TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

HABILITATION THESIS

Environment Perception with Vision and Lidar Sensors

Assoc. Prof. Florin Ioan ONIGA, PhD

Computer Science Department

Faculty of Automation and Computer Science

Technical University of Cluj-Napoca

2019

Table of contents

1. Scientific, professional and academic achievements and experience

A summary of the candidate's achievements and experience is presented in this chapter. The candidate received his PhD title in 2011, and the content of this habilitation thesis is based on the candidate's activity since 2011 to present.

Presently, the candidate is an associate professor at the Computer Science Department of the Technical University of Cluj-Napoca (TUCN), Romania. Since 2003, he had the following academic positions within the same department (chronologically): research assistant, assistant professor, lecturer, senior lecturer, and associate professor.

The candidate is a member of the Image Processing and Pattern Recognition Research Center of the Technical University of Cluj-Napoca (group leader Prof. Dr. Eng. Sergiu Nedevschi). After 2011, the candidate's research focus was stereovision and lidar based environment perception, with fundamental and applied research. He developed new models and methods for environment perception, with relevant contributions for obstacle and road detection and representation. More details will be provided in this chapter.

The main achievements are enumerated in the following tables, covering candidate's publications (conference/journal papers), books, and research projects/grants.

Table 1.1. Papers published in journals (3), ISI proceedings conferences (6), and IEEE/Scopus conferences (2)	
1	**F. Oniga**, S. Nedevschi, "A Fast Ransac Based Approach for Computing the Orientation of Obstacles in Traffic Scenes", *2018 IEEE Intelligent Computer Communication and Processing (ICCP)*, pp. 209 - 214, Cluj-Napoca, September, 2018. [IEEE/Scopus]
2	R. Brehar, C. Vancea, **F. Oniga**, M. Negru and S. Nedevschi, "A study of the impact of HOG and LBP based temporal association on far infrared pedestrian detection", *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, 2016, pp. 263-268. [*ISI proceedings*]
3	**F. Oniga**, E. Sarkozi, S. Nedevschi, "Fast obstacle detection using U-disparity maps with stereo vision", *2015 IEEE Intelligent Computer Communication and Processing*, pp. 203-207, Cluj-Napoca, September, 2015. [*ISI proceedings*]
4	**F. Oniga**, S. Prodan, S. Nedevschi, "Traffic light detection on mobile devices", *2015 IEEE Intelligent Computer Communication and Processing*, pp. 287-292, Cluj-Napoca, September, 2015. [*ISI proceedings*]
5	A. Petrovai, A. Costea, **F. Oniga**, S. Nedevschi, "Obstacle detection using stereovision for Android-based mobile devices", *2014 IEEE Intelligent Computer Communication and Processing, Cluj-Napoca*, September 2014, pp. 141-147. [*ISI proceedings*]
6	R. Danescu, A. Ciurte, **F. Oniga**, O. Cristea, P. Dolea, V. Dascal, V. Turcu, L. Mircea, D. Moldovan, "Surveillance of medium and high Earth orbits using large baseline stereovision", in *AIP Conference Proceedings* (Vol. 1634, No. 1, pp. 144-150, November 2014). [*ISI proceedings*]
7	**F. Oniga**, A. Trif, S. Nedevschi, "Stereovision for Obstacle Detection on Smart Mobile Devices: First Results", Proceedings of the *IEEE Intelligent Transportation Systems Conference*, ITSC 2013, The Hague, Netherlands, 6-9 October 2013, pp. 342 – 347. [*ISI proceedings*]
8	S. Nedevschi, V. Popescu, R. Danescu, T. Marita, **F. Oniga**, "Accurate Ego-Vehicle Global Localization at Intersections Through Alignment of Visual Data With Digital Map", *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, No. 2, June 2013, pp. 673-687, ISSN 1524-9050. **WOS IF 2017: 4.051**
9	A. Trif, **F. Oniga**, S. Nedevschi, "Stereovision on Mobile Devices for Obstacle Detection in Low Speed Traffic Scenarios", *Proceedings of the IEEE Intelligent Computer Communication and Processing*, ICCP 2013, Cluj-Napoca, Romania, September 5-7, 2013, pp. 169 - 174. [IEEE/Scopus]

10	R. Danescu, C. Pantilie, **F. Oniga**, S. Nedevschi, “Particle Grid Tracking System for Stereovision Based Obstacle Perception in Driving Environments,” *IEEE Intelligent Transportation Systems Magazine*, vol. 4, No. 1, March 2012, pp. 6-20. **WOS IF 2017: 3.019**
11	R. Danescu, **F. Oniga**, V. Turcu, O. Cristea, “Long Baseline Stereovision for Automatic Detection and Ranging of Moving Objects in the Night Sky,” *Sensors*, vol. 12, No. 10, October 2012, pp. 12940-12963. **WOS IF 2017: 2.475**
12	**F. Oniga**, M. Miron, R. Danescu, S. Nedevschi, “Automatic Recognition of Low Earth Orbit Objects From Image Sequences,” *2011 International Conference on Intelligent Computer Communication and Processing*, ICCP 2011, Cluj-Napoca, September 2011, pp. 335–338. [IEEE/Scopus]

Table 1.2. Published books	
1	**Florin Oniga**, *De la bit la procesor. Introducere în arhitectura calculatoarelor*, Editura U.T. Press, Cluj-Napoca, 2019, ISBN 978-606-737-366-0
2	**Florin Oniga**, Mihai Negru, *Arhitectura Calculatoarelor – Îndrumător de laborator*, Editura U.T. Press, Cluj-Napoca, 2019, ISBN 978-606-737-350-9
3	S. Nedevschi, T. Mariţa, R. Danescu, **F. Oniga**, Raluca Brehar, Ionel Giosan, Silviu Bota, Anca Ciurte, Andrei Vatavu, *Image Processing - Laboratory Guide*, Editura U.T. Press Cluj-Napoca, 2016, ISBN 978-606-737-137-6.
4	M. Negru, **F. Oniga**, S. Nedevschi, *Computer Architecture – Laboratory Guide*, Editura U.T. Press Cluj-Napoca, 2015, ISBN 978-606-737-123-9.
5	S. Nedevschi, T. Marita, R. Danescu, **F. Oniga**, R. Brehar, I. Giosan, C. Vicas, *Procesarea imaginilor. Indrumator de laborator*, Editura U.T. PRESS, Cluj-Napoca, 2013, ISBN 978-973-662-796-5.
6	S. Nedevschi, R. Dănescu, **F. Oniga**, T. Mariţa, *Tehnici de viziune artificială aplicate în conducerea automată a autovehiculelor*, Editura U.T. Press, Cluj-Napoca, 2012, ISBN 978-973-662-787-3.

The scientific impact and recognition of the candidate’s research work, measured by the number of citations, is:

- **1562** citations, h-index = **19**, Google Scholar
- **936** citations (all) / 855 (without self-citations), h-index = **16**, Scopus
- **661** citations / 621 (without self-citations), h-index = **13**, ISI Web of Science

The candidate is a reviewer for the following ISI ranked journals:

- *IEEE Transactions on Intelligent Transportation Systems*
- *IEEE Intelligent Transportation Systems Transactions and Magazine*
- *IEEE Transactions on Vehicular Technology*
- *Sensors (MDPI)*
- *Remote Sensing (MDPI)*
- *Signal Processing: Image Communication (Elsevier)*

Since 2013, the candidate has been a member of the program committee of the *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)* that takes place yearly in Cluj-Napoca, Romania. He also served as a reviewer for many international conferences, in various years.

Research activities and contributions

After 2011, the candidate was involved in more than 11 research projects, within the Image Processing and Pattern Recognition Research Center of the Technical University of Cluj-Napoca. A selection of the most important projects is presented in Table 1.3. A summary of the relevant activities and results obtained by the candidate in these projects is discussed next.

Table 1.3. Research projects (selection)/candidate role

1	UP-DRIVE - Autonomous parking and driving, Horizon 2020 project, 2016-2019, **team member**
2	SEPCA - Percepție vizuală semantică și control integrat pentru sisteme autonome - SEPCA, national project, PNIII-P4-PCCF, 2018-2022, **senior researcher**
3	RB-1 - Measurement of road surface using stereo, international, contractor Robert Bosch GmbH - Chassis Systems Control division, Germany, 2013, 63415 Euro, **project responsible**
4	RB-2 - Identification of 3D lane boundaries based on road limiting infrastructure and surface discontinuities using stereo measurement, international, contractor Robert Bosch GmbH - Chassis Systems Control division, Germany, 2014, 71598 Euro, **project responsible**
5	RB-3 - Measurement of road surface conditions using stereo, international, contractor Robert Bosch GmbH - Chassis Systems Control division, Germany, 2015, 79087 Euro, **project responsible**
6	AMHEOS - Automatic Medium and High Earth Orbit Observation System Based on Stereovision, PNII-PCCA national project, 2012-2016, **team member**
7	CoMoSef - Co-operative Mobility Services of the Future, Eureka European project, 2012-2015, **team member (senior)**
8	SMARTCODRIVE - Cooperative Advanced Driving Assistance System Based on Smart Mobile Platforms and Road Side Units - PNII-PCCA national project, 2012-2015, **team member (senior)**
9	MULTISENS - Multi-scale multi-modal perception of dynamic 3D environments based on the fusion of dense stereo, dense optical flow and visual odometry information, PNII-Idei fundamental research project, 2012-2016, **team member**

The candidate was involved (team member) in the AMHEOS national project (continuation of a previous project, LEOSCOP). The objective of this project was to develop a system for detecting objects on the medium and high Earth orbit, based on a very large baseline stereovision system. The candidate contribution was the detection of satellites based on image processing techniques. The method developed by the candidate was first published at an international conference (*ICCP 2011*, Table 1.1, row 12), and later refined and published as a part of the overall system in a journal paper in *Sensors* (Table 1.1, row 11) and a conference paper indexed in *AIP Conference Proceedings* (Table

1.1, row 6). This method is detailed in chapter 2.

One of the main contributions presented in this thesis is the development of environment perception methods suitable for smart mobile devices, within the SMARTCODRIVE project (team member). The candidate contributed to the development of a 3D stereovision reconstruction system suitable for mobile devices equipped with stereo cameras. The proposed 3D reconstruction approach was published and presented at two international conferences (*ITSC 2013*, *ICCP 2013*, Table 1.1, rows 7 and 9). Another original contribution was the development of a traffic light detection system based on mobile devices, which was published at the *ICCP 2015* conference (Table 1.1, rows 4). All these methods suitable for mobile devices will be detailed in chapter 3.

Within the MULTISENS national project, a system for precise localization in intersections was developed, based on aligning computer vision features with a local extended map. The proposed system was published in the journal *IEEE Transaction on Intelligent Transportation Systems* (Table 1.1, row 8). The candidate's contribution to this system, as a team member of the project, was the curb detection module, as curbs were one of the features used for map alignment. In the same project, the candidate contributed to the development of a fast, low computational method for obstacle detection from the u-v-disparity maps, which was published at an international conference (*ICCP 2015,* Table 1.1, row 3). This second contribution will be detailed in chapter 4.

The candidate was responsible, on behalf of TUCN, and has coordinated 3 research projects for Robert Bosch GmbH, Germany, in the period 2013-2015 (Table 1.3, rows 3-5). The teams involved in these projects teams varied from 2 up to 4 researchers. The topic of research was environment perception from stereovision, suitable for market-deployable driving assistance systems. The achievements within these projects were models and algorithms beyond the state of the art (increased accuracy/precision, and processing times in the range of few milliseconds or lower) for: road surface detection, lateral delimiters detection, small road features detection, such as curbs, speed bumps, lane grooves, and potholes. For the last three features no state of the art solutions were available at the moment of these projects. Beside the coordination activities, the candidate was actively involved in research, and his main contributions will be presented in the next paragraphs (also, these contributions will be detailed in chapter 5). The developed methods had a very low computational complexity and memory requirements, making them suitable for deployment on low power embedded systems.

The main objective of project RB-1 was the development of a method for computing the vertical elevation of the road surface, with an accuracy/precision much greater than the raw stereo measurements. The candidate developed a method to compute the increased accuracy elevation of the road surface by proposing an original elevation interpolation scheme based on the stereo uncertainty model and temporal integration. Based on the improved road surface, an original approach was developed to detect special road related obstacles, such as speed bumps. No stereovision based approach for speed bump detection was available in the literature at the time of this project.

In the second project RB-2, the main objective was to develop models and algorithms to detect the drivable area in front of the ego car, modelled as a 3D lane (a variable-width lane with 3D left and right boundaries). The candidate contribution in this project was the development of a method for detecting 3D lateral delimiters along the ego car trajectory. The method developed was grid based, relied on temporal integration to improve the density and to filter out 3D noise, and relied on a road model that allowed complex vertical variations along the longitudinal ego car axis. The temporal integration of the grid dealt with dynamic obstacles by relying mainly on cell level analysis.

The main novelties brought by the candidate in the third project RB-3 consist of: method for modelling roads with complex lateral-longitudinal geometry, algorithms for the detection of near road interest obstacles: potholes, curbs, and lane grooves. To the best of the candidate's knowledge, except for the detection of curbs, no methods were available in the literature for the detection of the near road interest obstacles, at the time of this project.

A multi- lidar and camera system is used for environment perception in the ongoing Horizon 2020 project UP-DRIVE. As a team member, the candidate research work aims at developing a real-time obstacle detection method with multiple synchronous lidars. The main contributions of the author are the development of a method for road surface detection from the multi-lidar data, and the detection of oriented obstacles using a voxel representation of the multi-lidar 3D measurements. So far, an original low-complexity method for computing the orientation of 3D obstacles was published at an international conference (*ICCP 2018*, Table 1.1, first row). This contribution will be presented in chapter 6.

Within the SMARTCODRIVE and COMOSEF projects, in addition to research activities, the candidate also had the role of team leader (delegated by the project manager), and coordinated some of the research activities of the TUCN team. Also, the candidate maintained the continuous communication and planning with the national funding authorities, consortium members and represented TUCN at multiple consortium meetings (in COMOSEF, there were 21 academic and industrial partners from 7 countries EU and non-EU).

In most of the projects he was involved, the candidate collaborated and guided other younger researchers (bachelor, master and PhD students) in their research work. Guidance was provided in specific research topics, with contributions that were published in international journals (Table 1.1, rows 8 and 10) or international conferences (Table 1.1, rows 2 – 5, 7 and 9).

Teaching activities and contributions

Starting with the assistant professor position occupied in 2003, and up to the current position of associate professor, the candidate has been involved in teaching activities for more than 16 years. The teaching activity consisted of laboratory, project, seminar, and/or lectures for the following courses (bachelor / master programs):

- Computer Architecture – (Romanian) Lecture / Laboratory
- Computer Architecture Fundamentals (Romanian) – Lecture
- Image Processing (Romanian / English) – Laboratory / Project
- Pattern Recognition Systems (Romanian / English) – Laboratory / Project
- Artificial Vision (Romanian) – Seminar.

For the Computer Architecture course (Romanian), the candidate has been teaching laboratory classes since 2004 and the lectures since 2013 to present. The laboratory practical work involves building (VHDL description) a single cycle MIPS processor (Hennessy & Patterson), a pipeline version of MIPS, and extending these processors to allow communication with external sources via a serial communication protocol (UART). All the circuits described in VHDL are tested on Digilent Development Boards (currently on Basys 3 – Artix 7 boards). The candidate contributed significantly to a computer architecture laboratory guide with 12 lab tutorials that was published in two versions (Romanian and English, Table 1.2, rows 2 and 4). Also, the candidate published a book (Table 1.2, row 1), in Romanian, that presents fundamental aspects of computer architecture in a fast-forward, example guided manner.

The candidate is also co-author of a laboratory guide (Romanian and English, Table 1.2, rows 3 and 5) for the Image Processing course. Relevant research results of our research center (IPPRRC) were published in a book (Table 1.2, row 6) that presents advanced techniques based on stereovision for driving assistance systems. The candidate has co-authored this book.

For the semester projects of Image Processing and Pattern Recognition Systems courses, the candidate has supervised and guided students to fulfill their individual projects with various topics: fingerprint image analysis, region growing, image morphing, stereo matching and 3D reconstruction, noise filtering, texture recognition, face detection and recognition, 3D obstacle detection etc.

