class to every pixel of the image, as the example shown in figure 2.2. Semantic segmentation is a challenging task since it requires to combine dense pixel-level accuracy to multi-scale reasoning. Complete scene understanding is one of the key tasks for computer vision since nowadays many applications use image data. Some of these applications include: autonomous driving [2, 3, 56, 57], medical analysis [58, 59], search engines

[60, 61], human-machine interaction [62, 63] and so on.

Figure 2.2: An example of a pixel-level segmented image from Cityscapes dataset [2].

Other three main computer vision tasks are image-level classification, object detection, and instance segmentation. Image-level classification means to treat the image as a single category, while object detection refers to the localization and recognition of different objects in the image and instance segmentation means to the process of pixel-level recognition of multiple instances of different objects.

As described in [64], image segmentation task, can be formally expressed with the following formulation: a state from a label space $L = \{l_1, l_2, ..., l_n\}$ has to be assigned to each element of a set of random variables $\chi = \{x_1, x_2, ..., x_k\}$ where each label l_i represents a class (for example: traffic sign, road markings, sidewalk, pedestrian etc.) and χ is a 2D image.

Over the years have been developed an enormous number of techniques for image segmentation and they can be divided into two main categories: deep learning methods and traditional computer vision methods.

Before the deep learning revolution, traditional image segmentation methods used hand-crafted features like HoG [65, 66], SIFT [67] or SURF [68], just to name a few. There are also some edge-based segmentation techniques [69, 70]. Other classes of methods use Markov Random Network or Conditional Random Fields and their variations [71, 72, 73]. In the medical image domain, thresholding methods are quite effective since images are usually in grayscale [74, 75, 76]. Some unsupervised

algorithms, like for example k-means [77], are used to cluster similar pixels.

As with image classification, convolutional neural networks (CNN) have had enormous success on segmentation problems since they surpassed, with a large margin, traditional methods concerning both accuracy and efficiency.

Since CNN where initially designed for image classification problems, one of the most popular earlier deep learning segmentation technique was based on a sliding window classification approach where every pixel is separately labeled using a patch of image around it [78, 79, 80, 81, 82, 83, 84], as shown on image 2.3. The image is divided into many smaller patches which are treated as a classification problem. However, these approaches ended up on very computationally expensive methods since every pixel of the image requires to solve a separate classification problem.

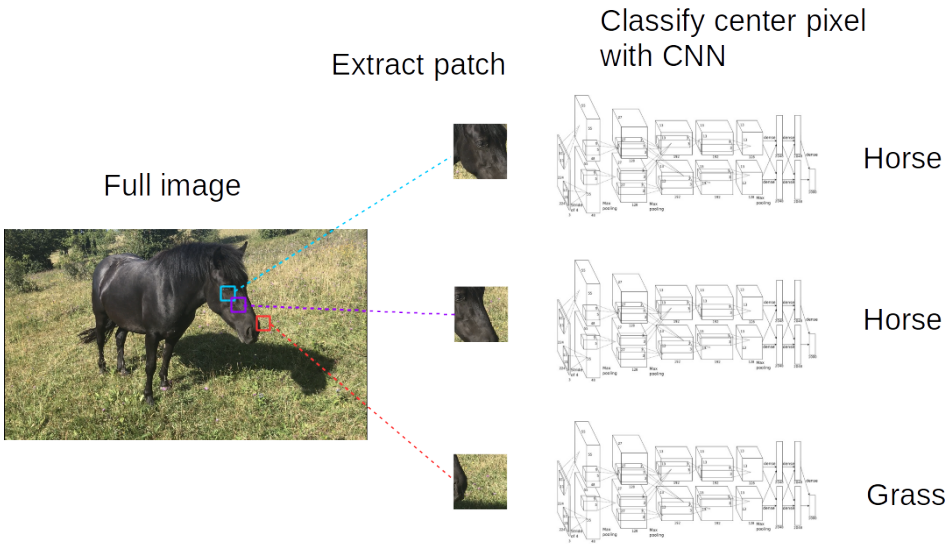


Figure 2.3: Image segmentation with sliding window example.

From 2014 CNNs are adapted explicitly for segmentation and fully convolutional

networks (FCN) [85] without fully-connected layers became very popular. In this way, segmentation can be applied to any image size, and they are also much faster with respect to sliding windows approaches. In the following years more or less all the successive state-of-the-art techniques followed this approach. An example of this network architecture is shown in figure 2.4.

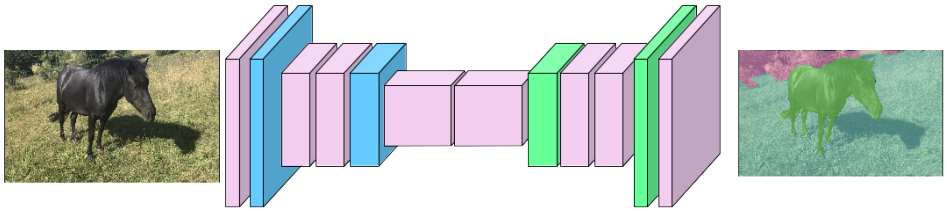


Figure 2.4: Image segmentation with fully convolutional network architecture example.

Rather than applying all the convolutions to the full resolution image, these nets go through a small number of convolutional layers at the original resolution and then downsample the feature map using max-pooling and strided convolutions. The first part of the network consists of a series of downsampling and convolutions. A the end, rather than transitioning to a fully-connected layer, the spatial resolution is increased in the second half of the network. In this way, the output vector has the same size as the input image. These architectures can be made very deep since they work at lower spatial resolutions for may layers.

FCN [85] is the first work proposing the idea of replacing fully connected layers with fully convolutional layer. In this architecture, the output size is the same of input, which is an essential element for image segmentation. The model is trained end-to-end: it takes as input arbitrary sized vectors and produces same size output.

FCN adopted the VGG-Net [86] and, at that time, achieved the state of the art for the dataset PASCAL VOC [87] with an inference time of one fifth of a second.

In [88] deconvolution network, which consists in deconvolution and un-pooling layers, is introduced. Deconvolution is the opposite operation of convolution, and it can recover the original input size.

According to [89] CNNs final layers responses are not sufficiently localized for accurate object segmentation. For these reasons, fully-connected CRF are usually combined with CNNs final layers.

Dilated convolutions [90, 89] were broadly applied in image segmentation tasks in order to produce dense predictions. One kind of these networks is called dilated residual networks (DRN) [91]: it can reduce artifacts produced by regular a dilated convolution operation.

At the beginning of deep learning, many research used the VGG architecture [86] as the network backbone. However other architectures have been developed. With the release of ResNet [92], the semantic segmentation task known a new revolution. From the intuition behind ResNet, it was possible to introduce new models which were able to reach and surpass state-of-the-art performances by employing shallower and more lightweight networks. The reduction of computational cost and memory footprint is particularly critical for many real-time applications. ResNetXt [93] was presented as the next generation of ResNet.

Also GoogleNet [94] has been further developed with Inception-v2, Inception-v3 [95], Inception-v4 and Inception-ResNet [96] architectures.

Chapter 3

Curbs Detection

In this chapter, the stereo vision based curbs detection algorithm will be described in detail.

This work aims is to provide an accurate estimation of the shape and the position of road curbs with respect to the ego vehicle. This algorithm works on urban areas handling dynamic and static obstacles as well as complex road geometries like intersection, curves, roundabouts and parking slots.

With respect to other stereo vision based approaches in literature, that usually have a maximum detection range of 20m, this algorithm has an increased detection range that allows its use as an essential input for ego-lane estimation, not only in low-speed scenarios.

Furthermore, compared to other state of the art approaches, the algorithm is designed in order to be efficient in terms of computing time and memory consumption since it will be executed on embedded hardware. Moreover, the computational resources must be shared with other algorithms in order to realize a highly automated vehicle.

This chapter is organized as follows. Section 3.1 is an overview of the proposed method for curbs detection based on stereovision. In Section 3.2 the coordinate frame system and the chosen method for representing efficiently the 3D space will be described. In Section 3.3 the 3D model for describing curbs position and shape is

presented. Sections 3.4, 3.5 and 3.6 are respectively a detailed description of the curbs detection algorithm main steps: the DEM building procedure and low-level and high-level processing. Experimental results and benchmark procedure will be analyzed in Section 3.7.

3.1 Algorithm Overview

The proposed algorithm takes inspiration from the work described in [47] which rely on the idea that the main feature of curbs, in the three-dimensional space, is a sharp elevation change.

A stereo camera is chosen as a sensor since it provides dense 3D data and it can be easily integrated inside a car. Moreover, stereo camera data is shared between several perception algorithms composing the autonomous system. In the test vehicle, the sensor is mounted on the roof, beyond the windscreen, framing the space in front of the car even at great distance.

Most of the processing is done considering a DEM-based representation of the three-dimensional environment. The DEM structure compresses the 3D data in a grid: it enables real-time processing and it has low memory footprint while it keeps the connectivity between adjacent 3D locations. In every cell of the DEM, the minimum height of all the points laying inside it is stored.

An overview of the proposed algorithm is shown in figure 3.1.

The algorithm can be divided in three main steps:

1. DEM building: the disparity image is taken as input, its measures are then transformed in a 3D point cloud and arranged inside the DEM. A light obstacle detector is used to remove most of the high obstacles from the DEM;

2. low-level processing: raw curbs measures are extracted from the DEM analyzing elevation changes between adjacent cells. Raw features are then clustered in groups of candidates;

3. high-level processing: performs model fitting on the curbs candidates. Previous frame detections are then updated and used to perform temporal association on

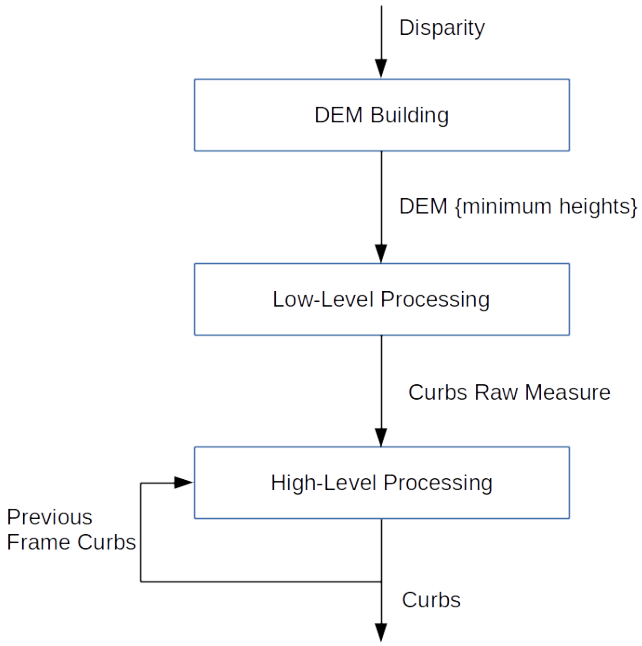


Figure 3.1: Curbs detection algorithm overview: the disparity is taken as input and it is then used to build the DEM-based representation of the 3D environment. Curbs raw features are extracted from the DEM and, successively, 3D models are extracted and temporally integrated.

the current candidates.

Algorithm output is represented by a list of the detected curbs with associated 3D elevation profile and tracking information.

3.2 Environment Model

In this work, 3D points (x_i, y_i, z_i) are expressed in the world coordinate system, which is shown in figure 3.2. This system is placed on the ground in front of the car with the

X axis pointing in the driving direction.

Figure 3.2: World reference system.

A Cartesian DEM aligned with *xy*-plane is considered, as shown in figure 3.3.

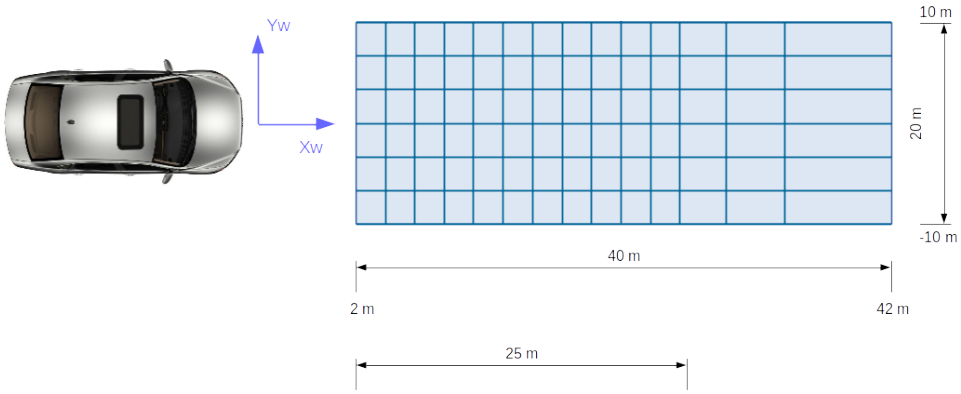


Figure 3.3: Logarithmic DEM.

Since 3D points are more and more sparse as the distance grows, the DEM is built considering logarithmically spaced cells. In this way, the density of points is kept constant regardless of the distance. Cells length is a function of the distance and is determined as follows:

$$length(x) = \begin{cases} 0.25 & x < 25.0 \\ max(\alpha * log(x), 0.25) & \text{otherwise} \end{cases} \tag{3.1}$$

The parameter α is chosen considering focal length, baseline and pitch of the stereo pair. All the cells closer than 25m share a constant length of 0.25m, this is necessary to ensure to not unify many real objects into a bigger one in closer distances.

Cell width is constant instead and fixed to 0.125m.

A DEM cell is represented by its bottom-left corner (r, c) and the height value h stored inside it. There are numerous strategies to determine the final height measurement h from all the values z_j associated to the cell: in this work minimum height is chosen for curbs in order to avoid floating objects, like trees or bridge, to overwrite the curbs.

3.3 Curb 3D Model

There are different kind of models for representing curbs shape, in this work, a polynomial model, as in [47], is chosen since it is a good trade-off between representational power and generalization. The model has to be descriptive enough in order to model different structures, such as sidewalks, roundabouts or traffic isles, but it also has to be robust to outliers due to noise artifacts.

Therefore curbs are represented as 3D curves in the world coordinate system, described by three features:

- horizontal profile: it is a cubic curve that describes the object in the xy-plane, the algebraic form is:

$$\bar{y} = f(x, \mathbf{p}) = \sum_{j=0}^{3} p_j x^j = p_0 + p_1 x + p_2 x^2 + p_3 x^3 \tag{3.2}$$

  however, since curbs can be horizontally oriented (for instance a roundabout in front of the vehicle) a second kind of polynomial is possible:

$$\bar{x} = f(y, \mathbf{p}) = \sum_{j=0}^{3} p_j y^j = p_0 + p_1 y + p_2 y^2 + p_3 y^3 \tag{3.3}$$

an example of vertical and horizontal curbs in the xy-plane is depicted in figure 3.4

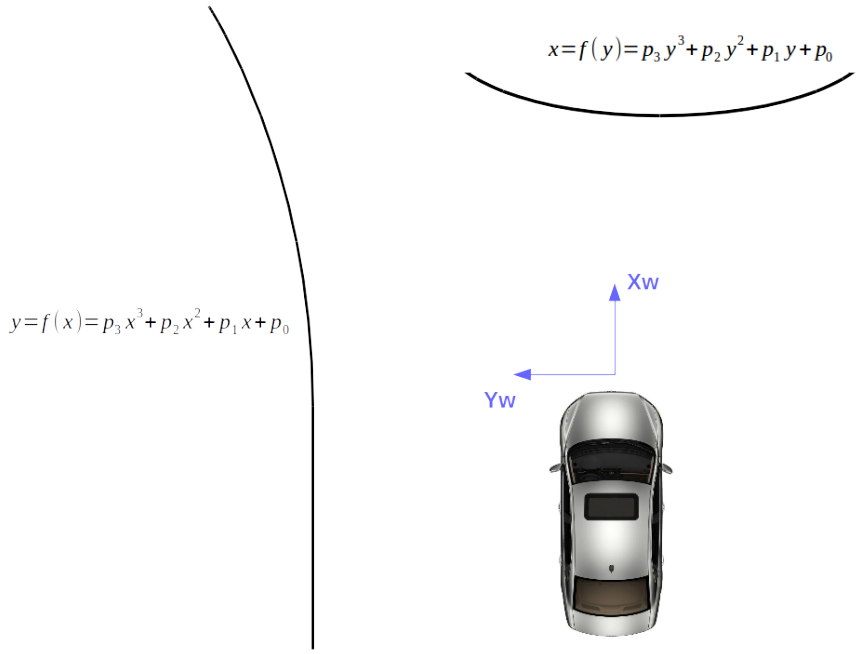


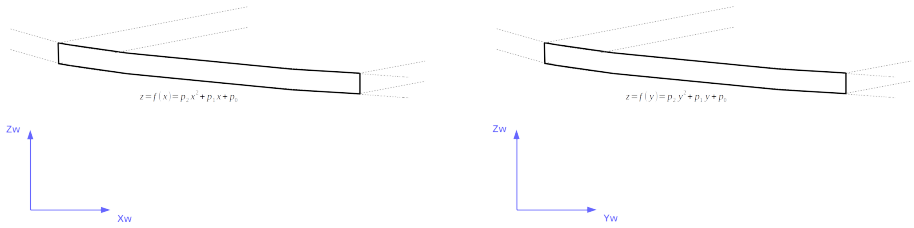
Figure 3.4: Curbs horizontal profile representation, in both vertical $\bar{y} = f(x, \mathbf{p})$ and horizontal $\bar{x} = f(y, \mathbf{p})$ cases.