The candidate has supervised more than 40 license (bachelor) and dissertation (master) theses, which were successfully defended by their student authors. Currently, he supervises 6 license theses and 3 dissertations that should be finalized and defended this scholar year (2018-2019, summer session). Most of these theses were in the field of computer vision (same as this habilitation thesis), and involved research work (some of the results were published at international conferences).

2. Detection and recognition of low Earth orbit objects from astronomical images

The candidate was involved (team member) in the AMHEOS national project (continuation of a previous project, LEOSCOP). The objective of this project was to develop a system for detecting objects on the medium and high Earth orbit, based on a very large baseline stereovision system. The candidate contribution was the detection of satellites based on image processing techniques. The method developed by the candidate was first published at an international conference (*ICCP 2011*, Table 1.1, row 12), and later refined and published as a part of the overall system in a journal paper in *Sensors* (Table 1.1, row 11) and a conference paper indexed in *AIP Conference Proceedings* (Table 1.1, row 6).

The contribution presented in this chapter is an approach for automatic detection and recognition of interest objects from the earth orbit that are visible in astronomical images. The main interest objects are satellites, but various objects, such as planes, stars, can be identified. The proposed technique starts with background estimation and removal. Then, potential objects are identified by labeling the background free image. Relevant features are computed for each individual object and used for classification, which is performed with a decision tree.

2.1. Introduction

In the context of increasing the number of artificial satellites that are launched every year, detecting satellites coordinates becomes more and more important, in order to avoid collisions and to keep the satellites on their orbits. The first step towards measuring the distance, and thus the orbit of objects is to detect their signature in the image. A defining characteristic of these objects is the linear nature of their image trace, which discriminates them from the point-like stars. The line segment trajectory is caused by the speed of the object, combined with the high exposure time of the imaging device. A typical object has the speed of around 8000 m/s, circling the Earth at an altitude between 300 and 2000 km. If we assume an average height of 600 km, the object's speed translates in an angular velocity of 0.74 degrees per second. At a typical exposure time of 5 seconds, for a camera equipped with a 70 degrees field of view lens, a LEO object travels 5.3% of the diagonal length of the image. For an image having the diagonal of 2826 pixels, the LEO object travels approximately 150 pixels. The length of the image line segment is related to the distance of the LEO object to the ground, increasing as the object is closer to Earth.

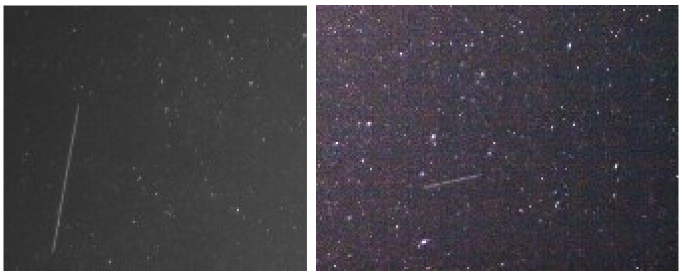


Fig. 2.1. Orbit objects (streaks) in a 5 seconds exposure image

Astronomical images contain a multitude of objects like stars, planes, outliers that might make it harder to detect satellite streaks, as well as to distinguish between satellites and the other astronomical objects. The proper approach is to detect and classify all of the above mentioned objects into their corresponding classes.

The task of detecting satellites streaks in astronomical images is quite new, and as a consequence, little amount of bibliography describes this area of research.

A first approach is the one proposed by Brad Wallace in [1]. The paper starts with describing the sensor characteristics in order to know precisely the main features of an astronomical image. An algorithm for streaks detection is proposed. Image-plane artifacts and large scale background are removed. A noise floor is determined for each image pixel, and objects above this floor are identified. Image moments are used to classify objects as point-like or streak-like.

Besides this, some of the most representative papers ([2], [3], [4]) have been published by Levesque et al. Their work is summed up in [2]. The approach is based on a series of steps, the most important ones being background estimation and removal, star detection and removal, followed by an iterative matched filter used to detect streaks. Measures for false positives rejection are also included. All these lead to the capability of detecting very faint objects. Iterative background removal is used. It assumes that the background is smooth, while the other objects are sharp. The average pixel value is corrupted by the presence of bright objects. These objects are clipped, so after several iterations very accurate background estimation is obtained. Stars are detected and erased in a 3-step process, starting with saturated stars, then with bright stars and finally with faint stars. Next, the technique for detecting satellite streaks is based on matched filter techniques, which are expected to help achieve the maximum sensitivity when detecting an object with a known shape. Their main drawback is the generation of false alarms by the other objects.

2.2. Overview of the proposed algorithm

The main differences between our approach and existing methods are the background removal technique and the classification step, which is able to cope with more object types. The proposed algorithm consists of the following steps:

1. Background detection and removal – an exponential moving average technique
2. Objects detection – based on labeling and feature computation
3. Classification – using a decision tree.

Our proposal for background removal is based on estimating the background image with an exponential moving average technique. The local statistics background removal algorithm proposed by Levesque [4] will be used as reference for comparison. Our proposed method provides similar results regarding streaks detection (even better regarding the removal of stars) but has a much lower computational complexity.

The classification part represents another contribution of this algorithm. A decision tree, trained with manually labeled data, is used for the classification of objects based on their features. The star/satellite discrimination presented by current methods is not enough because astronomical images can contain streak-like objects that are not satellites: planes, condensation trails, clouds etc. Our method provides a more general output that deals with more object types.

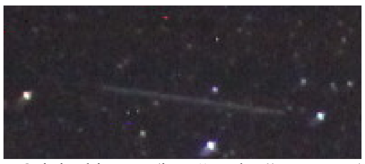
2.3. Background Removal

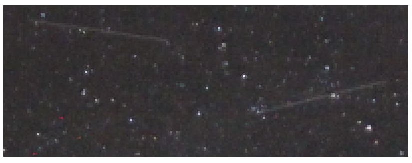
Background estimation and removal is the first step of the algorithm. Thus, it directly influences the input for the ones that follow and, consequently, the correctness of the obtained results.

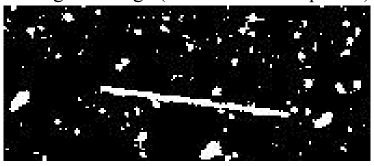
When looking for the best approach for background removal, both the accuracy and the speed of the technique should be taken into account. Another important aspect is the influence of the Earth's rotation on the background. Considering that the exposure time is about 5 seconds, stars will also present a certain displacement (several pixels) between two frames. However, if the camera is mounted on an automated star tracker device then the stars will have no motion between frames. Our test images were mixed. Some of the sequences were acquired with a tracker device, others without.

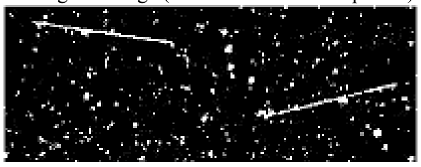
The exponential moving average technique proposed by us performs optimally for images acquired with a tracker device (stars will be removed also), but it also works with standard images (because stars do not move significantly between two frames). We will compare our approach with the background estimation and removal proposed by Levesque, referred later in text as the local statistics approach. The local statistics approach represents an adjustment of Levesque's local statistics background removal algorithm that has been presented in [4].

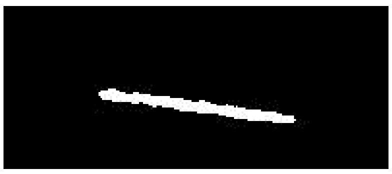
The approach that we propose for background removal gives similar results for satellite streaks. It provides improved results if a tracker device is used, by removing almost all of the stars. If the images are acquired in a sequence by a tracker device-mounted camera, then the background and the stars do not move. During an entire sequence of images acquired by a tracker device-mounted camera, the only moving objects are satellites, planes, and some outliers (like clouds, condensation trails), making the object detection and classification steps much easier. If the camera is not mounted on a tracker device, stars are also emphasized by the moving-average background and some parts of them are detected as outliers.


a. Original image (in a "tracker" sequence)


d. Original image (in a "non-tracker" sequence)


b. Background-free image obtained with the local-statistics based approach


e. Background-free image obtained with the local statistics based approach


c. Background-free image obtained with the proposed moving-average technique

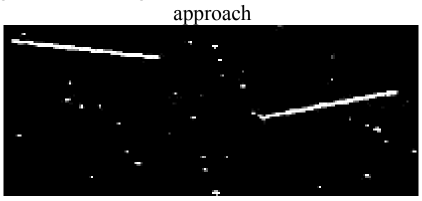

f. Background-free image obtained with the proposed moving-average technique

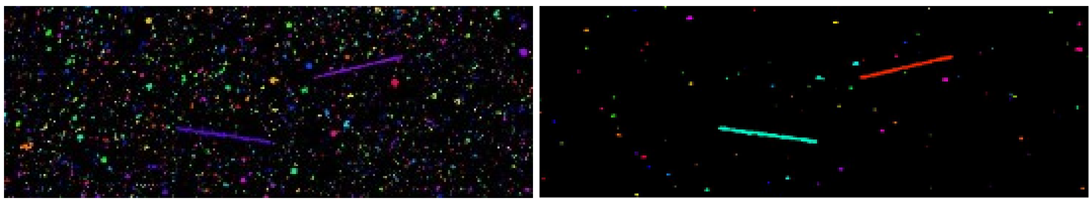
Fig. 2.2. Results of the background removal step

Fig. 2.2 presents the background-free images resulted after processing the original images with both of the approaches for background removal (reference and proposed). The results are similar, except the fact that the moving average approach removes most of the stars, as they are considered to be part of the background. Also, for the image in the "tracker" sequence (with the tracker device), the background-free image contains very few stars (none visible around the satellite).

2.4. Object Detection

2.4.1. Object Labeling

The algorithm used for labeling detects connected patches of interest pixels in the background free image. This algorithm works on the images obtained with both of the approaches presented in the previous subchapter. Fig. 2.3 shows the images obtained after the labeling algorithm is applied on background-free images. Each binary object has a specific color assigned after running the labeling algorithm.



a. b.

Fig. 2.3. a. Labeled objects on the background-free image obtained with the local statistics approach, b. Labeled objects on the background-free image with the moving average background removal approach

2.4.2. Object Features Computation

The labeled binary objects detected in the previous step must be classified as satellite streaks, planes or other objects. In order to perform that classification, we will approximate every binary object by an ellipse, and we'll compute the geometrical properties of this ellipse. The initially considered properties for the binary objects were: Area, Major Axis Length, Minor Axis Length, Eccentricity, Equivalent Diameter, Perimeter, Solidity and standard deviation of the pixel values. Some of these properties will not be used in the final classification, since they do not determine a clear separation between the classes, as we will show in the following section.

The remaining properties for the final classifier are: Area, Major Axis Length (L_{MAX}), Minor Axis Length (L_{MIN}) and Eccentricity (e).

2.5. Training of the Decision Tree

In order to build a decision tree, a training and testing database needs to be created by manually labeling each object. To ease this work, an intuitive classifier has been implemented, to obtain a database containing all the relevant objects and their classes, with some misclassifications that were corrected manually. The intuitive classifier takes into account the area, eccentricity and minor and major axis lengths of the objects. The area and eccentricity separate the stars from other types of objects, and the last two properties differentiate the satellites from planes. In this way, the actual classification was prepared. In order to have a coherent dataset for training, the database resulted from the intuitive classifier (having all the previously mentioned properties for each significant object) was manually verified and corrected. Next, a classifier was automatically generated by Weka, using J48 trees.

229 images contributed, each, with an average of 5 objects to the training set. This was necessary in order to establish a balance between the satellites, planes and other objects (like remaining stars and outliers). Small stars were discarded from the training set (area smaller than a threshold).

2.6. Results

In this subchapter, the classification results for both the local statistics and our proposal for background removal will be presented.

For the local statistics-based approach, the confusion matrix was the following:

Satellite	Other	Plane	Classified as
148	2	3	Satellite
3	1175	0	Other
1	0	5	Plane

The decision tree is the following (the area related condition was added to the result provided by Weka):

```
Area <=85: other
Area > 85
|   Eccentricity <= 0.99: other
|   Eccentricity > 0.99
|   |   MinorAxisLength <= 11.106: satellite
|   |   MinorAxisLength > 11.106: plane
```

For the moving-average approach, the confusion matrix is the following:

Satellite	Other	Plane	Classified as
146	8	1	Satellite
0	523	0	Other
0	2	9	Plane

The following decision tree was generated by Weka (the area related condition was added to the Weka result):

```
Area <=85: other
Area > 85
| Eccentricity <= 0.99
| |  Eccentricity <= 0.935: other
| |  Eccentricity > 0.935
| |  |  MajorAxisLength <= 47.543: other
| |  |  MajorAxisLength > 47.543: satellite
| Eccentricity > 0.99
| |  MajorAxisLength <= 200.337: satellite
| |  MajorAxisLength > 200.337
| |  |  Eccentricity <= 0.998: plane
| |  |  Eccentricity > 0.998
| |  |  |  MajorAxisLength <= 294.676: satellite
| |  |  |  MajorAxisLength > 294.676: plane
```

Considering the analysis performed by Weka, the only selected attributes that need to be computed are area, eccentricity, minor and major axes length.

As observed, the number of objects detected in the two approaches differs significantly, but the classification performance is similar. This behavior is caused by the fact that, for images acquired with tracker device-mounted cameras, the stars are considered background, and when tracker devices are not used, there are only parts of stars remaining. Those parts, being smaller, are not taken into account when building the training dataset, because of the imposed area constraint.

Sample results are given in the following figures. For both of the background removal approaches, there are situations when faint satellites are detected in an image, but not in the previous or following ones. Such an example is presented in Fig. 2.6.

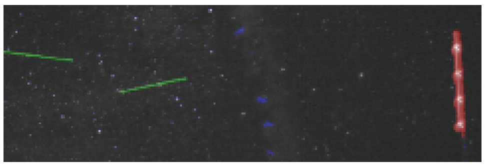


Fig. 2.4. Example of LEO objects classification: green – satellite, blue – outlier (other), red – plane

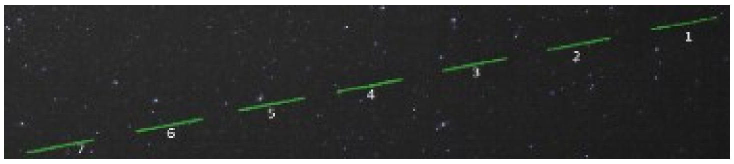


Fig. 2.5. Example of satellite detected in a sequence of images (the detection is overimposed on the first image from the sequence)

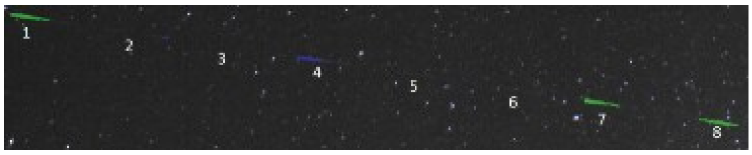


Fig. 2.6. Satellite streaks detection in eight successive frames: very faint or not visible streaks are not detected

2.7. Conclusions

The previously presented confusion matrixes prove that the classification results are good, with a weighted ROC Area of 0.987 for the local statistics-based approach, and of 0.965 for the sequence-based one. The classification performance is slightly lower when the moving average technique is used, but this is compensated by the much lower computational complexity of the method (several times faster). Furthermore, it manages to provide no false positives (stars classified as satellites), and the results would improve if only sequences with tracker devices are used.

3. Driving assistance based on image processing using mobile devices

One of the main contributions presented in this thesis is the development of environment perception methods suitable for smart mobile devices, within the SMARTCODRIVE project (team member). The candidate contributed to the development of a 3D stereovision reconstruction system suitable for mobile devices equipped with stereo cameras. Another original contribution was the development of a traffic light detection system based on mobile devices. All these methods suitable for mobile devices will be detailed in this chapter.

3.1. Stereovision on mobile devices

In 2012, market deployment of smart mobile devices that feature synchronous stereo image acquisition has raised the opportunity to use such devices for real-time 3D environment reconstruction by stereovision. In this research, we investigate a stereovision approach that can run in real-time on smart mobile devices, and we evaluate its potential for developing driving assistance functions. The stereo approach is a sparse approach: edges are detected in the left image, correspondent right image points are determined using area-based matching, and each left-right pair of points is mapped in 3D by triangulation. Our experiments have proved that despite the limitations imposed by the mobile device, both reconstruction accuracy at short-medium distances and real-time processing can be achieved. Thus, developing driving assistance functions with such devices is possible for low vehicle speeds / short range scenarios, which often occur in urban environments. The proposed 3D reconstruction system was published and presented at two international conferences (*ITSC 2013* and *ICCP 2013*, Table 1.1, rows 7, 9).

3.1.1. Introduction and related work

Depth information about the objects in the environment is of a great importance in driving assistance applications, and more specifically in obstacle detection. Moreover, the algorithm needs to be efficiently implemented in order to meet the high speed and low power consumption requirements of mobile applications. In order to achieve widespread deployment of a stereovision based system, a tradeoff must be made between the accuracy and maximum depth performance of the stereo-reconstruction algorithm and the required hardware complexity (power, stereo system size, etc.) of the cameras and the processing unit.

Many stereovision approaches have been proposed in the last decade, and next, we will briefly revisit some of the most relevant ones. In [5] Hirschmüller et al. present a real-time correlation-based stereovision system, by analyzing and comparing the results of various matching cost functions. They propose two methods of reducing the number of matching errors and also a solution to improve matching at object borders. In [6] an edge-based stereo-reconstruction system is presented, with focus on obstacle detection at far distances with high accuracy. A dense stereo solution with sub-pixel accuracy is described in [7], based on the Semi-Global Matching (SGM) method [8] and using the Census transform as the matching metric. The SGM was introduced in 2005 by Hirschmüller, and it was proved to provide accurate stereo-matching results in real time, by searching for pixel correspondences based on Mutual Information and also approximating a global cost. In [9] Stefano proposes an area-based matching approach, the Single-Matching Phase (SMP) local algorithm, which eliminates redundant differences and sums while computing the Sum of Absolute Differences matching cost. Even though extensive work has been done in the stereovision domain, none of these previously mentioned solutions were intended to run on smart mobile devices due to their increased computational complexity.

Recently some research has been directed towards the implementation on smart mobile devices of some computer vision algorithms, 3D reconstruction being an example. A very good analysis of the advantages and limitations of mobile devices regarding computer vision and augmented reality applications is presented in [10]. In [11], Langguth and Goesele describe a solution for sparse 3D reconstruction of objects using structure from motion. Multiple pictures of the scene are captured from different view angles using the camera of a mobile phone, the user being guided by the application to move the device in the best position for a better reconstruction. In [12] another approach is presented, in which two smartphones are used to create a master-slave photometric stereo system. Thus, both devices capture images, but the slave device also uses its flash to illuminate the scene, while the master applies the photometric reconstruction algorithm. In [13], Pan et al. describe their solution of generating a 3D model of the scene from panoramic images in real time. An application for fast 3D reconstruction of house plans is described in [14].

As already noted in [10] and [13], there are very few mobile device applications that perform the entire processing on the device. The majority of the solutions are either client-server applications, in which most of the processing is done on a server [10], or they perform an offline reconstruction from a set of previously captured images, without taking into consideration the computational requirements [13].

In this work we investigate whether such a smart device, equipped with a stereo camera, can be used as both acquisition and processing platform for stereovision based driving assistance. We propose a solution for edge-based stereo-reconstruction tailored for mobile devices, and, after the experimental evaluation of the system, we discuss the possibility of using such devices for driving assistance.

3.1.2. Algorithm Overview

The major steps of the stereovision algorithm are depicted in Fig. 3.1 and are further detailed in what follows. The algorithm is more or less based on the standard steps, with an additional measure for reducing the search space for the right image correspondent.

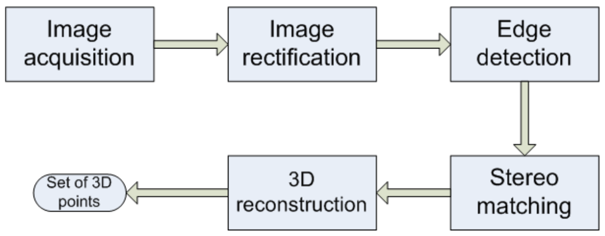


Fig. 3.1. Algorithm overview

Although the built-in cameras of the device are almost canonical, rectification is needed on the acquired images in order to simplify the correspondence searching step and to reduce the amount of computations required for 3D reconstruction, thus improving the processing speed. We chose to implement the rectification approach presented in [15].

The next step consists of selecting the relevant left image features for 3D reconstruction. Considering the need for low computational complexity, we use an idea already present in literature, which is to select edges as relevant features. Extracting edges in the left image was done using the Canny edge detection algorithm. The left image will be then used as reference in the correspondence searching step. For every edge point in the left image, its homologous point in the right image is searched on the horizontal epipolar line, and then an interpolation function is applied on the matching costs in order to achieve sub-pixel accuracy. Stereo matching will be described in more detail in the next section.

The last step of the algorithm is represented by the 3D reconstruction operation. Because the images are rectified, we can use the well-known equations for 3D reconstruction in a canonical configuration, which are widely available in the literature [16].

3.1.3. Stereo Matching

As previous research has shown, stereo matching is the most important and computationally intensive step of the stereo-reconstruction algorithm. Therefore, a series of constraints need to be applied in order to reduce the search space and also ensure that the number of false matches is reduced without affecting the number of good matches. A pseudo-code of the stereo matching function is the following:

```
Function Stereo-matching

1:  for each edge-point in the left image do
2:      compute the gradient magnitude $m_L$
3:      set gradient threshold $t \leftarrow m_L/2$
4:      for each point p in the disparity range in the right image do
5:          compute the gradient magnitude $m_R$
6:          if $m_R > t$ then
7:              compute $SAD_p$
8:              add $SAD_p$ to SAD-list
9:              if $SAD_p < SAD_{min}$ then $SAD_{min} \leftarrow SAD_p$ end if
10:         end if
11:     end for
12:     apply constraint for repetitive patterns
13:     if not repetitive pattern then
14:         apply sub-pixel interpolation
15:     end if
16:     add current match to matches-list
17: end for
18: return matches-list
```

The presented pseudo-code is explained in more detail in what follows.

3.1.3.1. The matching cost function

Many similarity metrics exist in literature: the normalized cross-correlation (NCC), the sum of squared differences (SSD), the sum of absolute differences (SAD) and so on. According to [5], SAD is the fastest and also provides better results compared to SSD or NCC. Therefore, we considered the SAD as being more appropriate for a fast implementation. This function is applied on a window of size *n* x *n* around the pixels to be matched. The best matching pixel in the right image is chosen to be the one which minimizes this cost.

The choice of the window size is very important: a small window (e.g. 3x3) implies a smaller number of computations for the matching cost, but it will yield less accurate results and a lot of false matches, whereas a larger window will affect the processing speed but will provide more accurate results. We use a 7x7 window for matching.

3.1.3.2. Constraints

The left image features are edge points, therefore exhibiting a significant value of the gradient magnitude. The right image correspondent should present a similar gradient value, slightly different (if any) due to contrast differences between the stereo pair. Thus a condition based on the gradient magnitude is applied. We first compute the gradient in the x and y directions in the left image. Then we compute the gradient magnitude m_L using the Manhattan metric (faster than the Euclidian metric). We then set a threshold value at half of the previously computed magnitude and we choose to match only those points in the right image which have the gradient magnitude greater than this threshold. This constraint is graphically represented in Fig. 2 (explanations in the caption).

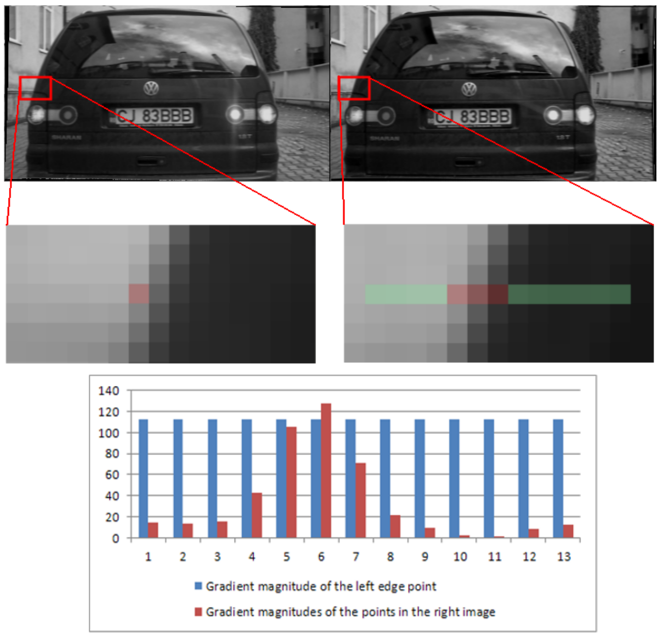


Fig. 3.2. Gradient-based constraint for search space reduction. The red point in the middle-left picture is the detected edge point. In the middle-right image, the green points represent the search space for the correspondent, and the red points are those pixels that pass the gradient magnitude constraint. The diagram depicts a comparison between the gradient magnitude of the left edge point and the gradient magnitudes of all the points in the disparity search range in the right image. Only pixels 5, 6 and 7 have a gradient greater than the threshold

The next constraint tries to eliminate the ambiguous matches of repetitive patterns. We compare the matching cost of the best match with all the other costs in the disparity range. If the global minimum is not smaller with at least 20% than all local minimums, the match is rejected.

3.1.3.3. Sub-pixel accuracy

Because the matching point in the right image might not always be located on an integer pixel position, but rather between two pixels, we perform a sub-pixel interpolation to achieve a better accuracy. We chose to implement the parabola interpolation and the symmetric “V” [17]. Our experiments showed that the two interpolation methods provide almost similar results on the short-medium depth range perceived by the system.

3.1.4. Experimental Results

The application was tested on an LG V900 Optimus Pad device, which has an Nvidia Tegra 2 chipset including a dual-core ARM Cortex A9 CPU. The baseline of the stereo camera system is 4.5 cm.

In order to test the reconstruction accuracy of the system, we acquired a sequence of left-right pairs of images of a car located at measured distances in the range 2 m – 9 m.

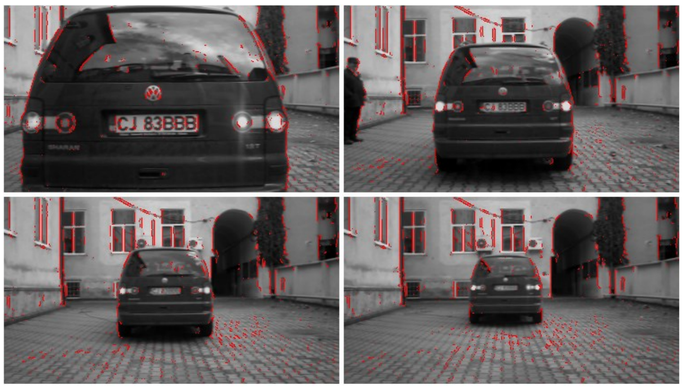


Fig. 3.3. The re-projection result of the 3D points onto the left image of the stereo-pair: top-left – the car is at 2 m; top-right – the car is at 4 m; bottom-left – the car is at 6 m; bottom-right – the car is at 8 m.

In order to test the depth estimation accuracy, we selected only the points that lay on the car and computed their mean and standard deviation on the z-axis. Some results can be seen in Table 3.1. We observe that up to a distance of 6 m the mean distance is accurately determined, at 7 m we have an error of only 30 cm, at 8 m the error is 50 cm and at 9 m the error increases to 1.3 m. The reconstruction results and the images used in our experiments can also be seen in Fig. 3.3.

Table 3.1. Accuracy/precision of depth estimation when using images of size 384x216

Measured distance (m)	**Mean (mm)**	**Standard deviation (mm)**	**Number of edge points**
2	2051	94	769
4	3968	375	360
6	5942	828	194
7	6778	996	204
8	7530	508	131
9	7717	626	90

In Table 3.1 the standard deviation of the 3D points reconstructed in our experiments is also represented. We can see that as the distance grows, the points are more spread out on the z-axis. This phenomenon is visible in Fig. 3.4, where at 2 m the reconstructed points are very close to the mean, while at 8 m the points are in the range [6.5 m, 9 m]. However, when the distance gets larger, the standard deviation becomes less reliable due to the decreasing number of edge points.

There are more causes for these deviations from the real distance. First of all, the small resolution of the images does not allow an accurate reconstruction at far distances. The size of the images is limited by the device and operating system.

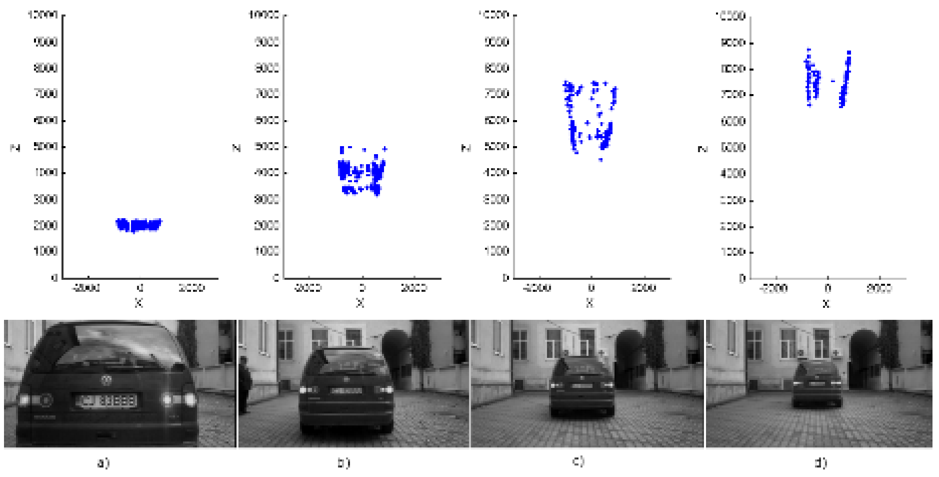


Fig. 3.4. Depth estimation results (top) and the corresponding 384x216-sized left image from the stereo-pair used for reconstruction (bottom): a) the car is at 2 m; b) the car is at 4 m; c) the car is at 6 m; d) the car is at 8 m. Values on each axis represent millimeters

Even though the stereo imager of the device has 5 Mega Pixel cameras, so far the Camera API provides only a small set of supported sizes for the acquired images. The sizes supported by the camera are the following: 720x408, 720x480, 720x576 and 768x432. We chose the last one, as it has the largest horizontal size, relevant for stereo-matching. However, in the dual-camera mode, the two left and right images are scaled down horizontally so that both of them fit in the same picture of size 768x432. As a consequence, the images appear elongated on the y-axis and a further vertical resizing is necessary to bring them back to the natural aspect ratio. Individually, the left and right images have a resolution of 384x216 pixels (the equivalent focal length of 404 pixels).

To test further the potential of such a system, we captured a set of larger images using a different function of the Camera API, which allows capturing pictures having a combined size of 1200x680. However, for now, this function does not allow sequence mode acquisition. We applied the same reconstruction algorithm on these larger images, which, after vertical resizing, have a size of 1200x340 pixels (each image from the pair has a size of 600x340 pixels). As expected, the reconstruction results are much better than in the case of smaller resolution images (Fig. 3.5 and Table 3.2). The depth can be estimated with small errors up to greater distances. But more importantly, the reconstructed points are not so spread out around the mean, as it can be observed from the standard deviation. For example, when the object is at 6 m, the standard deviation is only 12.2 cm compared to a deviation of 82.8 cm in the case of the smaller resolution images. Moreover, the standard deviation when the object is at 10 m is similar to the one obtained when the object is at 5 m, with the small image configuration.