- vertical profile: it is a quadratic curve that describes the object in the xz-plane, in the vertical case (figure 3.5a):

$$\bar{z} = f(x, \mathbf{p}) = \sum_{j=0}^{2} p_j x^j = p_0 + p_1 x + p_2 x^2 \tag{3.4}$$

or in the yz-plane, for horizontally oriented curbs (figure 3.5b):

$$\bar{z} = f(y, \mathbf{p}) = \sum_{j=0}^{2} p_j y^j = p_0 + p_1 y + p_2 y^2 \tag{3.5}$$



(a) Curbs vertical profile for the vertical curb case

(b) Curbs vertical profile for the horizontal curb case

Figure 3.5: Curbs vertical profile, in both vertical $\bar{z} = f(x, \mathbf{p})$ (3.5a) and horizontal $\bar{z} = f(y, \mathbf{p})$ (3.5b) cases.

- height: represents the elevation of the curb with respect to the road.

The cubic polynomial model used for the horizontal profile allows curvatures and curvature variations. Moreover, it is in accordance with the widely accepted clothoid model for lane markings. It can correctly model longitudinal curbs like sidewalks since their shape follows the boundary of the road and it also enables the detection of more complex structures such as traffic isles.

3.4 DEM building

The structure of the DEM building block is shown in figure 3.6.

The algorithm takes as input a 1358x612 disparity image. In order to save computing time, the image is then cropped to 1280x360 considering only the portion of the scene representing the road as shown in figure 3.7. The disparity is obtained from an implementation of the Semi-Global Matching algorithm [97].

Every pixel (u_j, v_j) in the rectified image, corresponding to a valid disparity value d_j, is triangulated in order to obtain its 3D coordinates (x_j, y_j, z_j).

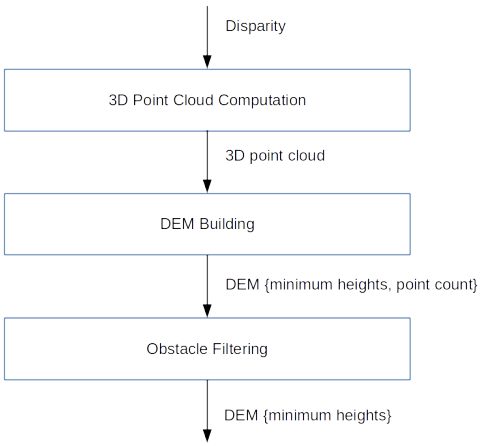


Figure 3.6: Curbs detection DEM building step.

Camera coordinates are firstly obtained with the following equations:

$$x_j^c = k_u * \frac{b}{d_j} \tag{3.6}$$

$$y_j^c = -(u_j - u_0)\frac{b}{d_j} \tag{3.7}$$

$$z_j^c = -(v_j - v_0)\frac{k_u}{k_v}\frac{b}{d_j} \tag{3.8}$$

Where (u_0, v_0) is the principal point, b is the baseline, k_u and k_v are the focal length.

World coordinates points are then obtained from the transformation mapping camera coordinates into world coordinates, ie the rotation matrix ${}^w R_c$ and the translation $(x_w, y_w, z_w)^T$:

$$\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix} = {}^w R_c \begin{pmatrix} x_j^c \\ y_j^c \\ z_j^c \end{pmatrix} + \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} \tag{3.9}$$

(a) Input image.

(b) Input DSI.

Figure 3.7: Algorithm input image and the DSI resulting of the SGM algorithm. Disparity values are encoded with colors: green for close and red for far disparity measurement.

A region of interest (ROI) of size 20mx40m in front of the vehicle is considered. All the 3D points laying outside of the ROI are discarded. 3D points having z_i coordinate higher than 2m in absolute value are discarded as well, since they are out of interest.

The 3D point is then assigned to its corresponding DEM cell (r, c). DEM cells are represented by the minimum height h of all the height values z_i associated to the cell. In addition, also the number of points is stored for every cell.

Empty cells are marked as invalid and they will not be considered in the further processing steps.

The resulting DEM is shown in figure 3.8: different elevations are represented with grayscale colors, 128 represents ground level while 255 represents cells 2m high and 0 cells -2m high. Empty cells are represented in color.

The objects in the DEM can lead to small jumps in the height values which can

Figure 3.8: Input DEM for curbs detection. Different elevations are represented with gray scale colors, 128 represents ground level while 255 represents cells 2m high and 0 cells -2m high. Empty cells are represented in color.

generate false positives. Therefore an obstacle filter module is added. This filtering is based on the idea presented in [10] where the expected road density is computed for every cell C as the area of its trapezoidal projection on the image (depicted in figure 3.9):

$$erd(C_{slope=40\%}) = C_w * C_h \tag{3.10}$$

The expected road density is computed considering a slope of 40%; the main idea is that cells containing obstacles will have a higher density of points than the road. A value of 40% for the slope is used in order to avoid uphills road to be considered as obstacles.

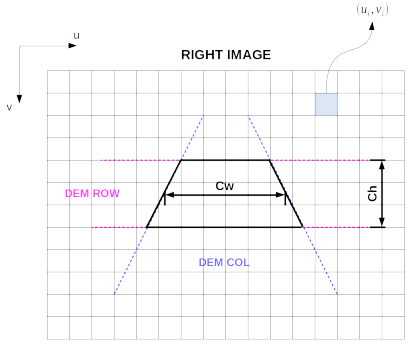


Figure 3.9: Projection of DEM cell in the right image

The classification rule for a cell C is:

$$state(C) = \begin{cases} obstacle, & \text{if } point\_num(C) > erd(C_{slope=40\%}) \\ ground, & else \end{cases} \tag{3.11}$$

The cells classified as obstacles are marked as invalid and they will not be considered in the further processing steps.

This classification is not a real obstacle detector, but it helps to have raw filtering of false positives due to other kinds of obstacles.

3.5 Low Level Processing

Curbs raw data extraction is based on the idea that curbs generate on the DEM sharply height changes in the interval between 0.05m and 0.30m. However, the noise in the image and in the 3D reconstruction can generate similar height variations, producing a considerable number of false positives and making curbs raw data extraction a challenging task.

Low-level processing steps are depicted in figure 3.10.

In order to reduce noise artifacts, a Gaussian filter on a 3x3 window is firstly applied on the DEM minimum heights. The Gaussian filter has the disadvantage of

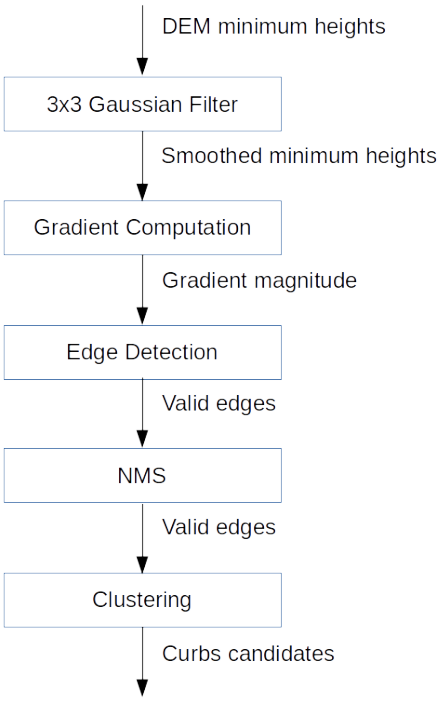


Figure 3.10: Curbs detection low-level processing

smoothing height variations, making them less sharply. Therefore a median filter would be more appropriate, however for performance reasons it is not applicable.

Since it is not possible to make hypothesis on curbs orientation, their raw features are extracted computing height gradient in both horizontal and vertical direction. The gradient $g(r, c)$ for the cell (r, c) is computed as:

$$g_x(r, c) = h(r + 1, c) - h(r - 1, c) \tag{3.12}$$

$$g_y(r, c) = h(r, c + 1) - h(r, c - 1) \tag{3.13}$$

$$g(r, c) = \sqrt{g_x(r, c)^2 + g_y(r, c)^2} \tag{3.14}$$

Only gradient magnitudes included in the interval between 0.05m and 0.30m are considered valid.

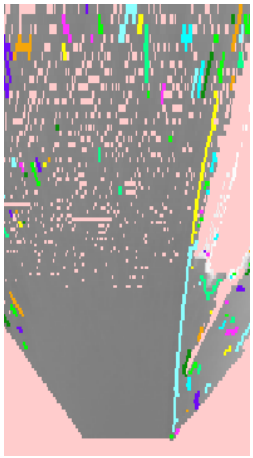
Ideally, the final result should have thin edges. Thus, non-maximum suppression (NMS) is performed to thin out curbs candidates. Some edge detectors, like Canny [28], perform the NMS step considering the direction of the gradient: i.e., the locations with gradient pointing in the x-direction are compared to the locations above and below. Since, for performance reasons, it is not possible to compute the gradient direction for every cell location, the NMS is computed in both horizontal and vertical direction. Are kept only edges that are maximum in the horizontal or vertical direction. The remaining edges represent the locations with the highest height variations. In figure 3.11 is depicted an example of detected edges.

Figure 3.11: Non-maxima suppression result where valid edges are highlighted with red.

Valid edges are then grouped with a flood-fill algorithm on an 8N neighborhood.

The final result of the low-level processing is shown on the DEM and on the input image in figure 3.12. It is possible to notice that a lot of false candidates are present,

they are principally due to the noise in the 3D reconstruction and discontinuities in the road pavement. Moreover, since the flood fill algorithm is limited to the 8N neighborhood, real curbs can be fragmented in more than one candidate. Further processing is therefore necessary.



(a) Raw curbs candidates in the DEM reference frame.

(b) Raw curbs candidates on the input image.

Figure 3.12: Curbs detection low-level output, clusters of points are highlighted in different colors.

3.6 High Level Processing

Once raw candidates are extracted a high-level processing block is executed in order to filter additional false positives and to obtain a 3D model and tracking information.

Curbs candidates, which are represented by clusters of points (r, c) in the DEM reference system, are transformed in the world coordinate system.

High-level main steps are shown on figure 3.13.

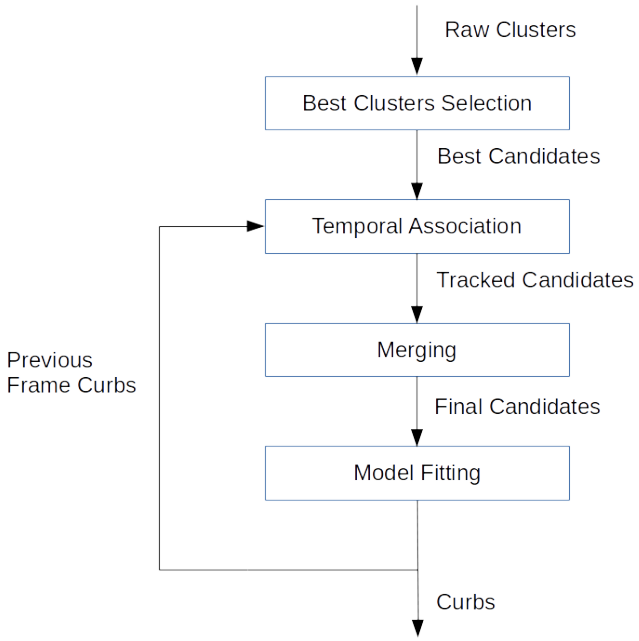


Figure 3.13: Curbs detection high-level processing.

3.6.1 Best Clusters Selection

Since the output from the low-level block is affected by a large number of false positives, it is necessary to preliminarily filter out some of them keeping only the clusters that represent the best possible candidates.

It is possible to observe that the vast majority of the false positives, due to the noise, generate little clusters characterized by a small set of 3D points while larger clusters represent real curbs. For this reason, all the small clusters are considered invalid.

Among the remaining cluster, are kept only the most relevant, considering both the size, in terms of the number of points, and their proximity to the ego-vehicle.

Since the DEM is logarithmically spaced, it is important to notice that it is possible for a cluster at a great distance to be characterized by a low number of points while representing a real curb. However, this filtering is a good trade-off since it is able to filter out the majority of the noise. Some distant curbs could be erroneously discarded, but this event is pretty rare and involves just very distant curbs.

3.6.2 Temporal Association

Past frame curbs are updated to the current one using vehicle ego-motion.

As shown in figure 3.14 a 3D point from the frame $k-1$ can be transformed in the current frame k if the translation $t_{k-1}^{k} = (tx_{k-1}^{k}, ty_{k-1}^{k}, tz_{k-1}^{k})^T$ and the rotation angle γ are known.

A point $p_i^{k-1} = (x_i^{k-1}, y_i^{k-1}, z_i^{k-1})^T$ is updated to the current reference frame $p_i^k = (x_i^k, y_i^k, z_i^k)^T$ with the transformation:

$$\begin{pmatrix} x_i^k \\ y_i^k \\ z_i^k \end{pmatrix} = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i^{k-1} \\ y_i^{k-1} \\ z_i^{k-1} \end{pmatrix} + \begin{pmatrix} tx_{k-1}^{k} \\ ty_{k-1}^{k} \\ tz_{k-1}^{k} \end{pmatrix} \tag{3.15}$$

Past detections are updated considering four 3D points from the horizontal profile model. These points are then updated to the current frame considering relation 3.15. 3D horizontal models are then recomputed in closed form from the updated points. Vertical models are not updated since the association between current candidates and past detections are made on the xy-plane only.

Now it is possible to associate current candidates with past detections. A new candidate is said to be associated with a past detection if its points overlap the updated past detection.

Past detections that are not associated with current measurement are kept updated for some frames if there are no more new associations they will be discarded.

Current measurements that are not associated with past detections will be inserted in the tracking system and will be available for the output. However, they are flagged

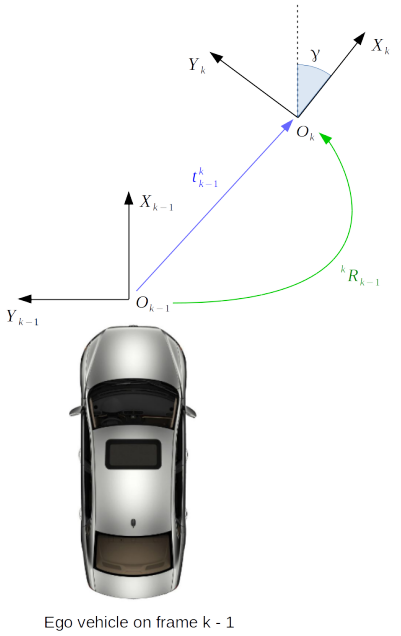


Figure 3.14: Ego motion on the xy-plane

with a particular state indicating that the curb has been detected for a small period of time.