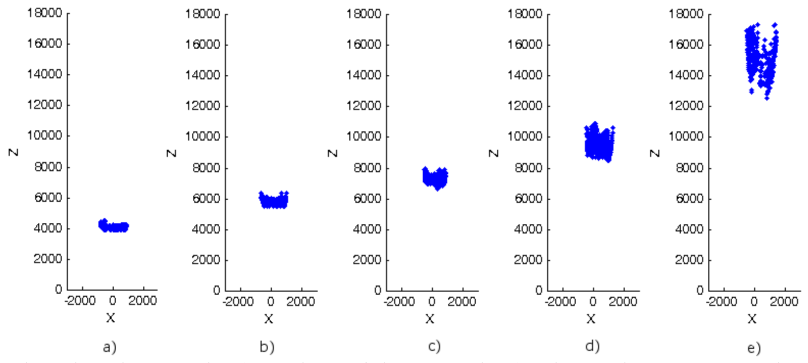


Fig. 3.5. Depth estimation results (top view of the 3D points) when using 600x340-sized images and the car is at: a) 4 m; b) 6 m; c) 8 m; d) 10 m; e) 15 m. Values on each axis represent millimeters

Table 3.2. Accuracy of depth estimation when using images of size 600x340

Measured distance (m)	**Mean (mm)**	**Standard deviation (mm)**	**Number of edge points**
4	4053	78	2645
6	5818	122	1197
8	7351	198	765
10	9493	465	801
15	15102	1009	337

Concerning the time performance of the application, we managed to achieve an average speed of 6.5 frames per second on 384x216 grayscale images when using a window of 7x7 in the stereo-matching function. When the camera API will allow acquiring larger stereo images in sequence mode, multi-resolution stereo matching can be used to keep the processing time low, while benefiting from the better accuracy of larger images. Currently the implementation of the application is single-threaded, but because recently most smart mobile devices feature multi-core processors, as a future improvement, it can be split on multiple threads.

3.1.5. Discussion

Several conclusions can be drawn from the conducted experiments and evaluation. The current sequence aqusition mode supported by the camera API (384x216 stereo images) limits the maximum depth reliable for obstacle detection at about 7-8 meters. Without the sequence mode, when using 600x340 stereo images, the reliable depth is up to 15 meters. By reliable depth we are refering to the maximum depth where, through feature grouping techniques applied on the 3D points (and the image features), the obstacle can be located as a cuboid.

The question still remains: Does this limited range/processing time allow the development of driving assistance functions? This obviously depends on the targeted scenarios, and the main issue is related to the stopping distance of the ego vehicle. With the current maximum range for online processing, the system can perceive obstacles up to 7-8 meters. The stopping distance is given by the reaction time of the driver (usually around 1 second for undistracted drivers) and the braking distance. The braking and stopping distances, assuming a dry flat road, and average tires (friction coefficient 0.7), are shown in the following table, for speeds typical to urban environments.

Table 3.3. Braking and stopping distance for typical urban speeds

Speed km/h	5	10	15	20	25	30
Speed m/s	1.3	2.7	4.1	5.5	6.9	8.3
Brake Dist. m	0.14	0.5	1.2	2.2	3.5	5.0
Stop Dist. m	1.5	3.3	5.4	7.8	10.4	13.3

In low urban speeds (5-15 km/h), warning the driver can help avoiding accidents involving pedestrians or vehicles. Such low speed scenarios are quite often in modern crowded urban areas, where a lot of stop-and-go maneuvers are performed, or the average speed is low on some road sectors due to the high traffic volume. Even at medium urban speeds (20-30 km/h), automatic warnings might help the driver to mitigate potential accidents. As the reliable depth will likely increase with the future software API, so will the range of speeds where the system can be used.

3.2. Traffic light detection on mobile devices

The opportunity of using smart mobile devices (e.g. smartphones) for traffic light detection, on-board of vehicles, was investigated in this research work. The focus was on developing an algorithm suitable for the processing capabilities of a mobile device, with promising detection capabilities. Candidate points for traffic lights are extracted using Gaussian distributions for hue, saturation and lightness with parameters learned from training images. Candidate regions are built by clustering connected candidate points. These regions are then filtered by using shape information. For each candidate region we perform a fast circularity check based on the image gradient in the area. Finally, we evaluate the algorithm using manually labeled images. The detection rates (80%-85%) show that, with future refinement, the system has applications in real traffic scenarios. This contribution was published at the *ICCP 2015* conference (Table 1.1, rows 4).

3.2.1. Introduction and related work

The traffic scenes today involve drivers paying constant attention to multiple information sources, simultaneously. In such conditions it is easy for a driver to miss certain traffic relevant information. A frequent error, due to driver's distraction, is missing traffic light information at an intersection. Such a mistake can have potentially serious consequences, including both material and human casualties [18]. Therefore, assisting distracted drivers through automatic traffic light detection and warning can help avoiding or, at least, mitigating imminent collisions.

Since the introduction of Driving Assistance Systems (DAS), research on traffic light detection has become of a greater interest among researchers. The tasks of detection and classification in the image space are usually time consuming, and require a lot of processing power. In the recent years, due to the increasing computing power and memory capacity, smart mobile devices became a suitable candidate for deploying such tasks. In addition, such devices also have an integrated video camera, allowing high frame rate image acquisition, and complex user interaction (for visualization and inputs).

In [19] the authors propose a system based on modeling Gaussian distributions of hue and saturation, with parameters extracted from training images. First a thresholding is performed based on the normal distributions. Then, morphological operations and aggregation of results from the previous frames is used to filter out false candidates. Authors of [20] propose a system that extracts candidate pixels using a HSL space filtering followed by a genetic approach for ellipse detection. This approach can handle the problems of shape deformation and partial occlusions very well due to the genetic algorithm. In [21] a template matching technique using cross correlation is presented. They use a color threshold segmentation method paired with several types of color and shape filtering to select candidates for the template matching algorithm.

In [22] a comparison is presented between image processing techniques and learning approaches for traffic light detection. For the image processing method they use a similar approach with [21] by using template matching on a set of previously extracted candidates. Spot light detection based on top-hat morphological operation is used in order to extract these candidates. For the learning approach, they experimented with several learning based methods applied on grayscale images such as: Multi-Layer Perceptron, Genetic and AdaBoost. They concluded that the image processing technique used by them was superior to the learning approaches both in precision and computation time.

Continuously adaptive mean shift (CAMSHIFT) is used in [23]. This target tracking algorithm reduces the number of false negatives and minimizes the possibility of a tracking loss. As a pre-processing step to the tracking algorithm, the authors also propose a method of extracting candidate regions based on color threshold segmentation and morphological operations. They experiment with a real vehicle to validate the performance of their approach. The setup used by them is composed of a vehicle mounted camera connected to a computer.

Color threshold segmentation paired with morphological operations is a common step used for extracting candidate regions, like in [19] - [21]. The conversion from RGB to another color space like HSV or HSL is a required pre-processing step for all methods based on color segmentation.

Other papers, like [24], focus on detecting only the suspended traffic lights. Their method relies on colors and features such as the area of traffic lamps or the black area of traffic lights. Their work also mentioned a technique for distance calculation, but relies on traffic lights having a standard height.

Methods reviewed so far were not intended to run on mobile devices. Most of the methods here either run on a computer connected to a vehicle mounted camera, or run offline on a set of previously captured images. For example, one of the fastest methods is [21], which runs in about 40ms on a standard PC. Even if this processing time is low, a mobile device application has to perform several additional tasks: image acquisition, high level reasoning, and user interface for feedback/warnings. Thus, developing a faster traffic light detection algorithm is required.

There are few approaches intended to run fully on a mobile device. A similar problem is tackled in [25], but not for driving assistance, where a traffic light detection system for the visually impaired pedestrians is described. The authors propose background color examination and temporal verification of successive frames for better results.

The contribution described in this sub-chapter is a method for detecting traffic lights using a vision system based on smart mobile devices, placed onboard of vehicles. A smart mobile device will be used for data acquisition using its camera, image processing for detection of traffic lights, and providing real-time feedback to the driver using visual and audio feedback. Due to the low power requirements imposed by using a mobile device, our algorithm must be efficient in terms of computation time in order to provide real-time feedback.

3.2.2. Traffic light detection algorithm

3.2.2.1. Proposed algorithm overview

The proposed algorithm consists of the following processing pipeline (having as input RGB images, and as output the set of detected traffic lights):

1. RGB to HSL color space conversion
2. Color thresholding
3. Connected components extraction
4. Geometric feature based filtering
5. Additional circularity constraint filtering

After image acquisition, the images are converted to the HSL space. The color thresholding step is done based on training images by computing the normal distributions for hue and saturation of the three possible traffic light states. The connected components extraction step labels and extracts information about each connected pixel region from the threshold image. Each component is then filtered based on its geometric features. In the last step, based on the occurrence of edges around the candidate regions, only those regions that resemble a circle are validated.

The color thresholding method is similar to the one used in most of the state of the art methods. The method proposed in [19] makes use of morphological operations in order to filter out small false detected regions. In our proposed algorithm, in order to save processing time, we replaced this step by imposing a threshold upon the area property of the connected components. More specifically, components with area under a certain threshold are discarded without the use of a costly opening morphological operation.

In contrast with the common validation method of template matching used in existing approaches, our approach uses a fast circularity check. Compared with the classical top hat approach [22], or the genetic approach for ellipse detection [20], our method is less exact but much faster. Beside the circularity check, an additional condition is applied (also fast), using the property that the traffic light bloc is dark (close to black) and only one light is on. This additional criteria is meant as a fast alternative to the more costly template matching used in the literature [21].

In order to minimize the false detections on round shaped car tail lights we'll use the assumption that most of these tail lights are located below the horizon line in the image. In contrast to these false candidates, traffic lights are usually placed above this horizon line. A simple calibration needs to be performed before the system is operational, in order to obtain an estimation for the horizon line. This calibration is done manually, through the mobile device user interface, before the system becomes fully operational.

Initially, we experimented with several sensors available on the mobile devices. We found the accelerometer and gyroscope sensor to be the best candidates for providing pitch estimation. However both approaches posed problems in terms of accuracy. The method using the accelerometer is greatly affected by linear acceleration while gyroscope sensor was problematic because obtaining the pitch implies some integration operations which proved to accumulate errors over time. Further research can be conducted on this topic in order to obtain automated horizon line detection. Also, using a visual based method (based on the vanishing point of the lane markings) can help but it will introduce additional processing time.

3.2.2.2. HSL space conversion

Conversion to the HSL space uses complex computations applied on each pixel in the image. Because of the real-time requirements of our application the computation time needs to be reduced as much as possible and one approach to this problem is to use look up tables. This table represents a mapping from RGB values to HSL values.

3.2.2.3. Color thresholding method based on Gaussian distributions

As noted in [26], color image segmentation methods include: histogram thresholding, fuzzy techniques, neural networks, edge detection, region-based methods and so on. Because of our computational constraints, color image segmentation based on thresholding is the most appropriate approach for this case because it provides good performance in terms of computation time.

Threshold values for color image segmentation are selected based on a set of manually labeled training images with red, green, and yellow traffic lights. The output of manual labeling consists of three sets of labeled pixels for each traffic light color: $\mathbf{P_R}$ for red, $\mathbf{P_G}$ for green, and $\mathbf{P_Y}$ for yellow. Extracting the thresholds from the training sets is done by evaluating the normal distribution.

For each traffic light color we compute the mean value and standard deviation for each component: hue, saturation and luminance. When choosing the threshold value for hue, saturation, and luminance we make use of the “68-95-99.7” property of normal distribution. Values within three standard deviations from the mean account for 99.7% of the set, and this property is used for selecting the interest pixels.

Each pixel is filtered using the obtained thresholds; note that each component has a lower and an upper threshold. After applying these thresholds, for each traffic light color, three images are obtained which are saved for further processing. These are binary images that contain non-zero values in the areas that satisfy the thresholds.

The predefined horizon line is used to process only the upper part of the image. However, there are some transitory situations where vehicles ahead are included in the search space (when braking hard, for example).

The results obtained by thresholding retain all the traffic lights, but also include some false detections, such as car tail lights. The next steps in the algorithm have the purpose of filtering these false candidates.

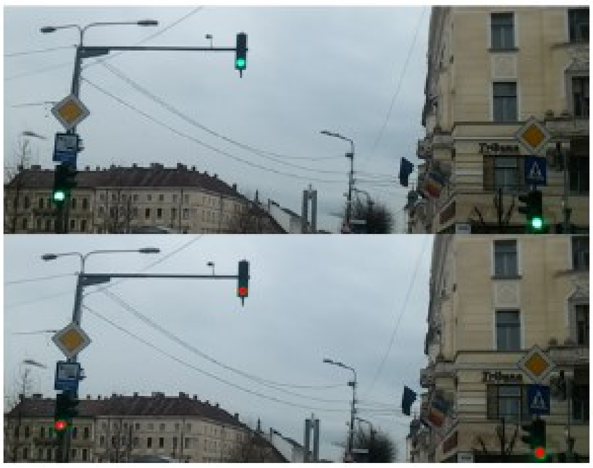


Fig. 3.6. Result of color threshold segmentation for the green traffic lights present on the top image. Detected green pixels are marked with red, on the bottom image

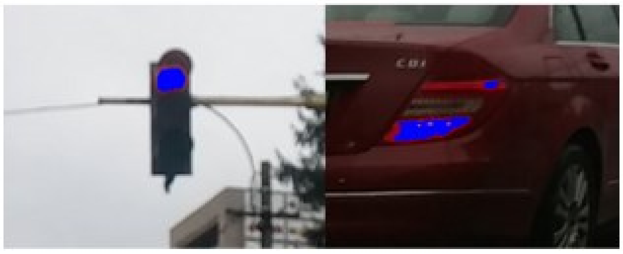


Fig. 3.7. Red traffic light and false detection on car tail lights (detected red pixels marked blue)

3.2.2.4. Filtering by geometrical features

Some false candidates need to be filtered out after color threshold segmentation. A pre-processing step that extracts connected components from the threshold image is applied.

A series of validations are performed with the purpose of confirming circle like features for each candidate. First, any region having an area larger than a set threshold is discarded. A region area is defined as the number of non-zero pixels it contains. For each extracted region the aspect ratio is computed. If the aspect ratio of the region is too far from a threshold, the region is also discarded. This criterion is based on the circle property of having an aspect ratio equal to one.

Filling ratio for a connected region is defined as the ratio between the number of pixels in a connected region and the area of its bounding box. For a circle, the ideal filling ratio is equal to 0.785 so a threshold smaller than this value should be chosen.

3.2.2.5. Circularity constraint filtering

Filtering by geometric features ensures that only components that can resemble a circle are considered for further processing. However, not all false regions are removed (examples in Fig. 3.8, bottom row). A more accurate circle validation which uses image edges is implemented in this step.

Edges are extracted based on the gradient magnitude. The advantage of using gradient magnitude over a more sophisticated method is the computation speed. To obtain the gradient magnitude, a simple derivative convolution mask is applied both in the horizontal and vertical directions.

A traffic light is composed of a bright blob surrounded by a darker area represented by the traffic light body. The magnitude of the gradient in the area will be high. Also, the inside of the traffic light has a constant intensity so the computed gradient here will be close to zero.

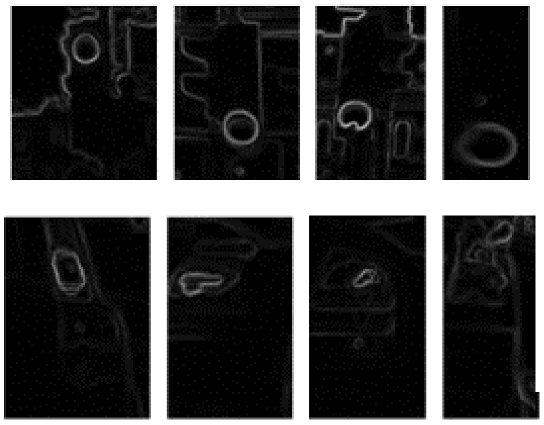


Fig. 3.8. Image gradients in candidate areas. First row contains traffic lights and in the second row car tail lights are presented.

Because the approximate position and parameters of the searched circle are known only a simple validation is required. Based on width, height and location properties of the candidate regions an estimated circle diameter and center can be obtained. Taking into account the image resolution and the previously mentioned circle parameters we can define a search area. The search area is circular with its inner / outer edges delimiting a band of pixels. The diameter of the outer edge is computed from the size of the bounding box, and the inner edge diameter is about 70% of the outer edge diameter.

Each pixel in the area with gradient magnitude over a certain threshold is counted to a gradient sum. This sum represents the filling ratio for the search area. If the filling ratio is below a certain threshold then the candidate is discarded. Through trial and error, a relaxed threshold of 50% was selected, otherwise the filtering is too strong. In this way we allow some variation in the traffic light shape, but still enforce a quasi-circular contour for the candidates, and thus filtering most of the false detections. Additionally, any false detection which does not contain a sharp transition at its edges will be filtered out.

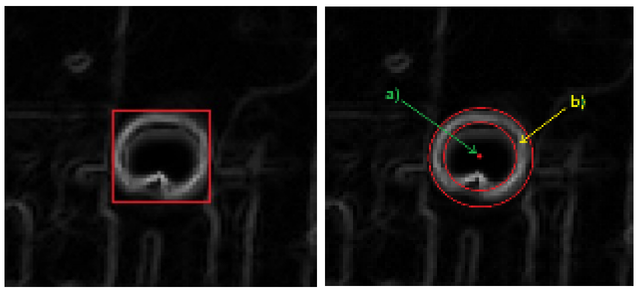


Fig. 3.9. Extracted bounding box for a traffic light and search area determined by its properties; a) center estimation; b) band of pixels used as search area

An additional condition is imposed here, which is meant to filter out most of the false traffic light detected on vehicles' tail lights, when the horizon line is not correct. Depending on the active color detected for the traffic light, assuming a standard three color semaphore, the rectangular area of the traffic light bloc can be computed. Except for the lit area, the rest of the traffic light block should be close to black.

If the diameter of the active light is *d* pixels, then the block rectangle has width of *d* and a height of about three times *d.* As position, the rectangle is placed relatively to the active light (see examples for red and green lights in the next two figures). For a traffic light candidate, we exclude the active pixels (from the labelled circular red or green area) and compute the average value of the L channel on the remaining pixels. If the average is below a threshold, then the traffic light is considered valid.

3.2.3. Experimental results

For testing the algorithm, several video sequences were acquired using a smart mobile device camera, during several hours of driving. The mobile device was placed on the dashboard of the vehicle, in fixed support. Images were captured at a resolution of 1280 × 720 pixels and a frame rate of 15 fps.

For training the HSL thresholds, despite the large number of images collected, there were not enough (not enough variability and different instances) images to train the HSL thresholds for the yellow light.

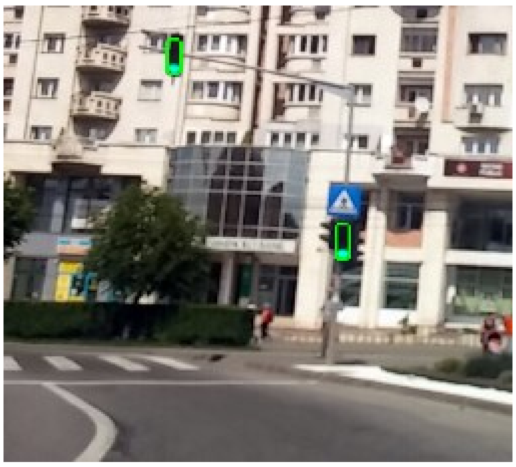


Fig. 3.10. Detected green traffic light example (the green rectangle shows the area of the traffic light block)

For training we used images from a separate set than the test images. This set contained 50 instances of red traffic lights and 50 instances of green traffic lights, each randomly selected. This size provided enough diversity for each color, for the three HSL channels.

For images with resolution of 1280 × 720 the average processing time of the algorithm on a standard PC (I7 Intel processor) was 11 ms. In contrast, on smaller images of 640x480 pixels, the algorithm runs in 5 ms, faster than the methods mentioned in the state of the art section. The implementation was carried out in C/C++ (for the mobile device, the Android NDK was used).

On a smart mobile device equipped with a quad-core processor at 2.3 GHz the results were in the range of 50-60 ms. On the same target, the computation speed for images with resolution of 640 × 480 is in the range of 30 ms.

For testing purposes we extracted two different datasets, one for close range and one for long range (results are for the larger image resolution).

Table 3.4. Detections results close range

	Green	Red
Total images	204	163
Traffic light instances	347	291
Traffic light scenes	14	8
Accuracy	85%	80%
False detections	5	3

Table 3.4 presents the detection results on the first dataset. This dataset contains images in the closer range of the traffic light, less than 20m. The false detections for the green traffic lights are caused by the pedestrian crossing lights or intermittent lights signaling right turns. For the red traffic lights false detections come mainly from car tail lights, in images where these are located above the horizon line. Compared to state of the art methods, the accuracy is similar or slightly lower (there are some methods with over 90% accuracy) but this is a tradeoff for achieving lower computational complexity.

Table 3.5 presents results from a different dataset where we included traffic lights at a greater distance, with a maximum of about 50m. The results in this case remain promising but are lower than in the first case because some of the traffic lights have a very small area and are discarded by the algorithm. One solution to this problem would be to use higher resolution images, but that is unfeasible given the current computation power of mobile devices.

Table 3.5. Detection results long range

	Green	Red
Total images	319	257
Traffic light instances	243	182
Traffic light scenes	6	6
Accuracy	73%	64%
False detections	2	5

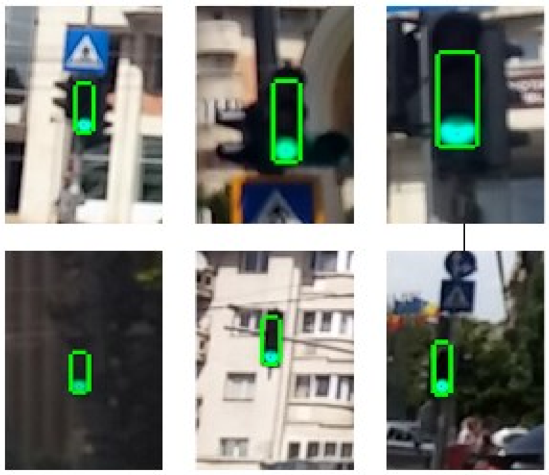


Fig. 3.11. Detections on green traffic lights

In Fig. 3.11 detection results on green traffic lights are presented. For demonstration purposes we have chosen traffic lights at different distances. Both suspended and un-suspended traffic lights are detected.

In Fig. 3.12 detection results on red traffic lights are presented. The marked area is the approximation the algorithm makes on the traffic light block. There are some differences in this sense but in general this approximation fits well on most traffic light blocks. These errors are mainly caused by factors such as the visibility of the traffic light or its intensity.

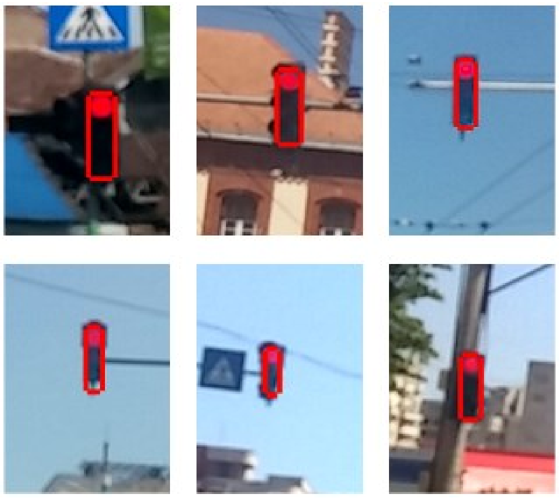


Fig. 3.12. Detections on red traffic lights

Fig. 3.13 presents false detection on car tail lights, obtained when the horizon line is not used (or if it is wrong) and the second filtering condition (check if the traffic light block is black) is not applied. This is a common situation for cars that have circular tail lights or appear to be circular in our images due to various distortions such as motion blur.

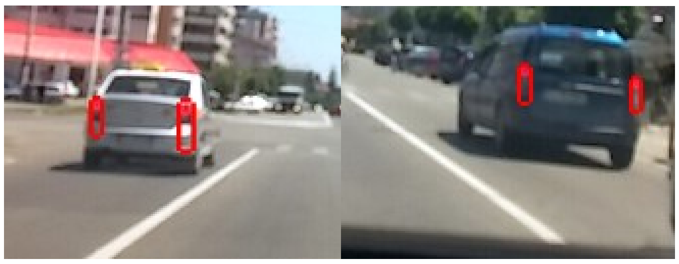


Fig. 3.13. False detections on car tail-lights

3.2.4. Conclusions

A fast traffic light detection system for smart mobile devices was presented. The color features of the traffic light are modeled as a Gaussian distribution and their parameters are learned using a set of training images. Because of this traffic light colors can be discriminated easily. An efficient validation that takes into account shape information is proposed. The algorithm suits the processing capabilities of a mobile device and provides acceptable detection rates. Future improvements will be made by implementing a tracking scheme to validate / refine the detection results.

4. Fast Obstacle Detection Using U-Disparity Maps

A computationally efficient approach for obstacle detection in driving assistance applications, based on stereovision, is presented in this chapter. The proposed approach involves three different steps aiming an increased quality of the results. The first step relies on the basics: obstacles correspond to peak regions in the u-disparity map. By applying the model of the stereo system, the peaks are detected with an adaptive threshold. The adaptive thresholding will calculate the accumulation of points required for an obstacle based on its distance (disparity) and will be related to the characteristics of the stereovision system. The second and third steps consist of refining the result of the previous, vertically respectively horizontally. This is necessary in order to fill out unmarked pixel regions which are classified as belonging to obstacles. The second step iterates vertically and propagates the obstacle label to neighbor pixels. The third step improves obstacle regions horizontally, with points that do not belong to the road. The solution is fast and reliable, on various scenarios, as every step is an improvement of the standard U-disparity approach for obstacle detection. This contribution was published at an international conference (*ICCP 2015,* Table 1.1, row 3).

4.1. Introduction

Obstacle detection is widely used in driving assistance applications to localize interest obstacles, or, secondary, it may be used as a preprocessing step for road surface detection. It can be done with different types of sensors, although most do not provide dense 3D data (radar, standard lidars) or, if they do, they are very expensive (lidars with high number of beams – ex. 64). Stereovision provides dense 3D data (and intensity/color) while having enough accuracy for most driving assistance applications. In addition, the received data can be further exploited, enabling increased possibilities, for example to classify obstacles.

The algorithm proposed in this chapter was developed upon the widely used u- and v-disparity concepts (Fig. 4.2). The disparity map generated by the stereovision system is used to build the two u- and v- disparity maps. Then, peaks are extracted from the u-disparity map, not by simple thresholding, but by using an adaptive thresholding scheme. The proposed solution does not require calibration of the threshold value with the distance, as it adapts to the characteristics of stereovision system and evaluates adaptive threshold values. Results are further improved with the horizontal and vertical enhancement steps. Some results of the solution are presented in Fig. 4.1.

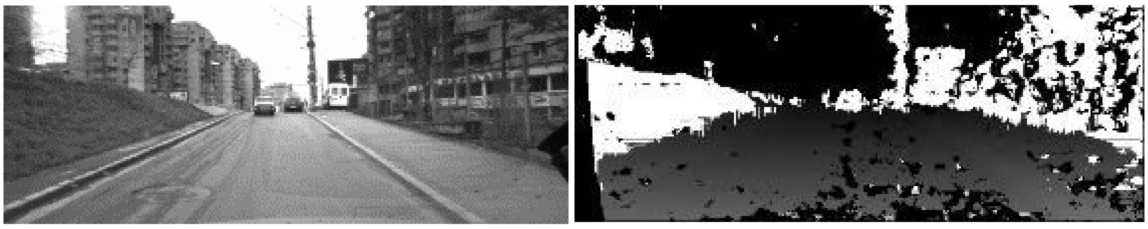


Fig. 4.1. Sample results of obstacle detection using the proposed approach. The disparity image is labelled with white where the obstacle pixels are detected

4.2. Related work

The primary output of a stereo system is the disparity map, which shows, for (most of) the left image pixels, which is the displacement (disparity) in pixels for the right image correspondent, relative to the left image position. Finally, for each valid disparity, the geometry of the stereo system allows the computation of the 3D location of each point with valid disparity. Due to the high volume of data to be processed (tens to hundreds of thousands of points), with a high frame-rate (10-20 fps), obstacle detection from stereovision presents some challenges. Some form of dimensionality reduction, or a representation with explicit connectivity, is employed.

Next, the focus will be mainly on methods that perform most of the processing on the primary output of the stereo matching: the disparity map. State of the art methods that are using the 3D point cloud as input (raw, or dimensionally reduced as digital elevation maps, voxel grids etc.) will not be reviewed.

In [27], Labayrade detects obstacles by classifying pixels using u-v-disparity map concept. Methods that estimate the roll/pitch/yaw using standard [28], [29] or quasi-standard [30], [31] stereovision techniques are detecting obstacles as belonging to a vertical plane.

Segmenting the road into blocks and identifying obstacle as sets of columns that stand perpendicular to the road surface is presented in [32]. Other methods, such as obstacle detection through color segmentation [33], [34], segmentation using watershed transformation [35], [36] are computationally complex. The U-V-disparity concept is used in [37] towards a more general representation of the driving environment, as obstacle are modelled as planar surfaces.

The solution presented next can identify the obstacles on the road surface, with very low computational complexity. It is independent of the road characteristics such as: planar or non-planar, curved or straight shaped surface, making it a feasible solution for general road scenarios.

4.3. Algorithm overview

Obstacle detection through stereovision is often based on the u-disparity map, where the map counts the appearances of disparity values in a column wise manner. This basically means that a column in the u-disparity map is the histogram of disparity values, on the corresponding disparity image column. The number of rows corresponds to the number of disparity bins, while the columns number is equal to the image width. The construction of the u-disparity map consists of iterating through the disparity image and incrementing the u-disparity map location, where the disparity of the pixel will correspond to the u map row position and the u map column position will be represented by the column location of the current pixel.

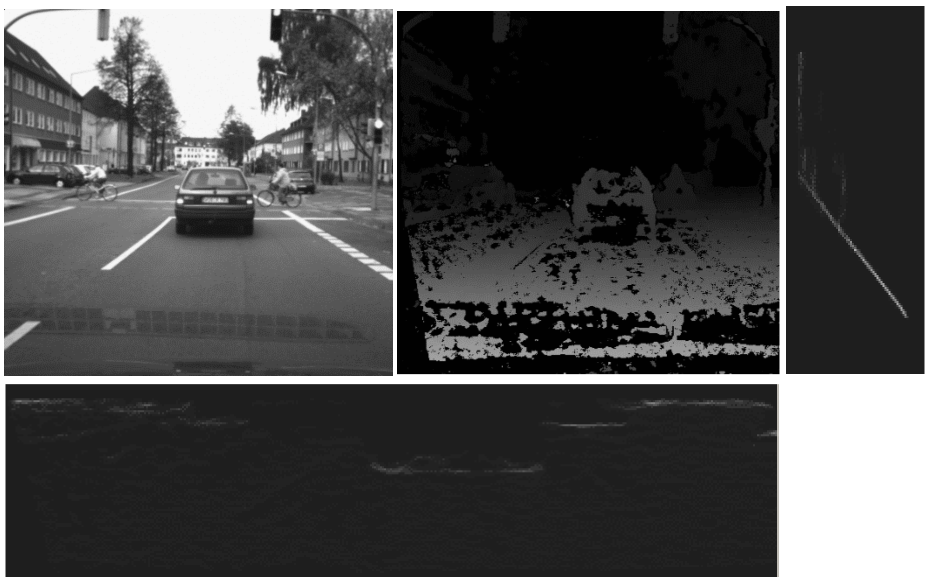


Fig. 4.2. The left image (top-left) and the disparity image (top-middle) are shown. The v-disparity map is on the right, with the oblique road profile clearly visible. The u-disparity map (oversized) is on the bottom, the obstacles appear as horizontal lighter streaks

Once the u-disparity is constructed, the obstacles will correspond to peak regions in the map. Obstacle pixels can be detected by applying a threshold function; peaks which are higher than the threshold are classified as obstacle points. The reason why this works is that an obstacle consists of multiple pixels with similar disparity. This idea was applied in [38] as a crude obstacle removal technique, with the purpose of removing a part of the obstacle pixels, before constructing the v-disparity map. The problem with this approach stands in its limitation to cope with the perspective projection model, where the height of the image projection of 3D objects decreases linearly with the distance of the objects from the stereo system (Fig. 4.3); many obstacle pixels are omitted or too much noise remains with a fixed threshold.