The tracking state of past detections associated with current measures is updated.

Temporal integration helps to filter out some of the false positives due to disparity noise since it is not constant in time. Moreover tracking information can be used as an additional cue for high-level applications.

3.6.3 Merging

In this step possible merging between clusters are checked. This step is essential since the low-level clustering is limited to the 8N neighborhood and curbs can be fragmented in more than one candidate.

Tracked and non tracked candidates are iteratively analyzed to determine if a

compatible candidate exists, if it is found their points are merged and the search continues considering the new merged candidate.

Two clusters can be merged if all these three constraints are met:

1. their orientations must be compatible. Candidates orientation is computed considering the aspect ratio of their bounding box in the xy-plane, so they can be divided in three main categories: vertically-oriented, horizontally-oriented and diagonal candidates. Vertical and horizontal oriented candidates can be merged only with other candidates of their type or with diagonal candidates. This is necessary to ensure to have a selective merging, where it is not possible to merge candidates that are representing different objects;

2. their extremities must be close: the last point of the reference candidate must be close to the last point of the other candidate. Moreover, their connection angle must be small;

3. their average elevation from the ground must be similar since two clusters with different elevation change represent different objects.

If two tracked clusters are merged, they will result on a unique curb with the tracking information inherited from the older of the two.

At the end of this step, remaining untracked candidates that are too small or too distant are discarded.

The result of the merging step is shown in figure 3.15.

3.6.4 Model Fitting

In this step, 3D models are computed starting from the merged clusters.

As mentioned above there are two kinds of models: vertical and horizontal curbs. During the fitting procedure, both are computed and the best one is chosen as the final model.

For simplicity the fitting step will be described in the case of a vertical curb, the procedure for horizontal curbs is entirely similar.

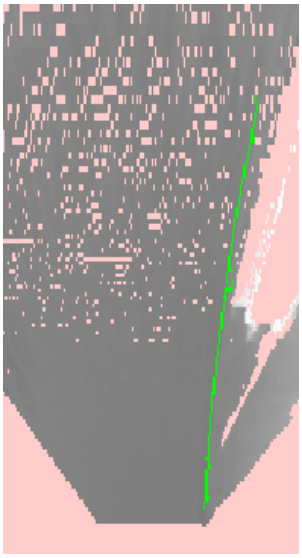


Figure 3.15: Merged curbs candidates

3.6.4.1 Horizontal Profile

The model fitting step for the horizontal profile is done by solving the system of n equations and 4 unknowns, where n is the number of points (x_i, y_i) of the candidate and the 4 unknowns are the coefficients p_0, p_1, p_2 and p_3 of the cubic curve. For vertical curbs the equations are in the form:

$$p_3 x_i^3 + p_2 x_i^2 + p_1 x_i + p_0 = y_i, i = 1..n \tag{3.16}$$

And the system can be written as a matrix equation:

$$AX = B \tag{3.17}$$

with:

$$A = \begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^3 & x_n^2 & x_n & 1 \end{bmatrix}, X = \begin{bmatrix} p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix}, B = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{3.18}$$

Since the number of points is above 4 the system in (3.17) is overdetermined, and it is solved in a least square fashion.

It is possible to have noise artifacts in the clusters and it is not reasonable to assume that every observation should be treated equally. A weighted fitting is done by assigning each data point to its deserved degree of importance in the parameter estimation process by including weight factors.

So the error function is the sum of the weighted quadratic errors:

$$E(w, \mathbf{p}) = \frac{1}{2} \sum_{i=1}^{N} w_i (\bar{y}(x_i, \mathbf{p}) - y_i)^2 \tag{3.19}$$

where w_i are the weights associated to the point (x_i, y_i).

By replacing 3.16 in 3.19 the cost function became:

$$E(w, \mathbf{p}) = \frac{1}{2} \sum_{i=1}^{N} w_i (p_3 x_i^3 + p_2 x_i^2 + p_1 x_i + p_0 - y_i)^2 \tag{3.20}$$

For $E(w, \mathbf{p})$ to have a minimum value, its partial derivatives with respect to the unknowns parameters p_i must be 0. The following system of equations must be solved:

$$\begin{cases} \frac{\partial E(w,\mathbf{p})}{\partial p_3} = 0 \\ \frac{\partial E(w,\mathbf{p})}{\partial p_2} = 0 \\ \frac{\partial E(w,\mathbf{p})}{\partial p_1} = 0 \\ \frac{\partial E(w,\mathbf{p})}{\partial p_0} = 0 \end{cases} \tag{3.21}$$

After writing explicitly each equation, the system in 3.21 becomes:

$$\begin{bmatrix} \sum_{i=1}^{n} w_i x_i^6 & \sum_{i=1}^{n} w_i x_i^5 & \sum_{i=1}^{n} w_i x_i^4 & \sum_{i=1}^{n} w_i x_i^3 \\ \sum_{i=1}^{n} w_i x_i^5 & \sum_{i=1}^{n} w_i x_i^4 & \sum_{i=1}^{n} w_i x_i^3 & \sum_{i=1}^{n} w_i x_i^2 \\ \sum_{i=1}^{n} w_i x_i^4 & \sum_{i=1}^{n} w_i x_i^3 & \sum_{i=1}^{n} w_i x_i^2 & \sum_{i=1}^{n} w_i x_i \\ \sum_{i=1}^{n} w_i x_i^3 & \sum_{i=1}^{n} w_i x_i^2 & \sum_{i=1}^{n} w_i x_i & \sum_{i=1}^{n} w_i \end{bmatrix} \begin{bmatrix} p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} w_i x_i^3 y_i \\ \sum_{i=1}^{n} w_i x_i^2 y_i \\ \sum_{i=1}^{n} w_i x_i y_i \\ \sum_{i=1}^{n} w_i y_i \end{bmatrix} \tag{3.22}$$

System 3.22 has 4 linear equation and 4 unknowns, therefore solving it is a trivial algebra problem.

The weights are initialized considering the orientation of gradient, the shape of the cluster and the elevation change:

1. firstly the gradient orientation of every point in the cluster is computed, if it is concordant with the cluster main orientation the point will have a high weight; otherwise the weight will be low:

$$w_i^o = \begin{cases} 1.0 & \textit{if orientations are the same} \\ 0.5 & \textit{one of the two is diagonal} \\ 0.0 & \textit{otherwise} \end{cases} \tag{3.23}$$

2. secondly the elevation change of every point is considered: if its value is close to the cluster average elevation change an extra factor is added to the weight:

$$w_i^e = \begin{cases} 1.0 & |z_i - avg\_z| < th \\ 0.0 & \textit{otherwise} \end{cases} \tag{3.24}$$

   Where th is the threshold defining the maximum difference between the elevation of the point and the average elevation of the cluster;

3. for vertically-oriented clusters the histogram of the distribution of the y_i coordinates is computed as in figure. The argmax y_{max} is then found and the third component of the weight is computed as:

$$w_i^d = \frac{1}{1 + |y_{max} - y_i|} \tag{3.25}$$

The same procedure is done for horizontally-oriented clusters considering the distribution of x_i coordinates instead. For diagonal clusters, the weight is fixed to 1.0.

The normalized initialization value became:

$$w_i^0 = \frac{w_i^o + w_i^e + w_i^d}{3} \tag{3.26}$$

The weighted least squares is iterated 3 times. Weights are then updated after every iteration considering the distance from the computed model. At the iteration n the weight for the point i is computed as:

$$w_i^n = \frac{w_i^{n-1}(3+n) + \frac{1}{1+e_i}}{4+n} \tag{3.27}$$

Where e_i is the error for the point i:

$$e_i = |y_i - p_3 * x_i * 3 - p_2 * x_i * 2 - p_1 * x_i - p_0| \tag{3.28}$$

The model is considered valid only if it has enough inliers. If the model is not valid, the cluster is rejected.

3.6.4.2 Vertical Profile

Similarly, the vertical profile is computed in a least square fashion from the set of n points (x_i, z_i):

$$p_2 x_i^2 + p_1 x_i + p_0 = z_i, i = 1..n \tag{3.29}$$

Only the inliers of the horizontal profile are considered in this step.
The error function is:

$$E(\mathbf{p}) = \frac{1}{2}\sum_{i=1}^{N}(\bar{z}(x_i, \mathbf{p}) - z_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(p_2 x_i^2 + p_1 x_i + p_0 - z_i)^2 \tag{3.30}$$

The minimum value of $E(\mathbf{p})$ is computed solving the following system of equations:

$$\begin{cases} \frac{\partial E(\mathbf{p})}{\partial p_2} = 0 \\ \frac{\partial E(\mathbf{p})}{\partial p_1} = 0 \\ \frac{\partial E(\mathbf{p})}{\partial p_0} = 0 \end{cases} \tag{3.31}$$

Which can be written explicitly in the system:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i & n \end{bmatrix} \begin{bmatrix} p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix} \tag{3.32}$$

For computational reasons weights are not considered in this step, and the fitting is iterated just one time. Moreover, the horizontal profile is much more noisy with respect to the vertical one. Once points are cleared from the first fitting is not necessary to have an iterative procedure.

The vertical profile is then validated considering its inliers number: if they are not enough, the cluster is discarded.

3.6.4.3 Height

Once the horizontal and vertical profiles are computed curbs height from the ground is extracted.

The height is computed as the average value of the DEM gradient magnitude corresponding to the curb inliers.

The final result of the detection is shown in figure 3.16.

Figure 3.16: Algorithm final result: the detected curb is reprojected back onto the right image

3.7 Experimental Results

In this section, the validity of the proposed method will be demonstrated with extensive qualitative and quantitative analysis.

The system was implemented for an embedded platform and it was able to run on it in less than 5ms, therefore well below the secure threshold of 30fps.

Three-dimensional data was acquired with a calibrated stereo rig mounted on the roof of the vehicle and pointing to the driving direction.

3.7.1 Qualitative Results

Before proceeding with the quantitative results, a qualitative analysis will be presented.

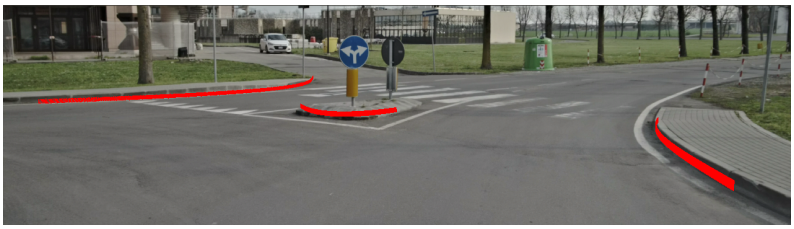
The proposed approach was tested on a demonstrator vehicle in various traffic scenarios. Tests covered as much conditions as possible, from highway to urban scenario and including different road geometries, some examples of detection results are shown in figure 3.17.

In all the scenarios curbs were robustly detected and tracked: results were evaluated by inspection of human experts. Moreover, curbs were used in a test vehicle as input for path planning and localization demonstrating to be able to produce robust results that can be used for high-level applications.

The detection has a maximum working range of about 40m, however, in order to be seen curbs lower extremity must be closer than 25m. This is well above other

(a) traffic conditions

(b) curved roads



(c) traffic isle

(d) horizontal curbs

Figure 3.17: Qualitative results, for various scene and road geometry, of the proposed stereovision-based method for curbs detection.

stereo vison based algorithms that work at most to 20m.

The chosen model demonstrated to be descriptive enough in order to model the

vast majority of the situations, however, as shown in figure 3.18, there are some cases where road curbs geometry is not accurately reconstructed. These situations are nonetheless acceptable since they tend to be rare and the detection is still present. A more sophisticated model would represent these cases more accurately; however, it would be more sensitive to noise outliers with the risk of fitting too noisy and too unstable curbs candidates.

Figure 3.18: Example of curb which cannot be modeled with the chosen polynomial model.

3.7.2 Quantitative Results

Since curbs can be manually annotated on the image, a quantitative benchmark procedure will be described.

3.7.2.1 Benchmark procedure

The proposed method was evaluated on two recordings collected with our test vehicle in different traffic scenario and, also on a subset of the KITTI odometry dataset [3].

Regarding the first two recordings, curbs were manually annotated with polylines defining the border between road and curbs: the labeled data describes how the optimal curbs detection algorithm should be. Since the algorithm was used for ego-lane estimation and localization, were annotated all curbs with a starting point closer than approximately 20-25 meters and all curbs belonging to the ego-lane or its adjacent lanes. Short segments of curbs (with length less than 1m) were annotated only if they were part of a longer curb that could be seen previously. Too distant curbs were not

annotated since they do not belong to the system requirements. In figure 3.19 are shown some examples of labeled ground truth.

Figure 3.19: Examples of curbs ground truth

Regarding the KITTI dataset, the annotation procedure is the same as before however, since KITTI dataset includes scenarios with low-height curbs, only the ones

with an elevation that is compatible with the proposed method were annotated (i.e., an elevation included in the range between 5cm and 35cm). Moreover, curbs in strong saturation and shadows scenarios were not annotated, and the required minimum detection distance was reduced to approximately 10 meters considering the different stereo-rig setup. Some example of annotations in the KITTI dataset are shown in figure 3.20.

(a) curbs with too low elevation

(b) example of curbs annotated at two sides of the road

(c) example of annotation with a distant curb

Figure 3.20: Examples of curbs ground truth of the KITTI visual odometry dataset.

3D detections were reprojected on the image space and associated with ground truth annotations. An annotation was said to be associated if a detection overlapped it. An annotation was considered a true positive if it was associated with at least one detection; otherwise, it was considered a false negative. Detections that were not

associated with annotations were considered as false positives.

Two measures were provided:

- recall or sensitivity, computed as:

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.33}$$

- precision, computed as:

$$Precision = \frac{TP}{TP + FP} \tag{3.34}$$

3.7.2.2 Benchmark results

Two different recordings acquired with our test vehicle were evaluated:

- the first one was recorded in Las Vegas. The sequence was captured in a traffic scenario with normal lighting conditions;
- the second recording was collected in Santa Clara (CA) in the late afternoon after a thunderstorm. The wet asphalt and low sun on the horizon created a very challenging test for curbs detection.

Both recordings have a total of 9000 frames, captured at 30fps. Annotations were at 1 fps; therefore for every sequence, there are 300 annotated frames.

A summary of the evaluation is shown in table 3.1.

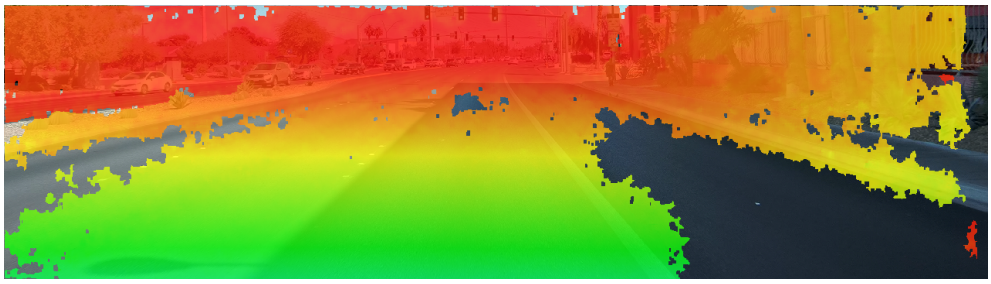
Recording	# annotated frames	# total curbs	Sensitivity	Precision
Las Vegas	300	271	90.7%	72.5%
Santa Clara	300	349	66.2%	75.4%
Overall	600	620	78.45%	73.95%

Table 3.1: Curbs detection results evaluated on two different datasets.

It is possible to notice that the algorithm worked robustly in the first evaluated recording: since lighting conditions were good, the 3D environment was correctly reconstructed, and curbs were detected in the 90% of the cases.

Missed curbs were usually due to the presence of poor illumination, shadows or reflective surfaces which made the quality of the 3D reconstruction very low. In these situations, the detection was challenging since the 3D data on the curb and on its neighborhood was not dense enough. An emblematic case is shown in figure 3.21. This effect was even more strong for curbs belonging to a side lane, like the one in figure 3.21.

(a) Input image



(b) DSI

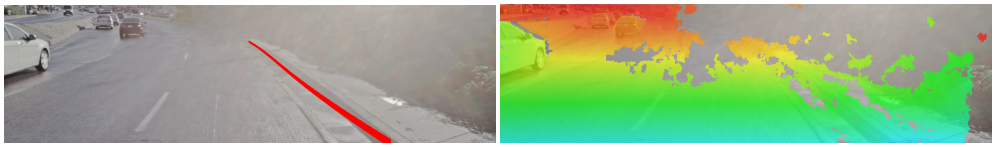
Figure 3.21: An example of difficult scenario for stereovision-based curbs detection where the quality of the 3D reconstruction tends to be very low due to the presence of shadows.

Most of the false positives, in the first recording, were due to the presence of multiple levels of curbs. This kind of false positives did not represent a problem since they were not placed on the road and since they were also occluded by other curbs. They could be removed, for instance with a ray tracing algorithm on the DEM as explained in [47]. In this work occluded curbs are not removed since occlusion

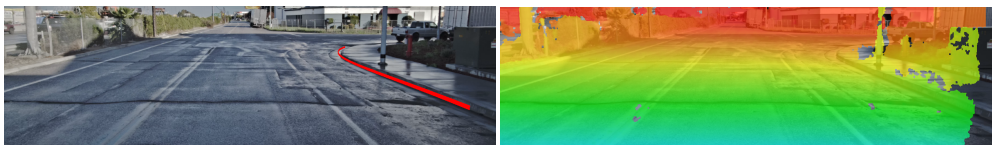
removal algorithms, such as the one in [47], are computationally expensive and these kinds of false do not represent an issue for the system.

Regarding the second recording, it consists in a challenging scenario since the presence of wet asphalt and poor lighting conditions the three-dimensional reconstruction quality became very low. In this case, the precision fell at 66% due to the lack of detections in dazzle situations or when the asphalt was so wet to become a perfect mirror.

In these conditions, even when curbs were detected, as in figure 3.22 they resulted in a very unstable and noisy detection. Also, the detection range tended to be lower (approximately 20m) than in normal situations.



(a) Dazzle.



(b) Wet asphalt.

(c) Both dazzle and wet asphalt, with poor image quality.

Figure 3.22: An example of detection in presence of wet asphalt and poor lighting conditions.

In these scenarios dangerous false positives, as the ones shown in figure 3.23, are possible. This particular kind of false positives should not happen since they are in the middle of the road they could disturb both path-planning and localization.

These considerations suggest that further processing is required in order to man-

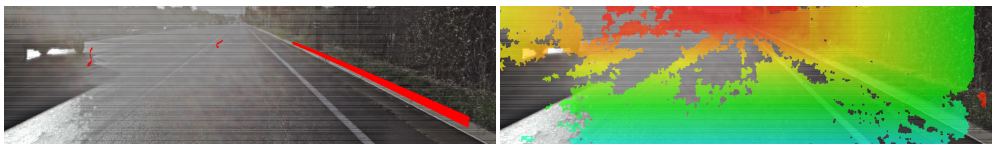


Figure 3.23: False curbs on the road, due to poor lighting and image quality.

age scenarios with poor three-dimensional reconstruction. In order to improve the performance in these cases, something can be done by correcting the image quality for cases where the lighting conditions are very poor.

Direct comparisons with other algorithms are not possible since there are not shared datasets and benchmarks with left and right pairs and since most of the works in literature propose only qualitative analysis. Some works, like for instance [41] and [48], present quantitative results without a detailed description of the dataset they used.

For these reasons, the proposed method was additionally evaluated on the KITTI dataset, since it is a severe and commonly used dataset which could be useful for further comparisons. In particular, the visual odometry dataset was chosen since it is used by other works on curbs detection in literature [34, 35] and since it includes images with strong saturation and shadows, there is a considerable number of misshapen roads and, most importantly, the vast majority of curbs have a low elevation from the ground which makes the detection particularly difficult for methods which model curbs as a sharp elevation change, like the one presented in this work.

Similar to [34], three recordings from the dataset were used: odometry03, odometry06, odometry15. The sequences are all captured in a suburban area with straight and curved roads together with static and dynamic obstacles. Since KITTI sequences are recorded at 10 frames per second, a frame every 10 was annotated as previously described.

Results are reported in table 3.2. The work presented in [34] reports an average true positive rate of 81% and a false positive rate of 3.2%, they are both referred as the ratio of the total length of the detected curbs and the total length of the curbs in a region covering 3m to either side of the vehicle. In [35] results are reported as

pixel-level metrics: the setup with the best performance achieves 89.73% in terms of $F1$-measure.

Recording	# annotated frames	# total curbs	Sensitivity	Precision
odometry03	80	111	73.9%	87.2%
odometry06	110	134	94.0%	88.7%
odometry15	190	184	72.8%	66.6%
Overall	380	429	80.23%	80.83%

Table 3.2: Curbs detection results evaluated on the KITTI visual odometry dataset [3].

The algorithm performs well in the odometry06 recording: in this scenario, curbs represents a sufficiently sharp elevation change, as shown in figure 3.24, that can be robustly detected in all sides of the road.

Figure 3.24: Examples of robust and stable detections on the dataset odometry06 from KITTI.

In odometry03 and odometry15 recordings, a great number of low curbs is present (some examples in figure 3.25). These curbs are at the border between detectable and non-detectable objects for the proposed method, errors and noise in the 3D reconstruction make the detection particularly challenging, especially at greater distances. In order to solve this issue the range of valid elevations could be extended to detected 4 cm high curbs also, however, the 3D reconstruction must have very high quality and, even in this case, the risk of having unstable and noisy detections is real.

In figure 3.26 are shown other challenging situation in the presence of extreme saturation and shadows, as previously discussed.

Figure 3.25: Examples of low curbs representing a challenging scenarios for the proposed method.

Figure 3.26: Examples of images with shadows and strong saturation.

Shadows and saturation are also responsible for the detection of false curbs on the road surface, which represents the vast majority of false positives in odometry15 (some examples are shown figure 3.27).

Figure 3.27: Examples of images with false positives curbs caused by shadows and saturation.

Other false positives are due to the presence of low height brushes which could generate elevation changes that are compatible with the presence of curbs, as shown in figure 3.28.

These considerations suggest that color intensity cues could be used together with 3D information in order to give to the system additional robustness, even in case of low-height curbs, vegetation or poor light conditions, at the cost of higher computational demand.

Figure 3.28: Examples of images with false positives caused by the presence of low height vegetation.

Chapter 4

Barriers Detection

This chapter provides a detailed description of the stereo vision based barriers detection algorithm.

This work is developed with the purpose of detecting generic vertical structures defining road boundaries as well as a possible driving corridor. Structures like: guardrails, fences, hedges, walls, and lines of standing or parked vehicles fall into this category. Since most of the state of the art approaches are focused on specific structures, this algorithm is designed to be generic enough in order to detect every kind of barrier regardless of their aspect and it should also be able to produce robust results that can be used for ego-lane estimation or obstacle avoidance.

The algorithm is designed in order to be efficient in terms of computation time and memory footprint, and it is also designed to share some of the processing with the curbs detection algorithm described on Chapter 3. Barriers detector runs in real-time on an embedded platform, sharing computational resources with other algorithms in order to realize a highly autonomous driving system.

This chapter is organized as follows. In Section 4.1 will be presented an overview of the proposed algorithm. In Section 4.2 the environment model will be described. The model used to describe barriers in the three-dimensional world is described in Section 4.3. In Sections 4.4, 4.5 and 4.6 algorithm steps will be presented. In Section 4.7 experimental results will be analyzed.

4.1 Algorithm Overview

Since the term barrier includes a wide variety of objects, it is not possible to make assumptions regarding their appearance. For this reason, the algorithm is based on the detection of sharply elevation changes from the road surface, since it is the only feature that characterizes the entire set of structures.

A three-dimensional reconstruction of the scene is therefore necessary. The stereo camera is chosen as the input sensor: it enables a dense reconstruction of the 3D environment even at greater distances, it can be easily integrated on a vehicle, and it is also shared between the several perception algorithms composing the autonomous driving system.

This algorithm is also designed to share some of the processing with the curbs detection algorithm described in Chapter 3, this would result in an important computation time-saving. In particular, it inherits the processing on the logarithmic DEM which is used to represent the 3D environment. The DEM is chosen for the 3D space representation because it enables real-time processing and because it requires little memory.

An overview of the proposed algorithm is shown in figure 4.1.

The algorithm is divided into three main blocks:

1. DEM building: disparity image is taken as input, image points are then triangulated in order to obtain the corresponding 3D coordinates and arranged inside the DEM;

2. low-level processing: in this step, raw features are extracted from the DEM analyzing height variations between connected cells. A clustering algorithm is then applied to group measurements in sets of raw candidates;

3. high-level processing: raw candidates are associated with previous frame detections, a model representing barriers in the 3D world is then fitted to the candidates.

The algorithm output consists on the detected barriers described by their position and shape in the 3D environment and their tracking information.

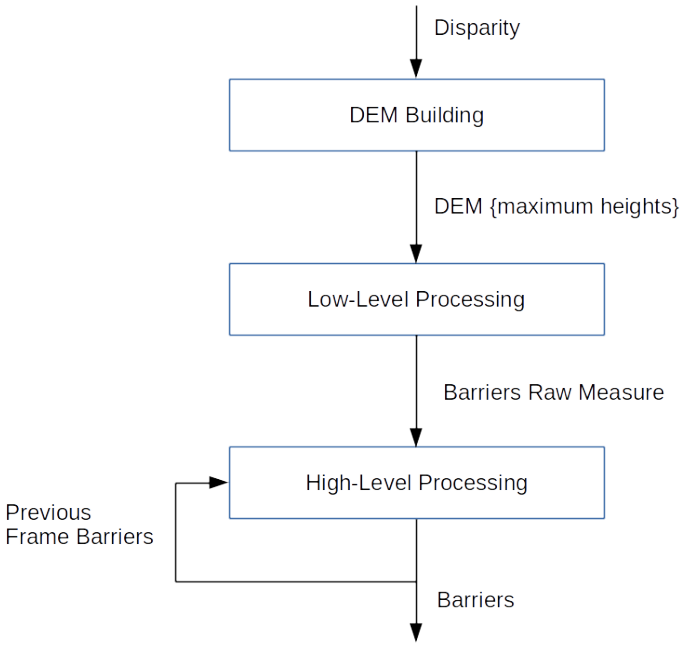


Figure 4.1: Barriers detection algorithm overview.

4.2 Environment Model

The reference frame used to express 3D points (x_j, y_j, z_j) is the world coordinate system. This system is placed in front of the vehicle with the X-axis pointing in the driving direction; a more detailed description can be found on Section 3.2.

A logarithmic Cartesian DEM, like the one described in Section 3.2, is used for representing the 3D space. In this case, the size of the grid is 20mx80m. Since the density of the 3D points becomes lower and lower with the distance, cells length is increased logarithmically:

$$length(x) = \begin{cases} 0.25 & x < 25.0 \\ max(\alpha * log(x), 0.25) & \text{otherwise} \end{cases} \tag{4.1}$$

Where α is a parameter chosen considering stereo camera baseline, pitch, and focal length. In the first 25 meters, cells length is constant to 0.25m in order to not merge different objects at closer distances.

The width of the cells is constant and fixed. Typical values are 0.125m and 0.25m.

The cells composing the DEM are represented by their bottom-left corner (r, c) and the height value h stored inside them. In every cell is memorized the maximum height from all values z_j associated to the cell.

4.3 Barriers 3D Model

The model used to describe barriers in the 3D environment is the same used for curbs and described in Section 3.3. This model has demonstrated to be robust enough to noise outliers, while it can describe most of the structures appropriately since it is in accordance to widely accepted clothoid model used for lane detection.

Barriers are represented as 3D curves in the world coordinate system, described by three features:

- horizontal profile: it is a cubic curve that describes the object in the xy-plane, the algebraic form is:

$$\bar{y} = f(x, \mathbf{p}) = \sum_{j=0}^{3} p_j x^j = p_0 + p_1 x + p_2 x^2 + p_3 x^3 \tag{4.2}$$

  however, barriers can be horizontally oriented (for instance a wall in front of the vehicle) a second kind of polynomial is possible:

$$\bar{x} = f(y, \mathbf{p}) = \sum_{j=0}^{3} p_j y^j = p_0 + p_1 y + p_2 y^2 + p_3 y^3 \tag{4.3}$$

- vertical profile: it is a quadratic curve that describes the object in the xz-plane, in the vertical case:

$$\bar{z} = f(x, \mathbf{p}) = \sum_{j=0}^{2} p_j x^j = p_0 + p_1 x + p_2 x^2 \tag{4.4}$$

or in the yz-plane, for horizontally oriented barriers:

$$\bar{z} = f(y, \mathbf{p}) = \sum_{j=0}^{2} p_j y^j = p_0 + p_1 y + p_2 y^2 \tag{4.5}$$

- height: represents the barrier elevation from the road surfaces.

A more detailed description can be found in Section 3.3.

4.4 DEM building

The DEM building block is shown in figure 4.2.

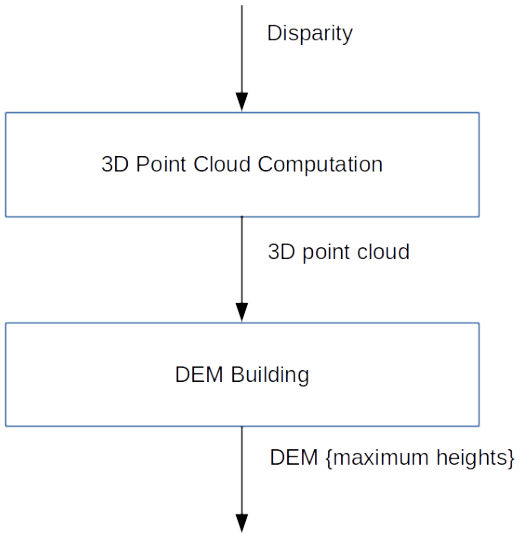


Figure 4.2: Barriers detection DEM building step.

The algorithm input is a 1358x612 disparity image, computed with an implementation of the Semi-Global Matching algorithm [97]. The disparity is cropped to a 1280x360 ROI including only the image portion representing the road. An example of an input image and the corresponding disparity is shown in figure 4.3.

(a) Input image.

(b) Input DSI.

Figure 4.3: Barriers detection algorithm input image and the DSI resulting from the SGM algorithm. Disparity values are encoded with colors: green for close and red for far disparity measurement.

The pixels (u_j, v_j) in the rectified image, corresponding to valid disparity values d_j are triangulated to obtain their 3D coordinates (x_j, y_j, z_j) in the world reference frame. Camera coordinates are obtained with the following equations:

$$x_j^c = k_u * \frac{b}{d_j} \tag{4.6}$$

$$y_j^c = -(u_j - u_0)\frac{b}{d_j} \tag{4.7}$$

$$z_j^c = -(v_j - v_0)\frac{k_u}{k_v}\frac{b}{d_j} \tag{4.8}$$

Where (u_0, v_0) is the principal point, b is the baseline, k_u and k_v are the focal length.

World coordinates points are then obtained from the rotation matrix ${}^w R_c$ and the translation $(x_w, y_w, z_w)^T$ mapping camera into world coordinates:

$$\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix} = {}^w R_c \begin{pmatrix} x_j^c \\ y_j^c \\ z_j^c \end{pmatrix} + \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} \tag{4.9}$$

The algorithm considers only the 3D points falling inside a region of interest (ROI) of size 20mx80m in front of the vehicle. 3D points having z_i coordinate higher than 2m are discarded since they are out of interest.

World points are then assigned to their corresponding DEM cell (r, c), which are represented by the maximum height h of all the z_i associated to the cell.

Empty cells are marked as invalid, and they will not be considered in the further processing steps.

The resulting DEM is shown in figure 4.4: gray-scale colors are used to represent different elevations, 128 indicates ground level while 255 represents cells 2m high and 0 cells -2m high. Empty cells, instead, are represented in color.