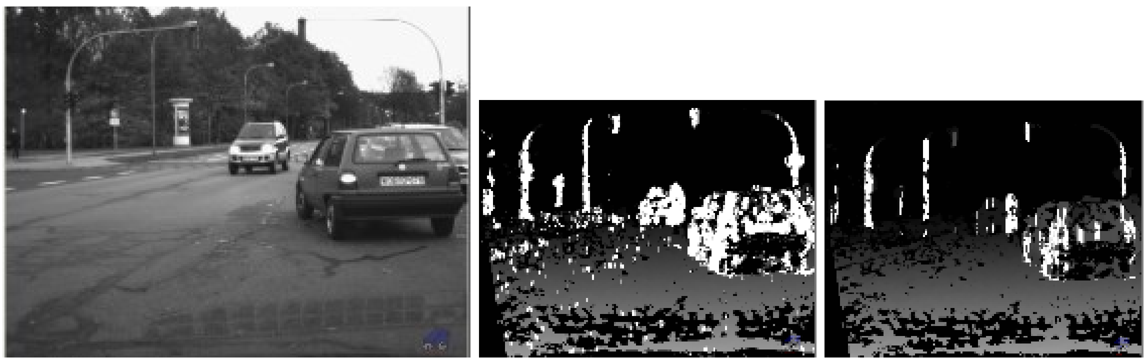


Fig. 4.3. If the threshold applied on the u-disparity map is not adapted with the depth, then noise remains if its value is too low, or most of the obstacle pixels are missed, if too high.

4.4. Adaptive thresholding using the stereovision characteristics

Each location of the u-disparity map must be classified as obstacle or not, by using a threshold adapted to the distance and the stereo system configuration (optical / geometrical parameters). Let us assume that the u-disparity map uses D_{max} bins for disparity counting (D_{max} is the maximum disparity that is computed by the stereo matching). For a given column c from the disparity image, the u-disparity corresponding column has D_{max}

locations, represented by U(d, c), where d=0..D_{max}. U(d, c) shows how many points from column c of the disparity image have the value d.

Obstacles usually have in their structure some vertical parts, or close to vertical (high slope). These vertical sides have similar disparity values, so there will be some cells in the u-disparity map where significant accumulation will occur. How large are these cells depends on the vertical size of the obstacle in the image. For a given obstacle, the vertical image size, due to perspective projection, depends on the distance between the obstacle and the stereo system.

Instead of using a fixed threshold upon the u-disparity cells, we can formulate an adaptive threshold that decreases with the distance. Based on the projection model of the stereo system, the adaptive threshold can be computed offline as the height, in pixels, of a 3D obstacle having a minimum size necessary for detection.

This is done by taking advantage of the stereovision characteristics, by using the standard model for a canonical stereo system:

Y_{3dmin} – minimum obstacle 3D height

y_{min} – image height after projection

F – focal length

Z – obstacle depth (3D) from the stereo system

Given the pinhole camera model:

$$\frac{y_{min}}{F} = \frac{Y_{3dmin}}{Z} \tag{1}$$

$$\Leftrightarrow$$

$$y_{min} = \frac{F*Y_{3dmin}}{Z} \tag{2}$$

The fundamental canonical stereo relation is:

$$Z = \frac{B*F}{d} \tag{3}$$

B – Baseline distance between the two cameras

d – Disparity value of the projected pixel

By replacing Z in equation (2) we get the height in pixels of the minimum obstacle to be detected, placed at such a distance that it has the disparity d:

$$y_{min} = \frac{Y_{3dmin}}{B} * d \tag{4}$$

For each possible disparity value, the adaptive threshold $Th(d)$ is computed as y_{min} with (4). At far distances, where the disparity is very small, this threshold would allow a lot of noise to be detected as obstacles. In order to avoid that, the adaptive threshold computed with (4) cannot go below a certain value (a minimum obstacle height in pixels).

For the stereovision system we have used, the baseline between two cameras is 22 cm. The value for the minimum obstacle 3D height value depends on the application (what obstacle sizes are relevant). A value of 30 cm was used for testing (so each obstacle should have at least a quasi-vertical side of 30 cm).

The online (per frame) classification of the u-disparity map consists of thresholding the map with the adapted threshold. Each u-disparity map location (d, c) is considered obstacle if U(d, c) is higher than the adaptive threshold $Th(d)$. For each location classified as obstacle, all the pixels from column c of the disparity image, with the disparity value d, are labelled as obstacle (Fig. 4.4).

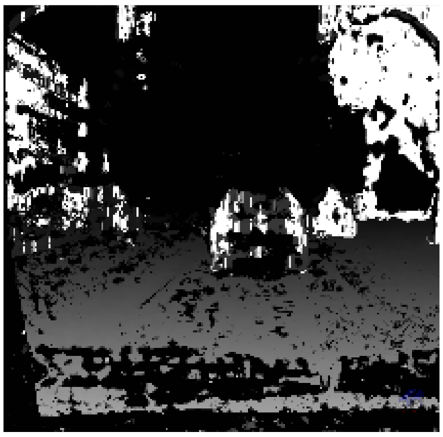


Fig. 4.4. Adaptive thresholding: obstacle pixels from the disparity map are labelled (white)

4.5. Vertical enhancement

After the adaptive thresholding step, many obstacle pixels are not correctly classified and are considered non-obstacles. In order to rectify this shortcoming, a vertical enhancement scheme is implemented consisting of scanning each disparity image column in both directions (top-down / bottom-up). Once an obstacle pixel is found, all the adjacent unlabeled pixels (up / down) having a similar disparity can be assumed to be obstacle. The vertical enhancement will improve the results by finding neighbor obstacle pixels which are not detected with the adaptive threshold. Basically, this step fills out most of the unmarked pixels, while the previous step bears the task of locating relevant areas of these obstacles. The result of this refinement step is shown in Fig. 4.5.

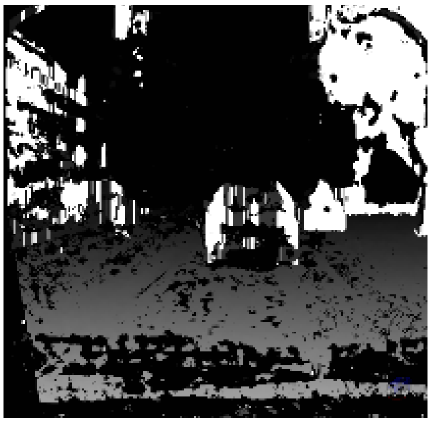


Fig. 4.5. The result of vertical enhancement (obstacle pixels are labelled with white).

4.6. Horizontal enhancement

The final step of the proposed solution attempts to complete the detection by filling all remaining gaps. Here, the additional cues that can be provided by the v-disparity map are exploited. Each row of the v-disparity map contains a histogram of the disparity values present on the corresponding image row. Given the particular position and orientation of the road surface relative to the stereo system (in driving assistance applications), the road pixels from an image row have similar disparity values. Thus, road disparities will usually accumulate in the v-disparity, and the maximum of each v-disparity row is likely to represent the road.

Instead of using the maximum of each row (which can belong to some large obstacle at certain distances), the road profile is extracted from the v-disparity map in an iterative fashion. The approach applied is related to the one described in [38], in particular for step 2:

1. Scan the v-disparity from the bottom (closest disparities) and find the first row with a non-zero maximum.
2. Iteratively, go to the next row, add the row maximum to the road profile if its location (corresponding disparity) is adjacent to the previous row maximum.
3. Stop if the profile becomes vertical on the last n added points (n=5), it is likely to climb an obstacle, or if the current row maximum is not adjacent to the previous one.
4. If the length of the extracted road profile is below a threshold (at least 25% of the road region height in the image, measured offline at calibration), the current profile is discarded, and the search restarts with step 1 from the next unvisited v-disparity row.

The horizontal enhancement works by scanning each disparity image row where a valid road profile exists:

- Given the scene geometry, obstacle pixels have higher disparity values than road pixels.
- Pixels unmarked by the previous two steps are analyzed: if their disparity is significantly higher (to exclude surfaces as sidewalks) than the road profile disparity then the pixels are labelled as obstacle (see Fig. 4.6).

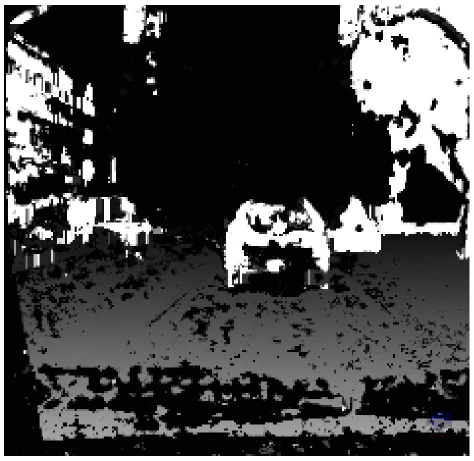


Fig. 4.6. The result of horizontal enhancement on the previous image results, see Fig. 4.5

4.7. Results

The proposed algorithm was implemented in the Visual C++, and integrated in a stereovision framework. The disparity map was estimated using a TYZX DeepSea [39] board, or using a semi-global stereo matching algorithm [7], as both alternatives were available in the framework.

The running time of the obstacle detection algorithm, on a standard PC (Intel I5 processor), is about 4 ms. This includes the computation of the u- and v- disparity maps, and the three main steps of the algorithm.

Several sequences (thousands of images) were used for testing, but only a qualitative evaluation was performed. The algorithm works fine, overall. Some small false obstacles do appear if the stereo disparity has rough errors. The frequency of these errors depends on the quality of the images (scene illumination, whether conditions etc.) that influence greatly the stereo matching process. Future quantitative evaluation with manually labelled images is required.

The results obtained with each step are illustrated below:

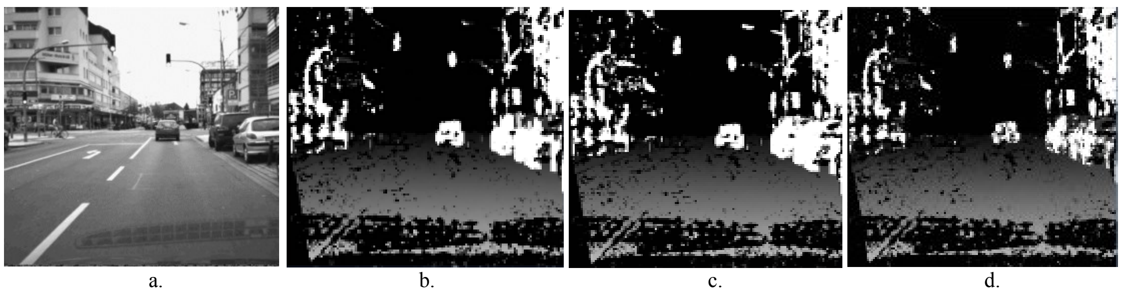


Fig. 4.7. Results of the steps of the proposed solution: b. Adaptive thresholding, c. Vertical enhancement, d. Horizontal enhancement

More results are shown on the next page, in Fig. 4.8.

4.8. Conclusions

The proposed solution is built up of three steps. The first step detects potential obstacles, while the last two steps improve the result of the solution by finding unmarked pixels that belong to an obstacle. By changing the required minimum height of an obstacle, one can change the detection target of the solution. Various sizes of obstacles can be detected depending on the requirements of the application. Furthermore, it can be extended to detect objects based on width characteristics. The proposed solution evolved from simple thresholding to adaptive thresholding with additional enhancement. The adaptive thresholding demands no calibration of the value with the distance, only a minimum obstacle size must be provided. The third step does not require a particular road shape (flat or planar), thus it can be applied in various scenarios.

Future improvements are necessary, towards refining the output, by analyzing the results at obstacle level. The algorithm, so far, is dealing with pixels individually, without using a higher level representation (and filtering criteria) such as clusters (one for each obstacle).

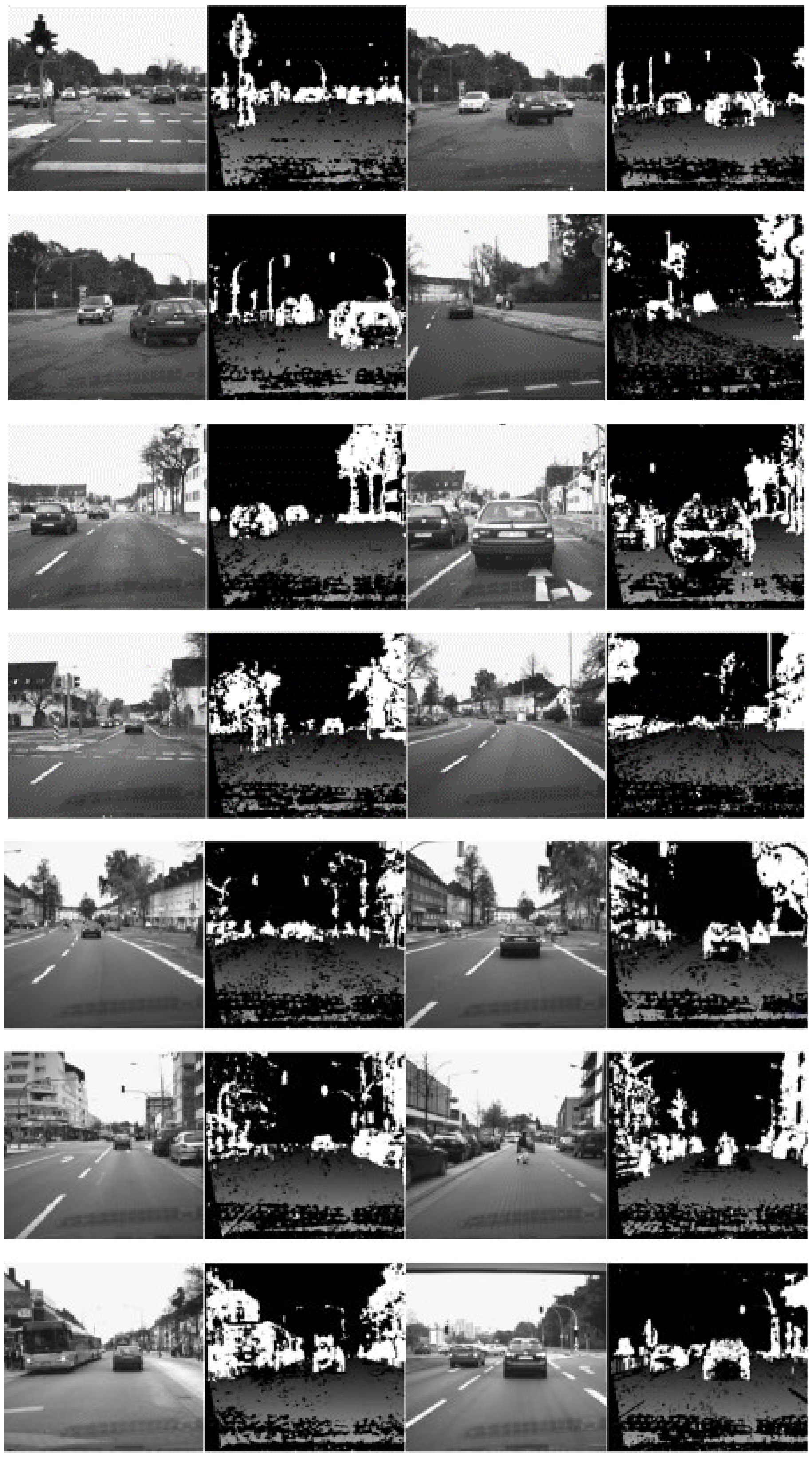


Fig. 4.8. Various results

5. Road surface and related road features detection

In this chapter, the research work done by the candidate within the three projects from (Table 1.3, rows 3-5) is presented. These projects, acronyms RB-1, RB-2 and RB-3, were coordinated by the candidate (as project responsible on behalf of TUCN). Beside the coordination of the team, the candidate was actively involved in the research work. The main topic of research was environment perception from stereovision suitable for market-deployable driving assistance systems.

The achievements within these projects were models and algorithms beyond the state of the art (high accuracy and processing times in the range of few milliseconds or lower) for: road surface detection, lateral delimiters detection, small road features detection, such as curbs, speed bumps, lane grooves, potholes. For the last three features no state of the art solutions were available at the moment of the project implementation. Due to non-disclosure clauses imposed by the beneficiary, no public dissemination of the results was possible.

There are two main approaches in the literature for road surface detection. The first one involves modelling the road with a global parametric model, and compute the parameters of this model from available 3D data. This approach copes well with missing data and is able to estimate the road surface level even in regions where no 3D reconstruction is available. The main drawback is that model violations are likely to appear: the real road surface might be too complex to be modelled globally. The simplest model (and the oldest proposed) is the flat road model: the road is a flat plane (null pitch and roll angles) at a predetermined elevation [40], [41]. This model requires a simple implementation but it has limited practical benefits. The next model proposed is the planar road model, which models roll and pitch angles of the road surface [42], [43]. Modeling vertical curvatures is not possible with this model. Polynomial surfaces (quadratic) were proposed in [44], [45]. A more general model is the B-spline model, but it was proposed for modelling only the vertical road profile along the longitudinal axis [46], [47]. It is actually a curve (not a surface model) and does not model complex lateral road profiles.