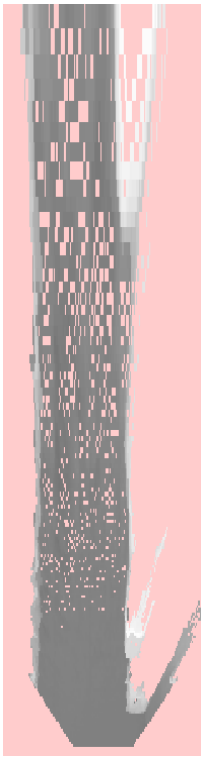


Figure 4.4: Input DEM example for barriers detection. Different elevations are represented with gray scale colors, 128 represent ground level while 255 represent cells 2m high and 0 cells -2m high. Empty cells are represented in color.

4.5 Low Level Processing

Once the DEM is built, the low-level processing block is executed. The basic idea behind the extraction of barriers raw features is that they generate on the DEM sharply height changes.

Figure 4.5 describes low-level processing steps.

Since barriers must be detected independently of their orientation, raw features are extracted computing height derivatives in both vertical and horizontal direction.

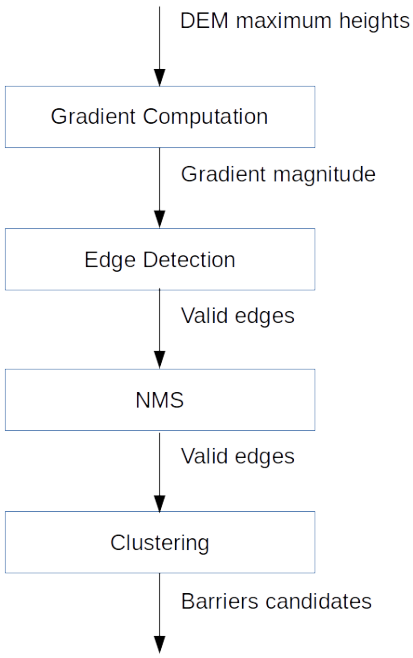


Figure 4.5: Barriers detection low-level processing.

The gradient $g(r, c)$ in the cell (r, c) is computed as:

$$g_x(r, c) = h(r + 1, c) - h(r - 1, c) \tag{4.10}$$

$$g_y(r, c) = h(r, c + 1) - h(r, c - 1) \tag{4.11}$$

$$g(r, c) = \sqrt{g_x(r, c)^2 + g_y(r, c)^2} \tag{4.12}$$

Only gradient magnitudes included in the interval between 0.30m and 2.0m are considered compatible for barriers detection.

Non-maximum suppression (NMS) is performed on the resulting edges to obtain tiny candidates. Are kept only edges that are maximum in the vertical or horizontal

direction. The resulting edges represent the locations with highest height variations. An example of detected edges, for the input image 4.3a, is shown in figure 4.6.

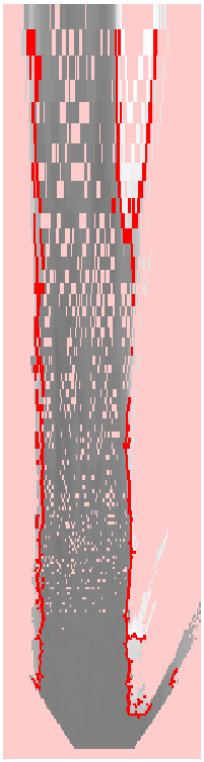


Figure 4.6: Non-maxima suppression result.

A flood-fill algorithm on an 8N neighborhood is used to group valid edges in clusters of candidates.

The final result of the low-level processing is shown in figure 4.7.

4.6 High Level Processing

High-level processing step is feed with the raw candidates extracted by the low-level block. In this step, additional false positives are filtered, and 3D models and tracking information are obtained.

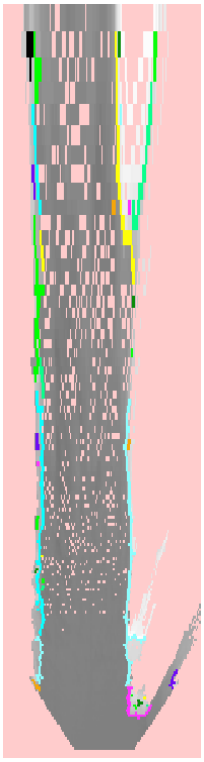


Figure 4.7: Barriers raw candidates.

Barriers raw candidates, which are represented by clusters of points on the DEM, are firstly transformed back in the world reference system.

High-level main steps are shown on figure 4.8.

4.6.1 Best Clusters Selection

In this step, smaller and distant clusters are filtered in order to remove some of the false positives keeping only the most promising clusters.

Since most of the noise is due to little clusters with a small set of points, all the smaller clusters are firstly rejected. Moreover, small candidates are not suitable for representing barriers, since they are typically very long and continuous structures.

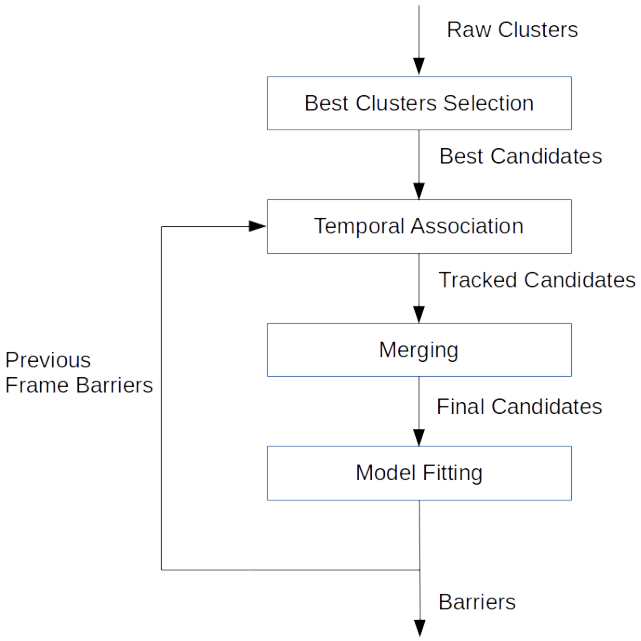


Figure 4.8: Barriers detection high-level processing.

Only the most relevant clusters, in terms of points number and proximity to the ego-vehicle, are kept for further processing. However, at great distances, it is possible that clusters characterized by a low number of points to represent a real barrier. This is due to the fact that DEM cells are logarithmically spaced: closer cells are shorter than more distant ones. Some faraway barriers could be erroneously discarded, but this event is rare and involves just very distant barriers, therefore this filtering is a good trade-off since it can filter out the vast majority of the noise.

4.6.2 Temporal Association

Vehicle-ego motion data is used to update past frame barriers to the current frame.

As described in Section 3.6.2 a 3D point from the frame $k-1$ can be transformed in

the current frame k with the translation $t_{k-1}^{k} = (tx_{k-1}^{k}, ty_{k-1}^{k}, tz_{k-1}^{k})^T$ and the rotation angle γ:

$$\begin{pmatrix} x_i^k \\ y_i^k \\ z_i^k \end{pmatrix} = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i^{k-1} \\ y_i^{k-1} \\ z_i^{k-1} \end{pmatrix} + \begin{pmatrix} tx_{k-1}^{k} \\ ty_{k-1}^{k} \\ tz_{k-1}^{k} \end{pmatrix} \tag{4.13}$$

Current candidates are now associated with past detection: a new candidate is said to be associated with a past detection if its points overlap the updated past detection. The tracking state of past detections associated with current measures is updated.

New measurements that are not associated with past detections will be inserted in the tracking system and will be available for the output, but marked with a particular state.

Past detections that are no more associated with current measurement are kept updated for some frames, if there are no more new associations they will be discarded.

4.6.3 Merging

Low-level clustering is limited to the 8N neighborhood and a single barrier can be fragmented in more than one candidate, therefore, in this step, possible mergings are checked.

Both tracked and non-tracked candidates are iteratively analyzed to determine if a compatible candidate exists, and if it is found their points are merged and the search continues considering the new merged candidate.

Two clusters can be merged if these two constraints are met:

1. their extremities must be close: the last point of the reference candidate must be close to the last point of the other candidate. Moreover, their connection angle must be small;

2. their average elevation from the ground must be similar since two clusters with different elevation change could represent different objects.

When two tracked candidates are merged, the resulting unique barrier will inherit tracking information from the older of the two.

At the end of this step, remaining untracked candidates that are too small or distant are discarded.

In figure 4.9 are shown merging process results.

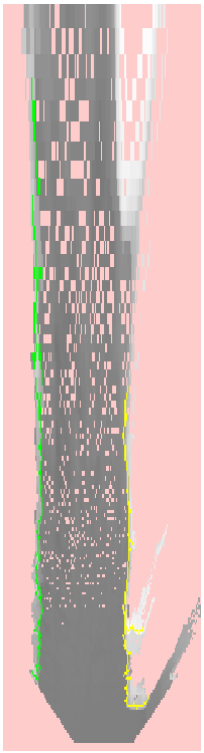


Figure 4.9: Merged barriers candidates.

4.6.4 Model Fitting

In this step merged clusters are used to compute 3D models.

As mentioned above there are two kinds of models: vertical and horizontal curbs. During the fitting procedure, both are computed and the best one is chosen as the final model. For simplicity, the fitting step will be described in the case of a vertical barrier, the procedure for horizontal one is completely similar.

4.6.4.1 Horizontal Profile

The model fitting step for the horizontal profile is done by solving the system of n equations and 4 unknowns, where n is the number of points (x_i, y_i) of the candidate and the 4 unknowns are the coefficients p_0, p_1, p_2 and p_3 of the cubic curve. For vertical barriers the equations are in the form:

$$p_3 x_i^3 + p_2 x_i^2 + p_1 x_i + p_0 = y_i, i = 1..n \tag{4.14}$$

And the system can be written as a matrix equation:

$$AX = B \tag{4.15}$$

with:

$$A = \begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^3 & x_n^2 & x_n & 1 \end{bmatrix}, X = \begin{bmatrix} p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix}, B = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{4.16}$$

Since the number of points is above 4 the system in (4.15) is overdetermined, and it is solved in a least square fashion.

So the error function is the sum of the quadratic errors:

$$E(w, \mathbf{p}) = \frac{1}{2} \sum_{i=1}^{N} (\bar{y}(x_i, \mathbf{p}) - y_i)^2 \tag{4.17}$$

By replacing 4.14 in 4.17 the cost function became:

$$E(w, \mathbf{p}) = \frac{1}{2} \sum_{i=1}^{N} (p_3 x_i^3 + p_2 x_i^2 + p_1 x_i + p_0 - y_i)^2 \tag{4.18}$$

For $E(\mathbf{p})$ to have a minimum value, its partial derivatives with respect to the unknowns parameters p_i must be 0. The following system of equations must be solved:

$$\begin{cases} \frac{\partial E(w,\mathbf{p})}{\partial p_3} = 0 \\ \frac{\partial E(w,\mathbf{p})}{\partial p_2} = 0 \\ \frac{\partial E(w,\mathbf{p})}{\partial p_1} = 0 \\ \frac{\partial E(w,\mathbf{p})}{\partial p_0} = 0 \end{cases} \tag{4.19}$$

After writing explicitly each equation, the system in 4.19 becomes:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^6 & \sum_{i=1}^{n} x_i^5 & \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^5 & \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i & n \end{bmatrix} \begin{bmatrix} p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i^3 y_i \\ \sum_{i=1}^{n} x_i^2 y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix} \tag{4.20}$$

System 4.20 has 4 linear equation and 4 unknowns, therefore solving it is a trivial algebra problem.

The model is considered valid only if it has enough inliers. If the model is not valid, the cluster is rejected.

4.6.4.2 Vertical Profile

Similarly, the vertical profile is computed in a least square fashion from the set of n points (x_i, z_i):

$$p_2 x_i^2 + p_1 x_i + p_0 = z_i, i = 1..n \tag{4.21}$$

Only the inliers of the horizontal profile are considered in this step.

The error function is:

$$E(\mathbf{p}) = \frac{1}{2} \sum_{i=1}^{N} (\bar{z}(x_i, \mathbf{p}) - z_i)^2 = \frac{1}{2} \sum_{i=1}^{N} (p_2 x_i^2 + p_1 x_i + p_0 - z_i)^2 \tag{4.22}$$

The minimum value of $E(\mathbf{p})$ is computed solving the following system of equations:

$$\begin{cases} \frac{\partial E(\mathbf{p})}{\partial p_2} = 0 \\ \frac{\partial E(\mathbf{p})}{\partial p_1} = 0 \\ \frac{\partial E(\mathbf{p})}{\partial p_0} = 0 \end{cases} \tag{4.23}$$

Which can be written explicitly in the system:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i & n \end{bmatrix} \begin{bmatrix} p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix} \tag{4.24}$$

The vertical profile is then validated considering its inliers number: if they are not enough, the cluster is discarded.

4.6.4.3 Height

Once the horizontal and vertical profiles are computed the height from the ground is extracted: it is computed as the average value of the DEM gradient magnitude corresponding to the barrier inliers.

The final result of the detection is shown in figure 4.10.

Figure 4.10: Algorithm final result: the detected barriers are reprojected back onto the right image.

4.7 Experimental Result

The robustness of this algorithm was proved on a test vehicle equipped with a stereo rig on its roof pointing in the driving direction.

The algorithm was executed on an embedded processor in real-time. Since this algorithm shares both the input and some of the processing with the stereovision-based curbs detector described on Chapter 3, the two processes were merged and executed simultaneously. In this case, the total processing time of both the algorithms was under 8ms.

The detection range of the algorithm is 80m.

Since the definition of barrier is very general and it can comprise a wide set of objects, ranging from hedges to guard-rails, the process of manual annotation is too expensive and error-prone. For this reason, qualitative results will be presented only for the particular case of guard-rails detection.

4.7.1 Qualitative Results

The algorithm was evaluated by a human expert in a wide range of different scenarios. Moreover, the algorithm demonstrated to be robust enough in order to be used as an input for ego-lane estimation task.

Some examples of detections are shown in figure 4.11.

This approach, detecting barriers as continuous elevation changes independently from the object visual aspect, demonstrated to be sufficiently generic in order to detect a wide set of different structures. In this way it is not necessary to develop several ad-hoc detectors (for instance only a guard-rails detector), overloading the system with expensive processing.

Most of the false positives were due to real obstacles that are moving slowly: in this case, the tracking algorithm, which is pretty simple, was not able to discriminate the movement and the candidate could be marked as a valid barrier. In figure 4.12 it is shown a case where the track was moving slowly and it was erroneously detected as a barrier.

(a) Fence.

(b) Driving corridor.

(c) Guard-rail and hedge.

(d) Wall.

Figure 4.11: Some qualitative results, in different scenarios, of the proposed stereovision-based algorithm for barriers detection.

Figure 4.12: Example of a false barrier detected in correspondence of a slowing-moving object.

4.7.2 Quantitative Results

As mentioned above, since the definition of barrier is quite large, it is not possible to manually annotate these structures. For this reason, a benchmark based on the detection of only guard-rails in highway scenario will be presented.

A sequence from KITTI [3] odometry dataset (odometry06) was manually annotated with polylines indicating the presence of a guard-rail, as shown in figure 4.13. This sequence is recorded in a highway scenario with merging where guard-rails delimit both sides of the road. Since the sequence comprises 1100 frames recorded at 10 frames per second, a frame every 10 was annotated.

Figure 4.13: Examples of annotation of guard-rails.

The benchmark procedure is similar to the one explained in Section 3.7.

Results are summarized in table 4.1, while some detections are shown in figure 4.14.

Algorithm performances were stable and robust: guard-rails were detected in both side of the road during the whole recording, even if they limit side lanes.

Recording	# annotated frames	# total barriers	Sensitivity	Precision
odometry06	110	188	92.5%	90.06%

Table 4.1: Barriers detection results evaluated on the KITTI visual odometry06 dataset.