The second category of road surface models are those that focus only on the local shape of the road. This is done by imposing parametric constraints locally in the 3D space, on the slope the 3D data, mainly along the longitudinal direction. Complex road surfaces can be modelled, but only if 3D surface data is available. Elevation gradient is used in [48] to assess the local navigability. Slope evaluation filters were used in [49]. A succession of planar patches without roll was proposed in [30], and later extended through region growing in [33].

Next, an overview of the research conducted by the candidate will be presented.

5.1. Accurate road surface estimation and speed bump detection

The goals of the first project (2013, RB-1) were (1) to compute the road surface with a vertical localization (centimeter level) beyond the typical accuracy of the raw stereo measurements and (2) to use the accurate road surface profile to detect speed bumps. The first goal has a different focus compared to road model methods available in the literature. State of the art road models are used mainly for road/obstacle separation in the 3D data, where several centimeters accuracy (even 10 or 15 cm) is acceptable to separate the 3D points of vertical obstacles like pedestrians, poles, guardrails etc., from the road surface.

5.1.1. Accurate and fast estimation of road vertical profile

For the purpose of accurate road surface estimation, a grid representation was proposed for processing of the stereo data. Each grid cell covers a square are in the top view and contains the set of road 3D points, with associated 3D stereo uncertainty. The road model will be a "freeform" one. A grid with the same size (depth x lateral) as the grid used for processing is generated, but with (at most) only one 3D point in each cell, representing the road level of the cell.

In what follows, when referring to a 3D point, beside the three standard coordinates (X, Y, Z), the point will also have standard deviation values for its coordinates (computed from the stereo uncertainty model). In the context of temporal integration, 3D points will have an age (measured in frames).

The road surface estimation algorithm was designed having in mind the need for increased accuracy/precision on the vertical axis. The purpose of the research work was not to detect various road surfaces/models. Some key features were proposed: local averaging by computing the mean values of the raw measurements in a grid cell, temporal integration of the raw stereo measurements (to have a better support for the local mean computation), and fitting low-order polynomials to small patches to get the final road elevation. The road surface estimation algorithm has three main steps that will be detailed next (bold emphasis).

Selection of 3D road (and nearby) points: Use a quadratic surface to select the road points (including the uneven regions such as bumps) for further analysis. First, a quadratic surface is computed from the raw stereo measurements that are in a region of interest in front of the ego vehicle. The quadratic surface is then used to select potential road points: those points that are around the surface (elevation Y_{road}), within a constant band plus the stereo vertical uncertainty (computed with the uncertainty model of the stereo system), and store them in the grid:

$$if\ |Y - Y_{road}| < Vslice + Y_{err},\ then\ P(X,Y,Z)\ belongs\ to\ the\ road$$

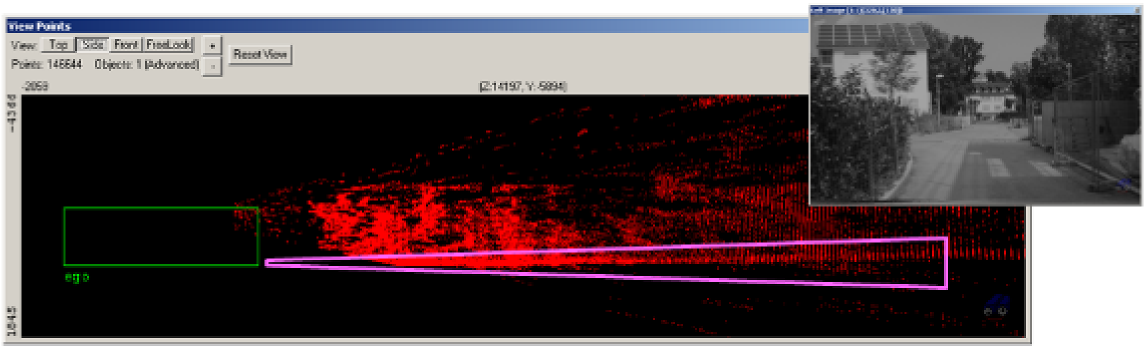

View Points
View: Top Side Front FreeLook
Reset View
Points: 146644 Objects: 1 (Advanced)
-2068
(Z:14197, Y:-5894)
ego

Fig. 5.1. Side view of the 3D points and the vertical interval (magenta) that increases with the depth, used for selecting potential road points.

Temporal integration of the raw 3D measurements: Enrich the raw stereo measurements of the current frame with measurement available from the previous frames, for the same 3D locations. The ego motion data was used for computing the horizontal (lateral-longitudinal) displacements, and a RANSAC approach was proposed to compute the vertical motion.

The standard, widely used model is employed for lateral-longitudinal motion compensation: circular trajectory, constant speed.

In order to have a fast and robust method for vertical motion estimation, a statistical approach based on the RANdom SAmple Consensus technique was developed. The vertical motion parameters are those parameters that align the previous and the current road points in a least square sense, without penalties from outliers (erroneous points or dynamic obstacles). These parameters are the relative pitch α, roll γ, and vertical displacement ΔH between the origins of the two reference frames.

Each road grid cell provides a 3D location given by the cell center (X_t, Z_t) and mean elevation Y_t in the current frame. For the same cell, we also have the previous frame mean elevation Y_{t-1} : after horizontal motion compensation it refers to the same 3D location Z_t=Z_{t-1}, X_t=X_{t-1}, except that the value Y_{t-1} was measured in the previous frame. Therefore, the relationship between the two elevations is:

$$Y_t = Y_{t-1} + \Delta H + \alpha Z_t + \gamma X_t$$

$$\Leftrightarrow$$

$$\Delta Y = Y_t - Y_{t-1} = \Delta H + \alpha Z_t + \gamma X_t$$

The elevation difference between the current frame measurements and the previous frame can be modelled with a planar surface. The plane equation is computed based on RANSAC and the vertical motion parameters are extracted.

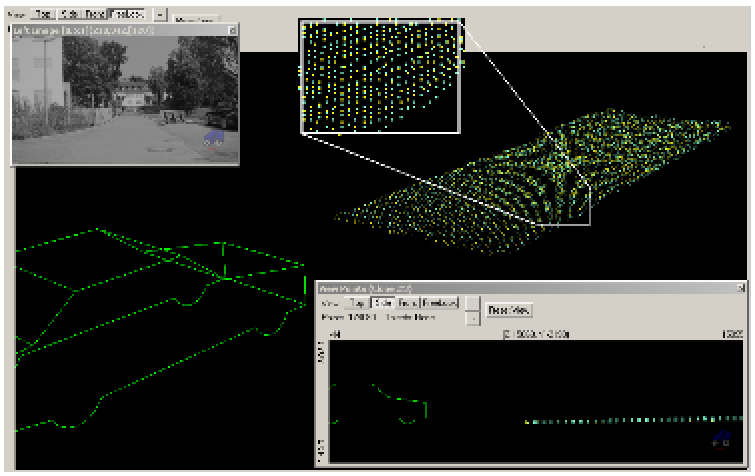


Fig. 5.2. A virtual camera view of the two set of points (built from cell centers and mean elevations) used for vertical motion estimation: previous (horizontal motion compensated) road points (yellow) and the current frame points (light blue)

For each new frame, a global grid $G_{global}(t)$ is built, containing for each cell the road points from the last frames, $G_{global}(t-1)$, mapped into the current frame, and fused with the road points from the current frame G_{road}.

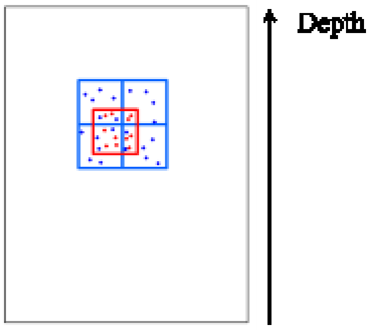
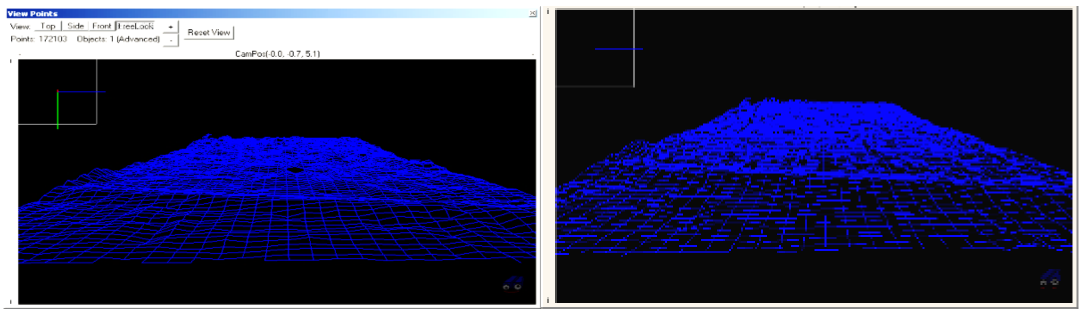


Fig. 5.3. A graphic explanation of how road points from the previous frame (red) are mapped into the current frame and distributed to the current grid cells (blue) they overlap

The improvement in the density of 3D points is visible in Fig. 5.4, where the mean road elevation for each cell is displayed: when temporal integration is used, the elevation of the road is smoother and fewer empty patches appear. As measurements accumulate over time, after enough frames, the global grid will have data for the whole region of interest, from 0 to the maximum depth, not only for the road surface visible in the current frame.



a. b.

Fig. 5.4. The mean road elevation displayed as a grid (the vertexes represent cell centers, perspective view from the camera): a. only current frame data, and b. with temporal integration

Road surface estimation: Perform local (cell) analysis to compute the road vertical profile (elevation, freeform), for each cell in the region of interest. Fitting with low-order polynomials for smoothing and final road elevation selection.

After the temporal integration step, the global grid G_{global} contains, for each cell, a set of 3D points, resulted from the stereo reconstruction of several consecutive frames, which are likely to belong to the road. The richer set of points should help estimating a better mean elevation value, as long as the proper weights are used. The 3D road points contained within each cell are the output of the stereo system, so a good idea is to relate the weights to the accuracy of reconstruction. The solution is to weight each point with the inverse of the variance of the 3D point's location, as estimated from the stereo uncertainty model. The standard deviation of the elevation for a cell is computed from the standard deviation of the constituent 3D points. This can be formalized by assuming the 3D points are not correlated (which is not entirely true due to the local masks used for stereo matching).

A new grid $G_{roadelev}$ is generated, each cell contains the cell's center 3D location on the X and Z axes, the mean road elevation on Y axis, and the standard deviation of localization on each of the axes.

There is one additional way to improve further the accuracy of the road, and this can be done by assuming that on small patches, the road can be model with a parametric surface. The parametric surface must have enough complexity to model locally road irregularities as speed bumps. If the size of the patch is smaller than the size of the irregularity, then a quadratic surface is fine. The size of the patch should not be too large, because it would filter out speed bumps, and it should increase with the depth to provide better filtering far away, where stereo data is poorer. For simplicity, a squared patch is used, as in the figure below.

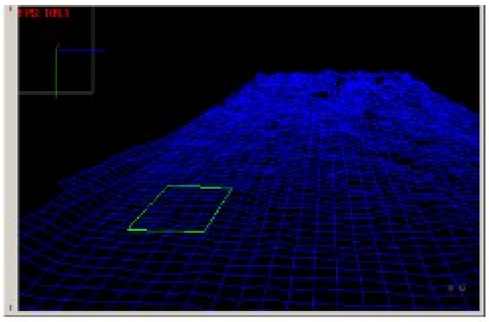


Fig. 5.5. Example of a local patch used for filtering with quadratic surfaces

For each cell of $G_{roadelev}$ the local filtering consists of two steps. First, a quadratic surface, is fitted (least squares) to the local patch of *m* x *m* cells, centered on the current cell. The surface is considered valid if at least half of the patch cells have data and these cells are not placed only on one side of the central cell (unbalanced distribution). Next, if the surface is valid, the elevation of the current cell is set as the surface elevation in the center of the cell. Two objectives are fulfilled (Fig. 5.6): reduce the road elevation noise and the elevation is computed for most of the empty grid cells (if the missing patches are small).

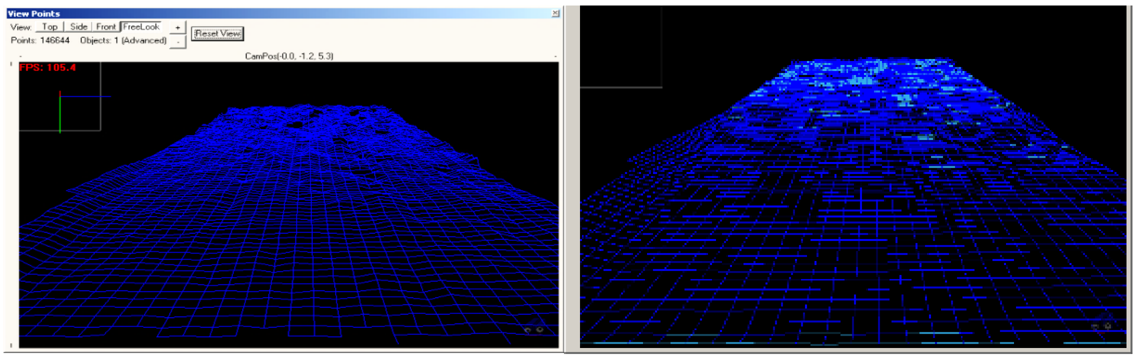

View Points
View: Top Side Front FreeLook
Points: 146644 Objects: 1 (Advanced)
Reset View
CamPos(-0.0, -1.2, 5.3)
FPS: 105.4

Fig. 5.6. The mean road elevation displayed as a grid (the vertexes represent cell centers): a. global grid data, b. after local surface filtering (light blue areas are showing areas where no elevation data was available from stereo). The perspective is from a virtual camera

The road grid, after temporal integration and subsequent filtering, will be used further for the task of speed bump detection.

5.1.2. Speed bump detection

The approach to localize speed bumps is based on analyzing the local slope in the road grid:

- Compute the local slope (2D, based on the gradient of the elevation)
- Filters out most uneven area that are not speed bumps

- Detect the two possible sides of the speed bump (uphill / downhill).

The first step involves detecting some interest points: cells likely to belong to uneven areas. The measure used for discrimination is the local gradient of the elevation, which is computed from the $G_{roadelev}$ data, by using a lateral/longitudinal derivative nucleus. The resulting magnitude and orientation are stored into two images, having the same size as the grid, and used for the future processing steps (Fig. 5.7.a). The selection of the interest points (Fig. 5.7.b) is based on the following rules: gradient magnitude above a threshold variable with the distance, and the gradient orientation specific to an uphill or downhill patch.

Now, clustering of the interest points must be performed for obtaining the speed bump primitives (results in Fig. 5.8.a). The clustering process is based on region growing, starting from a non-clustered interest point. Each region has a mean gradient orientation, computed from its current points. Neighboring interest points are added to the region if their gradient orientation is similar to the one of the region; mean orientation is updated.

Each cluster of interest points is analyzed to see if it represents an uphill/downhill area. Clusters too small are discarded. The eccentricity E, minor and major axes L_{min}, L_{max}, of the cluster are estimated by ellipse fitting. Specific thresholds are applied to mark each cluster as speed bump side, uphill or downhill based on the mean gradient orientation, or road patch. Results are shown in Fig. 5.8.b.