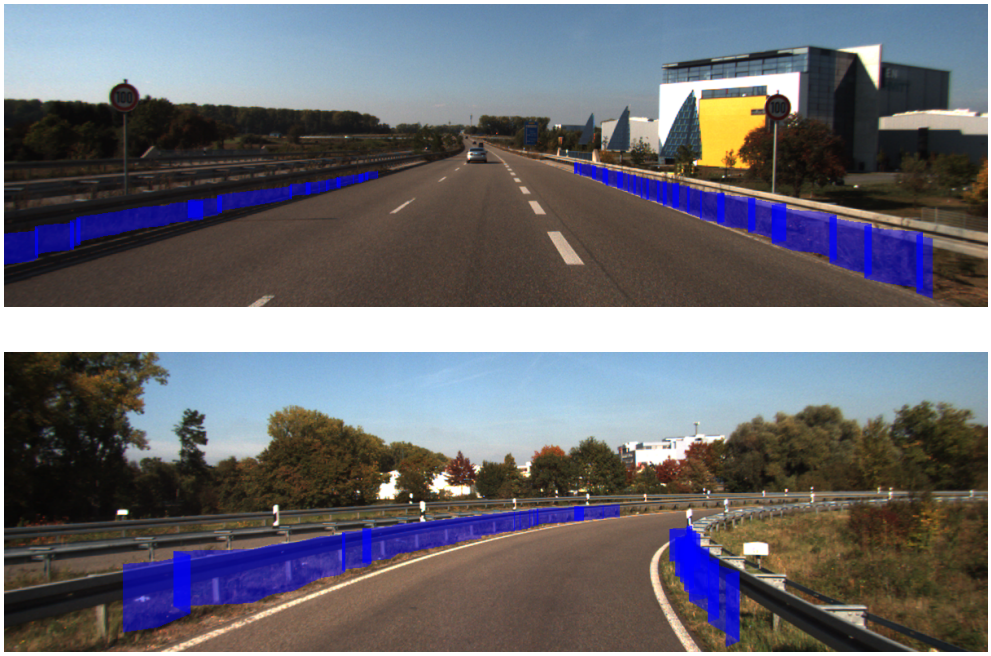


Figure 4.14: Some qualitative detection of barriers from the KITTI dataset.

Examples of errors are shown in figure 4.15. Some false negatives were present especially when barriers segments were particularly short. False positives appeared mainly in correspondence of trees and hedges at the roadside.

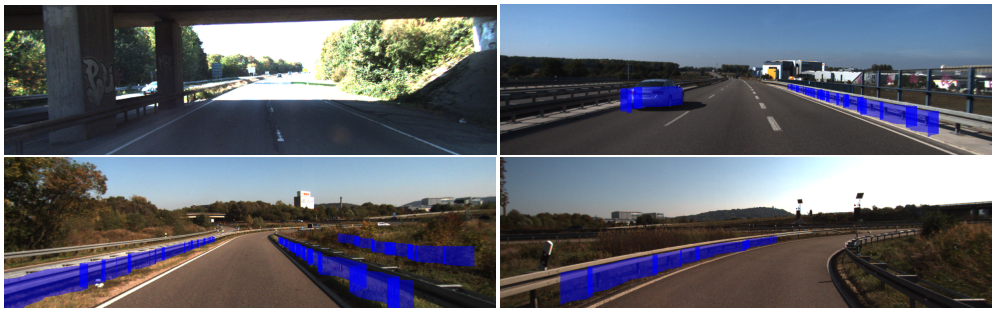


Figure 4.15: Some examples of errors in the detection of barriers.

Chapter 5

Deep Curbs

In this chapter, the proposed semantic segmentation approach to curbs detection will be described.

The development of this algorithm is born from the need to overcome all the limitations of the method proposed in Chapter 3. For this reason, an opposite approach based on mono camera and deep-learning as been chosen. Not much work has been done in this field as a deep learning application. With respect to the method presented in Chapter 3 and other stereo vision approaches in literature, this method is designed to increase the detection range and the robustness to scenarios with poor lighting conditions.

This chapter is organized as follows. Section 5.1 is a description of the chosen dataset, in Section is an overview of the used data augmentation techniques 5.2. In Section 5.3 the used deep learning model will be described. The loss function and the training procedure will be presented in Section 5.4 and 5.5. Experimental results will be analyzed in Section 5.6.

5.1 Dataset

Deep neural networks require millions of training images in order to achieve state-of-the-art performances. In particular, for the case of autonomous driving applications, it

is not simple to take advantage of deep learning power of since the vast majority of the existing datasets are limited in terms of annotations richness and scenarios variation. For these reasons models trained on these data-sets easily tend to over-fit the training data.

In the case of curbs detection, this problem is even more accentuate: it is quite simple to over-fit the training data since curbs are always at the border between drivable and non-drivable space, more or less in the same position on the image and with same colors. The risk is that models prefer to learn the contrast between the road and other objects without learning the concept of curb itself.

To overcome these limitations, the Berkeley Deep Drive (BDD100k) dataset [1] has been chosen for training. It is a large scale diving dataset: in total there are 100k driving videos collected in different geographical regions (such as New York and San Francisco Bay Area). Moreover, the dataset contains various scenarios from urban to highway streets in different lighting and weather conditions. Every video is 40 seconds long, the frame at the 10^{th} second of each video is annotated.

Annotated categories are:

- objects: bus, traffic light, traffic sign , person, bike, truck, motor, car, train, and rider;
- lane: curb, double white, double yellow, other double, single white, single yellow, other single and crosswalk;
- drivable areas: drivable and alternative.

There is a total of 157401 road curbs instances annotated as Bezier curves on the image, and curbs are classified as a special case of lane marking.

In this work have also been used lane markings and drivable areas annotations to add a penalty for erroneous classifications of curbs on the drivable space.

Some examples of the used annotations are shown in figure 5.1.

There are also 5683 fine-grained pixel-level annotated images, but they are not used in this work.

In table 5.1 is shown the weather distribution: it is possible to notice that the dataset contains a lot of extreme weather conditions like rainy foggy or snowy scenarios.

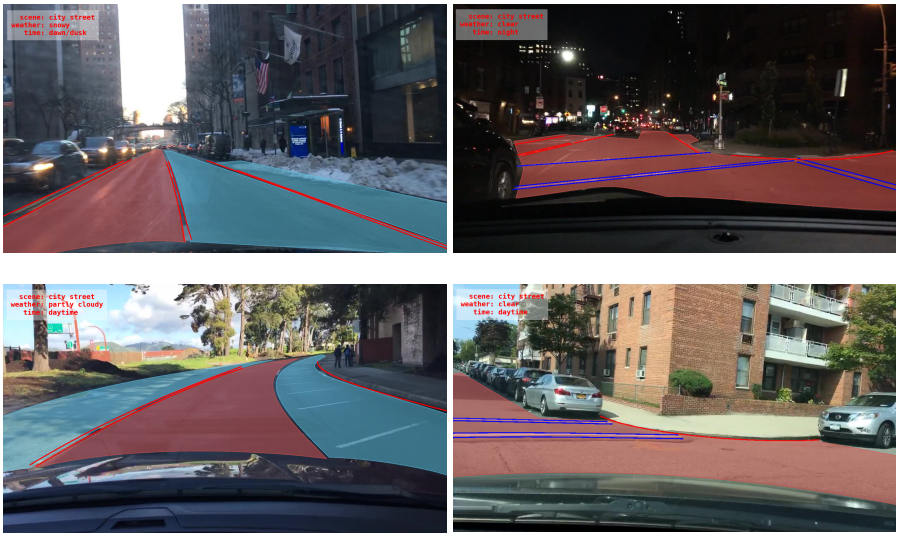


Figure 5.1: Examples of road markings and drivable area annotations from BDD100k dataset [1]. Road markings (curbs and lanes) are annotated in red if they are vertical or in blue if they are parallel. Drivable area is annotated in red, while alternative areas are in light blue.

As shown in table 5.2 the dataset is also well distributed across day hours, since it has approximately an equal number of daytime and night examples.

In table 5.3 as instead shown the number of different scenes.

	clear	partly cloudy	overcast	rainy	snowy	foggy
# instances	53525	7066	12591	7125	7888	181

Table 5.1: Distribution of images in weather

	dawndusk	daytime	night
# instances	7285	52511	39986

Table 5.2: Distribution of images in hours

This diversity enables to study the problem of semantic segmentation applied to curbs detection.

5.2 Data augmentation

For some applications, acquiring and annotating a sufficient amount of data for training a deep neural network can be difficult and expensive. In the specific case of curbs detection (and automotive applications in general), an insufficient amount of training data can lead to over-fitting, since curbs are more or less in the same position at the border of the road, and they have a similar visual aspect. Thus millions of annotated images are required.

This kind of over-fitting can be regularized by applying some techniques of data augmentation generating additional training data by transforming the original input image. It has been shown that data augmentation is especially beneficial for image

	residential	highway	urban	parking	gas stations	tunnel
# instances	11753	24987	61960	536	40	206

Table 5.3: Driving scenarios distribution

segmentation tasks [58] and that it is useful even in larger datasets [98]. It can also act as an additional regularizer for very deep architectures [99].

In this work, data augmentation is done online by directly transforming the input image before being fed into the deep neural network.

Used data augmentation techniques are:

- random vertical flip: since training examples are still valid if they are flipped a random rotation on the vertical axis is applied with probability 0.5. Therefore, the resulting batch has an equal chance to contain the flipped o non-flipped version of the image. An example is shown in figure 5.2;

(a) Original (b) Modified

Figure 5.2: Flip example.

- intensity scaling: images in the batch are transformed, with probability 0.5, by element-wise multiplication with a scalar. This would results in a contrast change since multiplying the image with a scalar has a more significant impact on larger image intensities. An example is shown on figure 5.3;
- additive noise: additive noise is added to the image before feeding it into the neural network. The noise is drawn from a Gaussian distribution:

$$N \sim \mathcal{N}(\mu, \sigma^2) \tag{5.1}$$

and the resulting image I_A is obtained as:

$$I_A = I + N \tag{5.2}$$

(a) Original (b) Modified

Figure 5.3: Intensity scaling example.

In figure 5.4 an example is shown. This should force the net to be tolerant against uncorrelated variations in pixels values, reducing the overfitting;

(a) Original (b) Modified

Figure 5.4: Additive Gaussian noise example.

- channels permutation: RGB channels order is randomly permuted with a probability of 0.5. This should encourage the neural network to learn the concept of curb without making strong assumptions regarding their color. An example is shown in figure 5.5.

(a) Original (b) Modified

Figure 5.5: Channels permutation example.

5.3 Model

In this work we used a deep neural network architecture named ENet (efficient neural network) [100], it is a light-weight and very efficient network that uses only a few parameters and computations to achieve state-of-the-art results. This net is designed specifically for low-latency semantic segmentation tasks.

ENet achieves a frame rate of 14.6 fps (69 ms) on the NVIDIA TX1 embedded system module for 640x360 input images. The size of the model in half precision floating point and for an input of size 3x640x360 is 0.7MB for a total of 370000 parameters. This enables to fit the net in the on-chip memory of embedded hardware.

ENet has been trained over different datasets (like Cityscapes [2] and CamVid [101]) and its performances match state-of-the-art models.

5.3.1 Architecture

The net architecture, presented in table 5.4, is an encoder plus decoder network with an initial stage: the encoder part is a CNN designed for classification, while the decoder is an upsampling model designed to propagate the categories back into the original image for segmentation. The initial stage consists in only one block; the encoder contains five stages while the last two stages belong to the decoder. A full convolution is placed as the last stage, this block alone takes a large portion of the

decoder processing time.

Block name	Type	Output size
initial block		16x256x144
bottleneck1.0	downsampling	64x128x72
bottleneck1.1		64x128x72
bottleneck1.2		64x128x72
bottleneck1.3		64x128x72
bottleneck1.4		64x128x72
bottleneck2.0	downsampling	128x64x36
bottleneck2.1		128x64x36
bottleneck2.2	dilated 2	128x64x36
bottleneck2.3	asymmetric 5	128x64x36
bottleneck2.4	dilated 4	128x64x36
bottleneck2.5		128x64x36
bottleneck2.6	dilated 8	128x64x36
bottleneck2.7	asymmetric 5	128x64x36
bottleneck2.8	dilated 16	128x64x36
as section 2 but without bottleneck2.0		
bottleneck4.0	upsampling	64x128x72
bottleneck4.1		64x128x72
bottleneck4.2		64x128x72
bottleneck5.0	upsampling	64x256x144
bottleneck5.1		64x256x144
bottleneck5.2		64x256x144
fullconv		*C*x512x288

Table 5.4: ENet architecture, in the case of 512x288 input size

The architecture is not symmetric since it consists on a larger encoder with respect to the decoder: the idea is that the encoder should be able to operate on low-resolution

data providing information for filtering while the decoder should upsample the data only fine-tuning the details.

Downsampling creates low-resolution images, and this operation has two main advantages:

- the net has a larger receptive field which allows to gain more context. This helps the net to learn not only how curbs look but also the context in which they appear;
- real-time performances can be achieved by compressing the input data in a more efficient representation.

For these reasons, ENet early blocks produce a small feature map by heavy reducing the size of the input data. However, in the case of semantic segmentation, downsampling has the disadvantage to require equal upsampling in order to output has the same size as the input. ENet addresses this issue as in SegNet [102]: in max-pooling layers indexes are saved and used then to produce sparse upsampling maps in the decoding module, this allows to reduce memory requirements.

Like ResNet [103], the architecture is divided into bottleneck modules. The sequence of operations composing these modules can be seen as decomposing a large convolutional layer into a series of simpler operations representing the low-rank approximation of the original layer. This factorizations allows large speedups and reduce the number of parameters. Bottlenecks can be downsampling (encoder) or upsampling (decoder) modules.

As shown in figure 5.6, a bottleneck module is made by three convolution layers: the first one is a 1x1 projection that reduces dimensionality followed by a main convolution module and a 1x1 expansion.

Between all the convolutions are placed Batch normalization [104] and PReLU [105]. Downsampling bottlenecks use max pooling, moreover a 2x2 convolution with stride 2 in both dimensions replaces the 1x1 projection. Upsampling modules, instead, use max-unpooling and padding is replaced with spacial convolution.

In some modules, the main convolution, is replaced with a sequence of 1x5 and 5x1 asymmetric convolutions since it has been demonstrated [106] [95] that convolutional

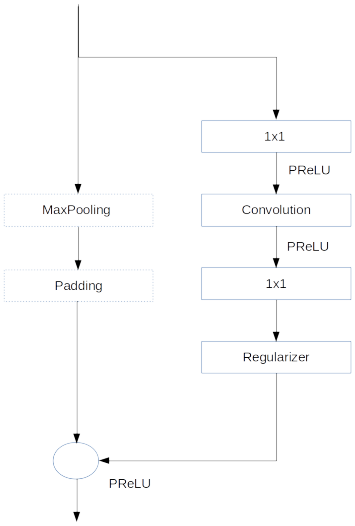


Figure 5.6: ENet bottleneck module

weights have a certain amount of redundancy and each nxn convolution can be decomposed into two subsequent smaller convolutions: one with an $nx1$ kernel and one with $1xn$ filter.

Spatial dropout [107] is used, after the 1x1 expansion, for regularization. Bias terms are never used in any of the projections in order to reduce the number of kernel calls and the overall memory operations.

The initial stage is shown in figure 5.7: there is a max-pooling block on 2x2 non-overlapping windows and a convolution with 13 filters. The two branches are then concatenated obtaining a total of 16 feature map. The initial block does not contribute to classification: its role is to act as a good features extractor, preparing the data for the following modules.

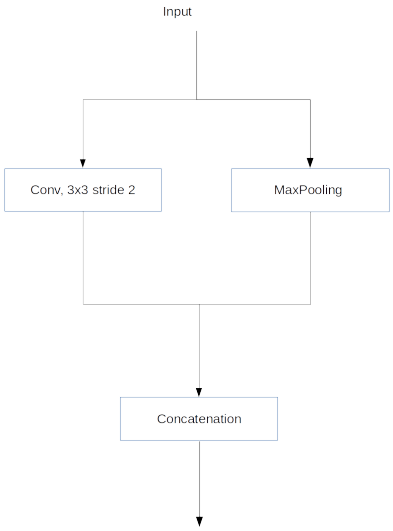


Figure 5.7: ENet initial module

5.4 Loss function

The loss function is computed by the average pixel-wise soft-max over the final feature map, combined with the cross-entropy function.

The soft-max is defined as:

$$p_c(\mathbf{x}) = \frac{exp(a_c(\mathbf{x}))}{\sum_{c'}^{C} exp(a_{c'}(\mathbf{x}))} \tag{5.3}$$

Where $a_c(\mathbf{x})$ denotes the activation in feature channel c at the pixel position $\mathbf{x} \in I$ and C is the number of classes, which is 2 in this case since classes are background

and curb.

The cross entropy, penalizes at each position the deviation of $p_{l(\mathbf{x})}(\mathbf{x})$ from 1 using:

$$E = \sum_{\mathbf{x} \in I} w(\mathbf{x}) \log(p_{l(\mathbf{x})}(\mathbf{x})) \tag{5.4}$$

where $l : I \rightarrow (1, ..., C)$ is the true label of each pixel and $w : I \rightarrow \Re$ is a weight map that is introduced to give some pixels more importance during the training.

A weight map $w(\mathbf{x})$ is precomputed for each training example to force the network to learn that curbs cannot be on the drivable area and to compensate the different frequency of pixels of the class curb in the training dataset. For every pixel $\mathbf{x} \in I$ the value of $w(\mathbf{x})$ is chosen as:

$$w(\mathbf{x}) = \begin{cases} 50 & \textit{if } \boldsymbol{x} \textit{ is labeled as a lane marking of drivable area} \\ 30 & \textit{if } \boldsymbol{x} \textit{ is labeled as a curb} \\ 1 & \textit{otherwise} \end{cases} \tag{5.5}$$

See figure 5.8 for an example.

(a) Input data (b) Ground truth (c) Weight map

Figure 5.8: Input data example (5.8a) with ground truth labeling (5.8b) and weight map (5.8c).

5.5 Training

ENet model is trained using the Adam optimization of stochastic descent [108]. Training is performed with batch size 20, weight decay $2e^{-4}$ and a starting learning

rate of $1e^{-5}$. In order to accelerate the convergence, the learning rate is decremented by one order of magnitude every time the error becomes stagnant.

As mentioned above the model is trained on the BDD100k [1] data set, which contains 60000 labeled images for training and 10000 for validation. The net is trained with images of size 512x288.

In figure 5.9 are shown the training and validation losses during the epochs.

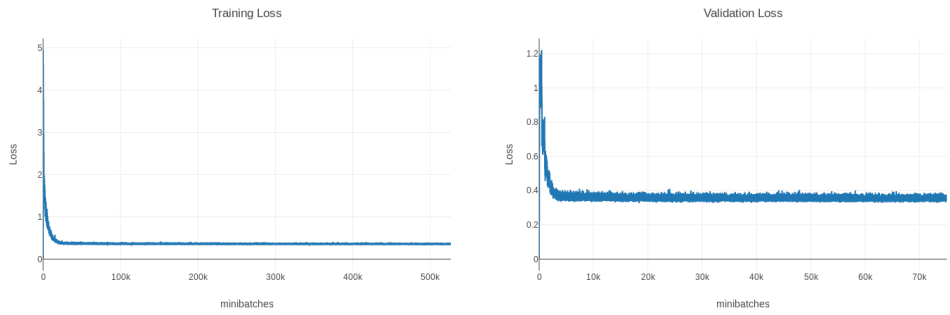


Figure 5.9: Training and validation errors over mini-batches.

5.6 Experimental Results

In this section, an extensive qualitative and quantitative analysis of the proposed method will be presented.

The algorithm was evaluated on the validation and test dataset of BDD100k [1]. Two different kinds of metrics are provided: object-level and pixel-level metrics.

It has not been evaluated on a test vehicle yet since this work is still in progress and it is not robust enough.

5.6.1 Qualitative Results

Figure 5.10 shows various example of segmentation results obtained with ENet architecture. These examples demonstrate that this approach yields robust qualitative results, event at far distances in the scene or with poor lighting conditions.

The detection tends to be coarse especially at far distances, and this suggests that a post-processing step to refine the segmentation would be necessary.

Figure 5.10: Segmentation results examples from BDD100k dataset [1], where curbs are highlighted in green.

However, some important misclassification errors are present, as the ones shown in figure 5.11.

In figure 5.11a a curb is detected on the rear window upright of the vehicle, it is probably because there is a sharp contrast change between the asphalt and the rear window upright. In figure 5.11b, instead, the right side curb is completely ignored since in the training set there are not many examples of curbs with this particular orientation. The left side curb, in figure 5.11c, is missed probably for its aspects since

(a) (b)

(c) (d)

Figure 5.11: Wrong segmentation results, from BDD100k dataset [1].

it is located in the middle of the road and since its color it is pretty similar to the asphalt. In figure 5.11d the curb on the right is missed due to a reflection on the windscreen; moreover, a false curb is detected in correspondence of the reflection.

5.6.2 Quantitative Results

The quality of segmentation was quantitatively evaluated, and two kinds of metrics were provided:

- object-level metric: the proposed approach was evaluated in terms of the number of detected curbs;
- pixel-level metric: intersection over union and Dice loss.

5.6.2.1 Object-Level Metric

A benchmark system on the image space was implemented. Segmentation results were firstly clustered in sets of contiguous pixels, and every cluster was then considered as a detected curb.

An annotation was said to be a true positive if a detection overlapped it. However, the overlap was considered valid only if a minimum number of points, proportional to the annotation curve length, was close enough to the annotation.

Annotations that were not overlapped with detections were considered false negatives.

Not associated detections were discarded if they were too small, and so they were not considered as false positives.

Table 5.5 summarizes the evaluation.

	# Annotated Examples	# Total Curbs	Sensitivity	Precision
ENet	10000	15821	89.50 %	76.5%

Table 5.5: Evaluation results in terms of detected objects on BDD100k [1] dataset.

Sensitivity indicates that results are very promising: on about ~16k instances of curbs almost the 90% was correctly detected. Considering that the dataset comprises challenging scenarios with poor illumination and extreme weather conditions these results are very encouraging and demonstrate that this is a good approach to achieve a robust level of detection.

There is still a not negligible fraction of false positives. Some of them are at the border of the road, these kind of false positives are not dangerous from the system point of view since they are at the border of the road. However, they could be an issue for localization.

5.6.2.2 Pixel-Level Metric

Two pixel-level metrics were evaluated since they are the most common in semantic segmentation problems:

- intersection over union (IoU):

$$IoU = \frac{TP}{TP + FN + FP} \tag{5.6}$$

- Dice loss or F1 score:

$$F1 = \frac{2TP}{2TP + FN + FP} \tag{5.7}$$

Since the segmentation tends to be coarse, the annotations were empathized making the polyline 9 pixels thick instead of only one pixel.

In table 5.6 and 5.7 IoU and F1 score results are presented.

	mean IoU	curbs IoU	background IoU
ENet	69.66 %	40.0%	99.32%

Table 5.6: Evaluation results in terms of intersection over union (IoU) on BDD100k [1] dataset.

	mean F1	curbs F1	background F1
ENet	78.40 %	57.41%	99.66%

Table 5.7: Evaluation results in terms of F1 score on BDD100k [1] dataset.

Despite the fact that the two metrics are positively correlated and that both take the average score over a set of inferences, there is a difference in how much they quantify that a classifier is worse than another. IoU metric measures something closer to the worse case performance, while F1 score is much closer to the average performance. However, both of them suffer from the fact that, in this particular problem, the two categories (curb and background) are highly unbalanced since the number pixels annotated as curbs are just a small fraction of the total.

Even if these metrics are not the best way to evaluate this approach, they indicate that the segmentation is still coarse and there are many false positives. This is in accordance with the considerations made in the previous evaluation.

Chapter 6

Conclusions

This work is focused on the development of road boundaries perception algorithms for autonomous vehicle applications. This is a challenging problem that requires to reach high levels of reliability and real-time performances.

The complete knowledge and comprehension of the vehicle surroundings are core tasks for every autonomous driving application and the algorithms aimed at the reconstruction of road geometry are at the basis of all the high-level applications. This work in particular addresses the specific problem of curbs and barriers detection, since these two structures are one of the most widespread elements that are usually present in the road scenario. Both curbs and barriers represent an essential input for many high-level applications such as, for instance: path-planning, ego-lane estimation, obstacle avoidance, and localization.

During this study emerged that the process of development of computer vision solutions for the automotive industry is accompanied by several constraints: reliable and robust results are required as well as methods for handling adverse environmental conditions. Moreover, the software has to be designed in order to require al little as hardware as possible.

Cars, in order to be sell, needs to be well-designed: they should have an attractive appearance and they should be aerodynamic to limit fuel consumption as much as possible. In this scenario, artificial vision based on camera technology is demonstrat-

ing all its powerful characteristics: cameras are cheap, they can be easily integrated on a vehicle and, above all, visual data has huge information content. One of the main drawbacks of computer vision is the amount of data to process, for this reasons automotive solutions has to be carefully designed.

With the goal to achieve reliable, robust and fast solutions in mind, the proposed methods were developed, validated and demonstrated with real-time implementation on an embedded platform.

The summarized conclusions for each chapter are presented in the following.

In Chapter 3 a stereovision-based approach for curbs detection is presented. It takes inspiration from methods present in literature with the objective to increase detection quality and enabling the execution on embedded hardware with real-time performance. The stereo camera was chosen as a sensor since it allows curbs three-dimensional reconstruction. The proposed method was validated with both quantitative analysis and qualitative tests on real-world scenarios. Moreover, it was able to run in real-time on an embedded platform sharing computational resources with other perception algorithms. This method demonstrated to be able to produce highly reliable results even at far distances which are used by other high-level tasks composing the autonomous system. The detection range is incremented with respect to other state-of-the-art methods. Quantitative analysis confirmed qualitative tests. However, this algorithm suffers from the typical problems related to stereovision: loss of accuracy in the presence of poor lighting or mis-calibration conditions. These problems are even more emphasized by the fact that the algorithm search for very low obstacles that are easily disturbed by noisy data.

Chapter 4 presented an approach for detecting road barriers based on stereovision. Unlike other state-of-the-art algorithms which are focused on specific structures, in this work the problem is addressed in its most generic meaning: it is considered as a barrier every vertical structure defining a driving corridor regardless of its visual aspect. This work takes inspiration from the approach to curbs detection presented in Chapter 3 and shares with it most of the processing. This trick, along with the idea of barriers as a generic structure, allowed to detect in real-time and with a minimal computational impact a wide range of objects with the same processing. The algorithm

was tested on many real-world scenarios, from urban to highway, and moreover, it was used as input for ego-lane estimation. As the method for curbs detection proposed on Chapter 3 also this algorithm suffers in poor lighting conditions while it is more robust to mis-calibration issues since it detects bigger structures that are less sensitive to 3D reconstruction noise.

In Chapter 5 a further approach for curbs detection is presented. The development of this algorithm born from the need to overcome all the limitations of the method proposed in Chapter 3. For this reason, an opposite approach, based on deep-learning and a mono camera, was chosen. Not much work has been done in this field as a deep learning application. A very lightweight and efficient model, called ENet, is used in this work since it has to be suitable for a future integration on an embedded processor. This approach demonstrated promising results enabling detections even at far distances also at nighttime. However it is still suffering from incorrect detection issues, post-processing operations are therefore needed in order to get reliable results.

As a future research, the deep learning-based solution presented in Chapter 5 could be used as an additional cue to the approach described in Chapter 3: while stereo vision allows to reconstruct the 3D geometry of the observed scene, at the other side monocular technologies take into account the appearance of the object and allow detections even at far distance. The fusion of the two solutions could improve the detection in difficult lighting and weather conditions, and it could also enhance the detection distance. Otherwise other kinds of solutions can be taken into account, for example a full deep learning end-to-end solution based on stereovision.

One problem related to the use of convolutional neural networks is that a large amount of data is required, especially in the automotive field. Having a large amount of data from the beginning is not always possible; moreover, the process of learning should be continuous and models have to be exposed to new knowledge. For this reason, datasets must be continuously extended with new examples. Adaptive incremental learning methodologies could be used to learn new unseen training data without the need of retraining models on the complete dataset. These techniques could be exploited for future research, in the specific task of curbs detection, in order to obtain increasingly reliable results.

Automotive industry requires high levels of reliability for computer vision applications. Some studies [109, 110] demonstrated that deep neural networks, as other machine learning techniques, are vulnerable to adversarial input examples, which are obtained by intentionally adding small perturbations to inputs with the objective of obtaining a completely wrong prediction. This leads to serious concerns about the security of deep learning models in safety-critical applications. Analyzing the causes of adversarial examples could help to fix these vulnerabilities, state of the art researches are still working on methods for adversarial example constructions and defense in order to obtain better understanding and higher security levels.

Bibliography

[1] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling, 05 2018.

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016. URL: `http://arxiv.org/abs/1604.01685`, `arXiv:1604.01685`.

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, June 2012. `doi:10.1109/CVPR.2012.6248074`.

[4] US Department of Transportation. August 2017 traffic volume trends. URL: `https://www.fhwa.dot.gov/policyinformation/travel_monitoring/17augtvt/page2.cfm`.

[5] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens,

Antone Vogt, and Sebastian Thrun. *Junior: The Stanford Entry in the Urban Challenge*, pages 91–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. URL: https://doi.org/10.1007/978-3-642-03991-1_3, doi:10.1007/978-3-642-03991-1_3.

[6] Hernán Badino, Uwe Franke, and David Pfeiffer. The stixel world - a compact medium level representation of the 3d-world. In Joachim Denzler, Gunther Notni, and Herbert Süße, editors, *Pattern Recognition*, pages 51–60, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] R. Labayrade, D. Aubert, and J. . Tarel. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 646–651 vol.2, June 2002. doi:10.1109/IVS.2002.1188024.

[8] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, M. Panciroli, and A. Prioletti. Proud-public road urban driverless test: Architecture and results. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 648–654, June 2014. doi:10.1109/IVS.2014.6856478.

[9] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. doi:10.1109/2.30720.

[10] F. Oniga and S. Nedevschi. Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection. *IEEE Transactions on Vehicular Technology*, 59(3):1172–1182, March 2010. doi:10.1109/TVT.2009.2039718.

[11] D. F. Huber and M. Hebert. A new approach to 3-d terrain mapping. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, volume 2, pages 1121–1127 vol.2, 1999. doi:10.1109/IROS.1999.812830.

[12] Simon Lacroix, Anthony Mallet, David Bonnafous, Gérard Bauzil, Sara Fleury, Matthieu Herrb, and Raja Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *The International Journal of Robotics Research*, 21(10-11):917–942, 2002. URL: https://doi.org/10.1177/027836490202101084l, arXiv:https://doi.org/10.1177/027836490202101084l, doi:10.1177/027836490202101084l.

[13] Maarten Vergauwen, Marc Pollefeys, and Luc Van Gool. A stereo-vision system for support of planetary surface exploration. *Machine Vision and Applications*, 14:5–14, 2001.

[14] H. Roncancio, M. Becker, A. Broggi, and S. Cattani. Traversability analysis using terrain mapping and online-trained terrain type classifier. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1239–1244, June 2014. doi:10.1109/IVS.2014.6856427.

[15] F. Oniga and S. Nedevschi. Curb detection for driving assistance systems: A cubic spline-based approach. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 945–950, June 2011. doi:10.1109/IVS.2011.5940580.

[16] J. Siegemund, U. Franke, and W. Förstner. A temporal filter approach for detection and reconstruction of curbs and road surfaces based on conditional random fields. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 637–642, June 2011. doi:10.1109/IVS.2011.5940447.

[17] M. Harville. Stereo person tracking with short and long term plan-view appearance models of shape and color. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, pages 522–527, Sept 2005. doi:10.1109/AVSS.2005.1577323.

[18] L. Matthies and A. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Proceedings. 1988 IEEE International Conference on*

Robotics and Automation, pages 727–733 vol.2, April 1988. doi:10.1109/ROBOT.1988.12145.

[19] F. Valenti, F. Ghidini, M. Patander, and A. Broggi. A ground truth building approach for evaluation of grid based discretization techniques in automotive scenarios. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 960–965, June 2016. doi:10.1109/IVS.2016.7535504.

[20] Patrick Pfaff, Rudolph Triebel, and Wolfram Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *The International Journal of Robotics Research*, 26(2):217–230, 2007. URL: https://doi.org/10.1177/0278364906075165, arXiv:https://doi.org/10.1177/0278364906075165, doi:10.1177/0278364906075165.

[21] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2276–2282, Oct 2006. doi:10.1109/IROS.2006.282632.

[22] Manish Kumar and Devendra P. Garg. Three-dimensional occupancy grids with the use of vision and proximity sensors in a robotic workcell. 73, 01 2004.

[23] H. Lategahn, W. Derendarz, T. Graf, B. Kitt, and J. Effertz. Occupancy grid computation from dense stereo and sparse structure and motion points for automotive applications. In *2010 IEEE Intelligent Vehicles Symposium*, pages 819–824, June 2010. doi:10.1109/IVS.2010.5548078.

[24] R. Danescu and S. Nedevschi. A particle-based solution for modeling and tracking dynamic digital elevation maps. *IEEE Transactions on Intelligent Transportation Systems*, 15(3):1002–1015, June 2014. doi:10.1109/TITS.2013.2291447.

[25] A. Y. Hata, F. S. Osorio, and D. F. Wolf. Robust curb detection and vehicle localization in urban environments. In *2014 IEEE Intelligent Vehicles Sym-*

posium Proceedings, pages 1257–1262, June 2014. doi:10.1109/IVS.2014.6856405.

[26] B. Qin, Z. J. Chong, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus. Curb-intersection feature based monte carlo localization on urban roads. In *2012 IEEE International Conference on Robotics and Automation*, pages 2640–2646, May 2012. doi:10.1109/ICRA.2012.6224913.

[27] Stephen Se and Michael Brady. Vision-based detection of kerbs and steps. In *In Eighth British Machine Vision Conference BMVC '97*, pages 410–419, 1997.

[28] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986. doi:10.1109/TPAMI.1986.4767851.

[29] A. Seibert, M. Hähnel, A. Tewes, and R. Rojas. Camera based detection and classification of soft shoulders, curbs and guardrails. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 853–858, June 2013. doi:10.1109/IVS.2013.6629573.

[30] V. Prinet, J. Wang, J. Lee, and D. Wettergreen. 3d road curb extraction from image sequence for automobile parking assist system. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3847–3851, Sept 2016. doi:10.1109/ICIP.2016.7533080.

[31] R. Turchetto and R. Manduchi. Visual curb localization for autonomous navigation. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 2, pages 1336–1342 vol.2, Oct 2003. doi:10.1109/IROS.2003.1248830.

[32] Xiaoye Lu and R. Manduchi. Detection and localization of curbs and stairways using stereo vision. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4648–4654, April 2005. doi:10.1109/ROBOT.2005.1570837.

[33] M. Enzweiler, P. Greiner, C. Knöppel, and U. Franke. Towards multi-cue urban curb recognition. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 902–907, June 2013. doi:10.1109/IVS.2013.6629581.

[34] R. Bichsel and P. V. K. Borges. Low-obstacle detection using stereo vision. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4054–4061, Oct 2016. doi:10.1109/IROS.2016.7759597.

[35] Mingmei Cheng, Yigong Zhang, Yingna Su, Jose M. Alvarez, and Hui Kong. Curb detection for road and sidewalk detection. *IEEE Transactions on Vehicular Technology*, PP:1–1, 08 2018. doi:10.1109/TVT.2018.2865836.

[36] Vivek Pradeep, Gérard G. Medioni, and James D. Weiland. Piecewise planar modeling for step detection using stereo vision. 2008.

[37] C. Fernández, R. Izquierdo, D. F. Llorca, and M. A. Sotelo. Road curb and lanes detection for autonomous driving on urban scenarios. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1964–1969, Oct 2014. doi:10.1109/ITSC.2014.6957993.

[38] E. Pollard, J. Perez, and F. Nashashibi. Step and curb detection for autonomous vehicles with an algebraic derivative-based approach applied on laser rangefinder data. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 684–689, June 2013. doi:10.1109/IVS.2013.6629546.

[39] W. S. Wijesoma, K. R. S. Kodagoda, and A. P. Balasuriya. Road-boundary detection and tracking using ladar sensing. *IEEE Transactions on Robotics and Automation*, 20(3):456–464, June 2004. doi:10.1109/TRA.2004.825269.

[40] Jörg Stückler. In-lane localization in road networks using curbs detected in omnidirectional height images. 2008.

[41] M. Kellner, U. Hofmann, M. E. Bouzouraa, H. Kasper, and S. Neumaier. Laserscanner based road curb feature detection and efficient mapping using local curb descriptions. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2602–2609, Oct 2014. doi:10.1109/ITSC.2014.6958107.

[42] K. R. S. Kodagoda, W. S. Wijesoma, and A. P. Balasuriya. Cute: curb tracking and estimation. *IEEE Transactions on Control Systems Technology*, 14(5):951–957, Sept 2006. doi:10.1109/TCST.2006.876929.

[43] R. Aufrere, C. Mertz, and C. Thorpe. Multiple sensor fusion for detecting location of curbs, walls, and barriers. In *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*, pages 126–131, June 2003. doi:10.1109/IVS.2003.1212896.

[44] Jun Tan, Jian Li, Xiangjing An, and Hangen He. Robust curb detection with fusion of 3d-lidar and camera data. In *Sensors*, 2014.

[45] Orazio Gallo, Roberto Manduchi, and Abbas Rafii. Robust curb and ramp detection for safe parking using the canesta tof camera. 06 2008.

[46] J. Siegemund, D. Pfeiffer, U. Franke, and W. Förstner. Curb reconstruction using conditional random fields. In *2010 IEEE Intelligent Vehicles Symposium*, pages 203–210, June 2010. doi:10.1109/IVS.2010.5548096.

[47] F. Oniga and S. Nedevschi. Polynomial curb detection based on dense stereovision for driving assistance. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1110–1115, Sept 2010. doi:10.1109/ITSC.2010.5625169.

[48] F. Oniga, S. Nedevschi, and M. M. Meinecke. Curb detection based on a multi-frame persistence map for urban driving scenarios. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 67–72, Oct 2008. doi:10.1109/ITSC.2008.4732706.

[49] J. Maye, R. Kaestner, and R. Siegwart. Curb detection for a pedestrian robot in urban environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 367–373, May 2012. doi:10.1109/ICRA.2012.6224593.

[50] T. Michalke, R. Kastner, J. Fritsch, and C. Goerick. A self-adaptive approach for curbstone/roadside detection based on human-like signal processing and multi-sensor fusion. In *2010 IEEE Intelligent Vehicles Symposium*, pages 307–312, June 2010. doi:10.1109/IVS.2010.5548055.

[51] A. Wimmer, T. Weiss, F. Flogel, and K. Dietmayer. Automatic detection and classification of safety barriers in road construction sites using a laser scanner. In *2009 IEEE Intelligent Vehicles Symposium*, pages 578–583, June 2009. doi:10.1109/IVS.2009.5164342.

[52] G. Alessandretti, A. Broggi, and P. Cerri. Vehicle and guard rail detection using radar and vision data fusion. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):95–105, March 2007. doi:10.1109/TITS.2006.888597.

[53] R. Danescu, S. Sobol, S. Nedevschi, and T. Graf. Stereovision-based side lane and guardrail detection. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1156–1161, Sept 2006. doi:10.1109/ITSC.2006.1707378.

[54] T. Scharwächter, M. Schuler, and U. Franke. Visual guard rail detection for advanced highway assistance systems. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 900–905, June 2014. doi:10.1109/IVS.2014.6856573.

[55] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. URL: http://dl.acm.org/citation.cfm?id=1623264.1623280.

[56] Andreas Ess, Tobias Mueller, Helmut Grabner, and Luc Van Gool. Segmentation-based urban traffic scene understanding. In *BMVC*, 2009.

[57] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras. Road-sign detection and recognition based on support vector machines. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):264–278, June 2007. doi:10.1109/TITS.2007.895311.

[58] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL: http://arxiv.org/abs/1505.04597, arXiv:1505.04597.

[59] Cohen Assaf, Rivlin Ehud, Shimshoni Ilan, and Sabo Edmond. Memory based active contour algorithm using pixel-level classified images for colon crypt segmentation. *Comp. Med. Imag. and Graph.*, 43:150–164, 2015. URL: http://dblp.uni-trier.de/db/journals/cmig/cmig43.html#CohenRSS15.

[60] Ji Wan, Dayong Wang, Steven C. H. Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *ACM Multimedia*, 2014.

[61] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, Aug 2002. doi:10.1109/TPAMI.2002.1023800.

[62] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *CoRR*, abs/1502.06807, 2015. URL: http://arxiv.org/abs/1502.06807, arXiv:1502.06807.

[63] Michael Lew, Erwin M. Bakker, Nicu Sebe, and Thomas S. Huang. Human-computer intelligent interaction: A survey. In Michael Lew, Nicu Sebe,

Thomas S. Huang, and Erwin M. Bakker, editors, *Human–Computer Interaction*, pages 1–5, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[64] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and José García Rodríguez. A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857, 2017. URL: http://arxiv.org/abs/1704.06857, arXiv:1704.06857.

[65] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005. doi:10.1109/CVPR.2005.177.

[66] Bourdev Lubomir, Maji Subhransu, Brox Thomas, editor="Daniilidis Kostas Malik Jitendra", Maragos Petros, and Paragios Nikos. Detecting people using mutually consistent poselet activations. In *Computer Vision – ECCV 2010*, pages 168–181, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[67] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94, doi:10.1023/B:VISI.0000029664.99615.94.

[68] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia. URL: http://www.sciencedirect.com/science/article/pii/S1077314207001555, doi:https://doi.org/10.1016/j.cviu.2007.09.014.

[69] Lauren Barghout. Visual taxometric approach to image segmentation using fuzzy-spatial taxon cut yields contextually relevant regions. In Anne Laurent, Olivier Strauss, Bernadette Bouchon-Meunier, and Ronald R. Yager, editors,

Information Processing and Management of Uncertainty in Knowledge-Based Systems, pages 163–173, Cham, 2014. Springer International Publishing.

[70] R. Kimmel and A.M. Bruckstein. Regularized laplacian zero crossings as optimal edge integrators. *International Journal of Computer Vision*, 53(3):225–243, Jul 2003. URL: https://doi.org/10.1023/A:1023030907417, doi:10.1023/A:1023030907417.

[71] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *CoRR*, abs/1210.5644, 2012. URL: http://arxiv.org/abs/1210.5644, arXiv:1210.5644.

[72] Xuming He, R. S. Zemel, and M. A. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II, June 2004. doi:10.1109/CVPR.2004.1315232.

[73] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 1–15, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[74] L. Zheng, G. Li, and Y. Bao. Improvement of grayscale image 2d maximum entropy threshold segmentation method. In *2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM)*, volume 1, pages 324–328, Jan 2010. doi:10.1109/ICLSIM.2010.5461410.

[75] S. Hu, E. A. Hoffman, and J. M. Reinhardt. Automatic lung segmentation for accurate quantitation of volumetric x-ray ct images. *IEEE Transactions on Medical Imaging*, 20(6):490–498, June 2001. doi:10.1109/42.929615.

[76] Anping Xu, Lijuan Wang, Sha Feng, and Yunxia Qu. Threshold-based level set method of image segmentation. In *Intelligent Networks and Intelligent Systems (ICINIS), 2010 3rd International Conference on*, pages 703–706. IEEE, 2010.

[77] John A Hartigan. Clustering algorithms. 1975.

[78] Dan C. Ciresan, Luca M. Gambardella, Alessandro Giusti, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *IN NIPS*, pages 2852–2860, 2012.

[79] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, Aug 2013. doi:10.1109/TPAMI.2012.231.

[80] Pedro H. O. Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene parsing. *CoRR*, abs/1306.2795, 2013. URL: http://arxiv.org/abs/1306.2795, arXiv:1306.2795.

[81] Feng Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371, Sept 2005. doi:10.1109/TIP.2005.852470.

[82] Bharath Hariharan, Pablo Arbelaez, Ross B. Girshick, and Jitendra Malik. Simultaneous detection and segmentation. *CoRR*, abs/1407.1808, 2014. URL: http://arxiv.org/abs/1407.1808, arXiv:1407.1808.

[83] Saurabh Gupta, Ross B. Girshick, Pablo Arbelaez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. *CoRR*, abs/1407.5736, 2014. URL: http://arxiv.org/abs/1407.5736, arXiv:1407.5736.

[84] Yaroslav Ganin and Victor S. Lempitsky. N^4-fields: Neural network nearest neighbor fields for image transforms. *CoRR*, abs/1406.6558, 2014. URL: http://arxiv.org/abs/1406.6558, arXiv:1406.6558.

[85] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL: http://arxiv.org/abs/1411.4038, arXiv:1411.4038.

[86] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL: http://arxiv.org/abs/1409.1556, arXiv:1409.1556.

[87] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010. URL: https://doi.org/10.1007/s11263-009-0275-4, doi:10.1007/s11263-009-0275-4.

[88] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[89] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. URL: http://arxiv.org/abs/1606.00915, arXiv:1606.00915.

[90] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015. URL: http://arxiv.org/abs/1511.07122, arXiv:1511.07122.

[91] Fisher Yu, Vladlen Koltun, and Thomas A. Funkhouser. Dilated residual networks. *CoRR*, abs/1705.09914, 2017. URL: http://arxiv.org/abs/1705.09914, arXiv:1705.09914.

[92] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[93] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. URL: http://arxiv.org/abs/1611.05431, arXiv:1611.05431.

[94] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015. doi:10.1109/CVPR.2015.7298594.

[95] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL: http://arxiv.org/abs/1512.00567, arXiv:1512.00567.

[96] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[97] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814 vol. 2, June 2005. doi:10.1109/CVPR.2005.56.

[98] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL: http://arxiv.org/abs/1409.0575, arXiv:1409.0575.

[99] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Vol-*

ume 1, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[100] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, 2016. URL: http://arxiv.org/abs/1606.02147, arXiv:1606.02147.

[101] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30:88–97, 2009.

[102] Badrinarayanan Vijay, Kendall Alex, and Cipolla Roberto. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL: http://arxiv.org/abs/1512.03385, arXiv:1512.03385.

[104] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL: http://arxiv.org/abs/1502.03167, arXiv:1502.03167.

[105] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL: http://arxiv.org/abs/1502.01852, arXiv:1502.01852.

[106] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. *CoRR*, abs/1412.5474, 2014. URL: http://arxiv.org/abs/1412.5474, arXiv:1412.5474.

[107] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. *CoRR*, abs/1411.4280, 2014. URL: `http://arxiv.org/abs/1411.4280`, `arXiv:1411.4280`.

[108] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL: `http://arxiv.org/abs/1412.6980`, `arXiv:1412.6980`.

[109] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL: `http://arxiv.org/abs/1312.6199`, `arXiv:1312.6199`.

[110] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

Acknowledgments

First of all, I would like to thank Ambarella and Vislab for giving me this opportunity. Thanks to my managers, Stefano, Pietro and Alberto, for their precious help in the difficult process of combining work and study, looking forward to learn new things from you. A thought to my colleagues here in Parma, always kind and happy to offer good advices, I am so proud to be in your own team. And thanks to all the guys in Santa Clara who are an amazing example of dedication, competency and politeness, thanks for always making me feel at home.

A special thanks to my parents, my brother Riccardo and his wife Elena for their support and encouragement. Mom, I hope to see you stop smoking very soon.

Thanks to all the friends with whom I share my best time and experiences. A special thought to my ladies Marika, Sonia, Martina and Sara.

And last...but not last, my favorite beekeeper and engineer Gianluca whose goodness, dedication and intelligence will always be a model for me. Thanks for making my PhD life so amazing.