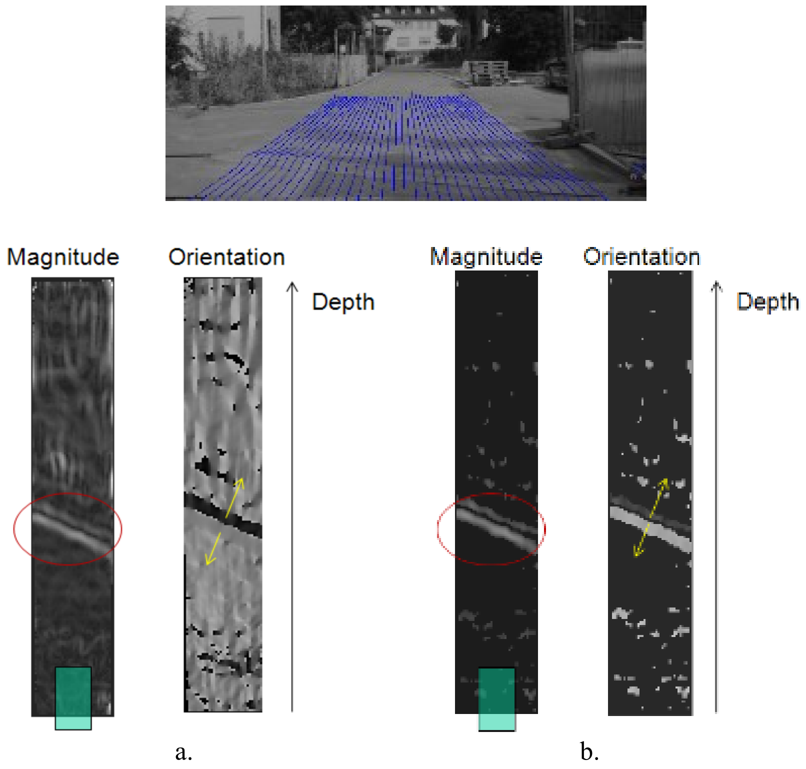


Fig. 5.7. (top) The road grid displayed on the left image (only the visible part, with longitudinal edges between vertexes, ego as the green rectangle), a. The magnitude and the orientation (gray level encoding) of the elevation gradient. The speed bump uphill/downhill areas are visible in both the magnitude and the orientation images, b. The interest points are selected (the other points have the magnitude and the orientation removed – set to 0 - black)

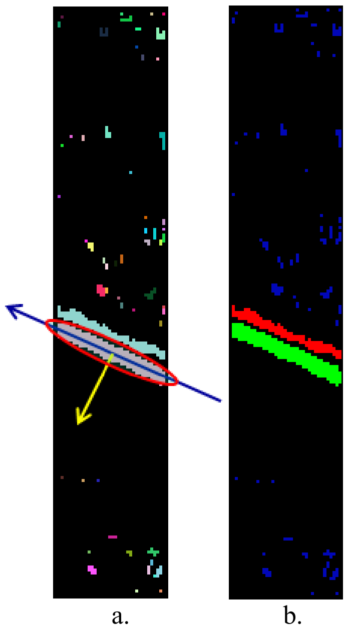


Fig. 5.8. a. Each cluster is fitted with an ellipse, the ellipse major axis (blue arrow) and the mean gradient orientation of the cluster (yellow arrow) are shown, b. Classified clusters: uphill with green, downhill with red, and road with blue

5.1.3. *Performance analysis*

Several test scenarios, with the ego car approaching speed bumps with known size, were used for evaluation. For a scene with a 6 cm speed bump, the estimated speed bump was 9 cm at 10 meters depth, and a height of 6.7 cm at 4 meters depth (from the front bumper). Generally, the height estimation error was usually less than 1 cm at depths of 4-6 meters (from the front bumper) and up to 2-3 cm at 10 meters. Compared to the raw stereo data, the accuracy/precision are improved several times.

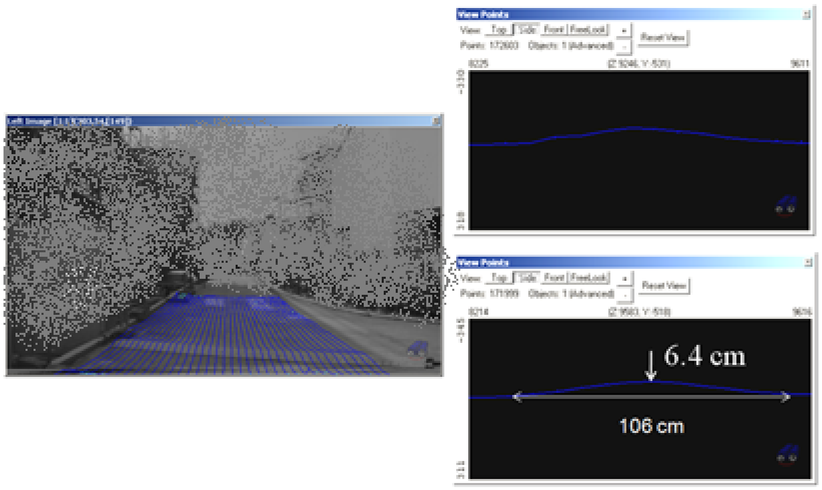


Fig. 5.9. A scenario with an oblique speed bump: on the right side are two side views of a speed bump slice

Regarding the quality of the road elevation, and implicitly of the grid classification, one of the most difficult scenarios are those where the road has oblique features in the intensity image (marks, shadows). These features are causing false uneven areas that cannot be distinguished from true uneven areas.

Speed was tested for the implementation in the TUCN environment, single thread, on Pentium I5 at 3.3 Ghz. The processing time for one frame is about 30 ms, when the number of 3D road points is high due to temporal integration. Memory requirements are about several hundreds of MB, however 90% of this is due to the fact that the global grid stores all the points from several frames.

An optimized memory/speed version of the algorithm was also developed in the last phase of the project, by not storing all the 3D measurements in the temporal integrated grid. Only mean elevation values were stored and integrated temporally. This version managed, with similar accuracy/precision, to run in less than 5 ms, with a memory requirement of only several megabytes.

5.2. Lateral delimiters detection

The research results from the second project RB-2 (2014) will be presented in this sub-chapter.

The main objective is the detection of the 3D lane delimiters. The delimiters are elevated areas that can be used independently for driving assistance applications, or they can serve as features for identifying the 3D lane boundaries (the 3D drivable corridor). The delimiters have two basic classes, left side or right side delimiters.

Left side delimiters can be isolated obstacles (vehicles, poles, pedestrians, etc.) that are placed on the left side of the current ego trajectory, or they are continuous delimiters (curbs, ditches, etc.) that have an elevation that increases from right to left. Similarly, right side delimiters have an elevation that increases from left to right.

The following issues were taken into consideration for the development of the delimiters detection algorithm:

- current frame grid must deal differently with road and obstacle cells (this requires prior detection of the vertical road profile)
- temporal integration must be as a mixture of cell / blob level integration
- obstacles must be dealt with in a different way as left / right delimiters, in contrast with continuous delimiters, which must be classified according to their lateral slope.

Solutions were developed for the computing the vertical road profile, dealing with stereo noise and with dynamic obstacles in the temporal integration, and the delimiters detection. Low memory requirements and high processing speed were compulsory.

5.2.1. Vertical road profile estimation

The approximate road elevation/shape should be known when building the current frame elevation grid. The preferred approach is to use the global grid from the previous frame to estimate the road level, update the road profile with the relative ego motion, and then build the current frame grid (mean elevation for road / maximum elevation for obstacles).

In the first project, for the initial selection of points we have used a quadratic road model, with a relaxed vertical uncertainty, to ensure that small undulations and nearby road features (such as speed bumps) are included in the selected points. When more precise separation is required between road and obstacles, and less dependent on the road model, a road model not constrained by a global parametric surface is more appropriate.

The approach developed is similar to the v-disparity based approaches, but it is applied in the grid space, instead of the disparity space (the intermediate disparity representation is not required). The search space is the lateral projection of the grid (Fig. 5.10, middle). For each grid row (depth), a histogram of the elevation values is built. Road level should coincide with the maximum of each histogram.

The stereo uncertainty and resolution (road points' density decreases with the depth) are causing sparseness in the search space; at far depths, the road profile is not continuous. A Gaussian smoothing is applied as a separable filter (result in Fig. 5.10, bottom), on each axis in the histogram space.

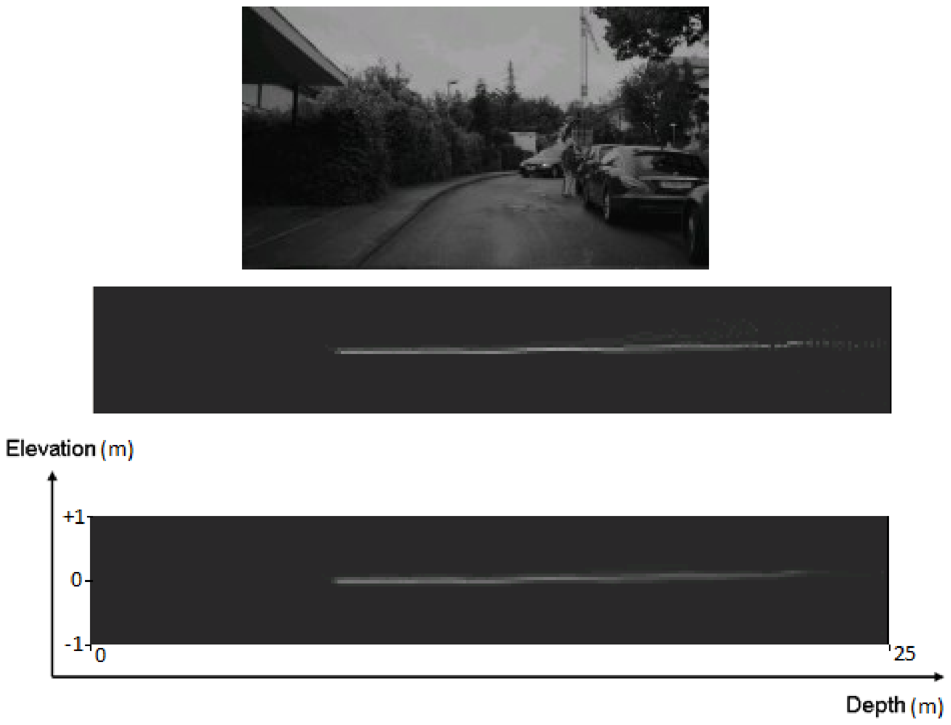


Fig. 5.10. Top: left image; middle: the search space used for road profile estimation, on each column the intensity encoding of the histogram of elevations is shown; bottom: the search space after Gaussian smoothing

Now, the road profile is extracted iteratively (Fig. 5.11, top) from the filtered search space. First, a starting point is selected, represented by a relevant histogram maximum, close to the ego car, within a certain elevation interval (likely road). Second, the next position of the road is searched iteratively, as the maximum of the next column from the three cells that are neighbors to the previous column road position. If there is a gap in the road profile, the next column road position is computed by using the linear approximation of the previous columns' road profile. At the end, the linear approximation of the road profile is used to extend the profile back toward the ego car. The result is shown in Fig. 5.11, bottom.

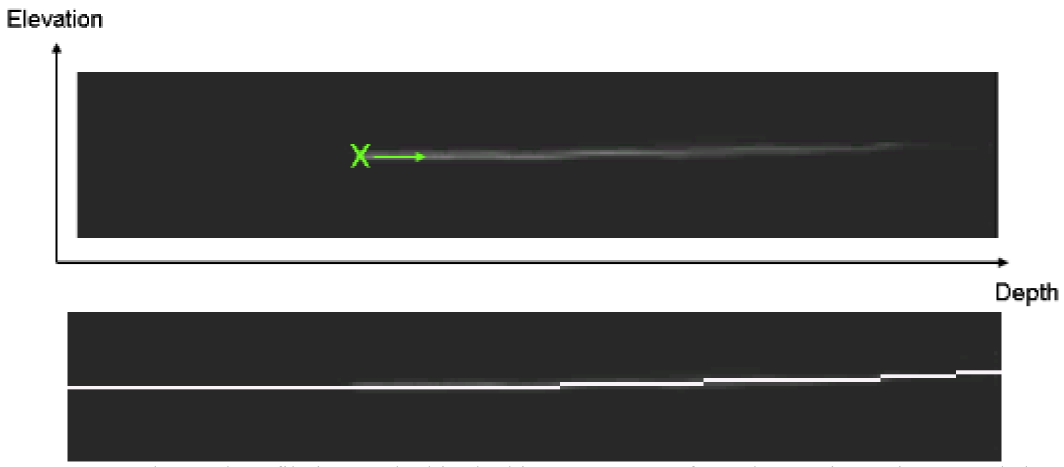


Fig. 5.11. Top: the road profile is searched in the histogram space from the starting point towards larger depths; bottom: the extracted road profile, extended back toward the ego car

Based on the estimated road profile, road and obstacle areas can be treated differently when building the current frame grid: the mean elevation values are stored for road cells, and the maximum elevations are stored for obstacle cells. Furthermore, each non-empty cell of the grid is labelled as road or obstacle.

The next figure shows the result of road/obstacle separation (Fig. 5.12). In addition, significant improvements are obtained for thin singular obstacles as poles/traffic cones (Fig. 5.13). Typical failure scenarios are scenes with road surface that has significant slopes on the lateral direction (Fig. 5.14).

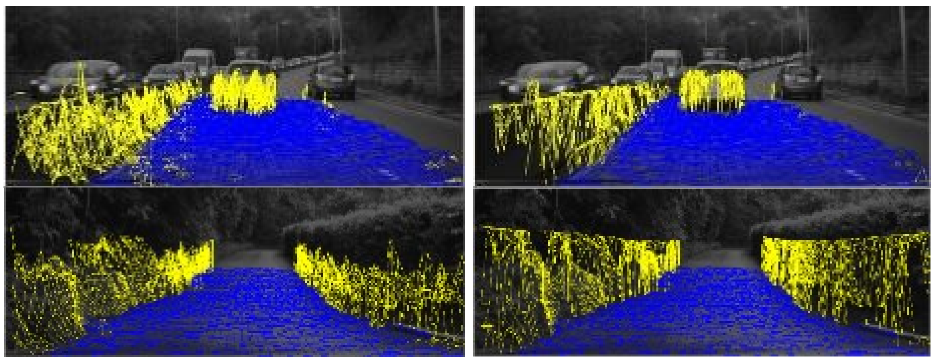


Fig. 5.12. Obstacles (yellow cells) have a better shape representation when the mean / max elevation storing scheme is applied (right column), in contrast with the max elevation storing scheme (left column) inherited from project 1

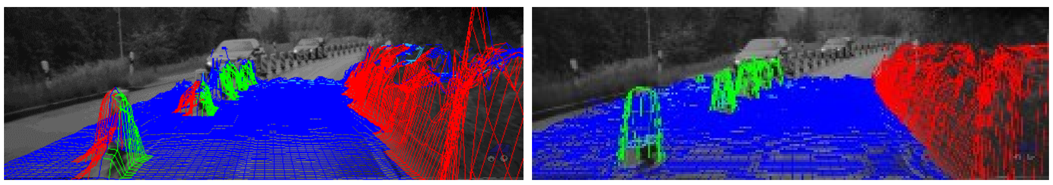


Fig. 5.13. The improvement in the shape representation of thin singular obstacles (mean/max scheme in the right image). However, these images are taken after all the algorithm steps are applied (to be described next), so the improvement is only partially (but significantly) due to the mean/max scheme

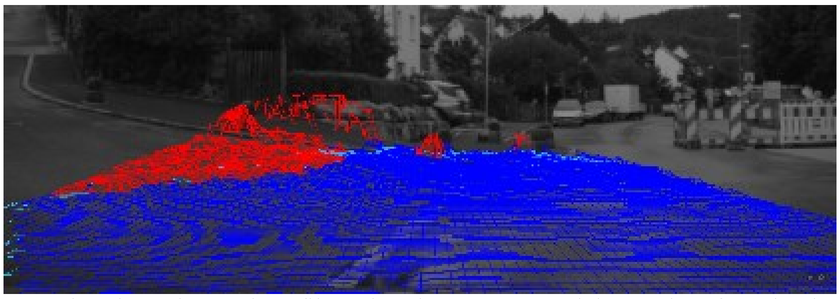


Fig. 5.14. A scenario where the road profile estimation cannot model a road surface that is composed of two surfaces: a quasi-planar one on the right, and a quasi-quadratic one on the left (elevation increases on the lateral direction)

When building the current frame grid, a much larger vertical interval is processed, not just a small interval around the road surface. Stereo artefacts/errors are present in the subset of 3D points used for grid building. These artefacts are 3D clusters of points that float in what actually is free space (Fig. 5.15). If they are not detected during the building of the grid, then large false obstacle areas will remain in the grid (Fig. 5.16, left image).

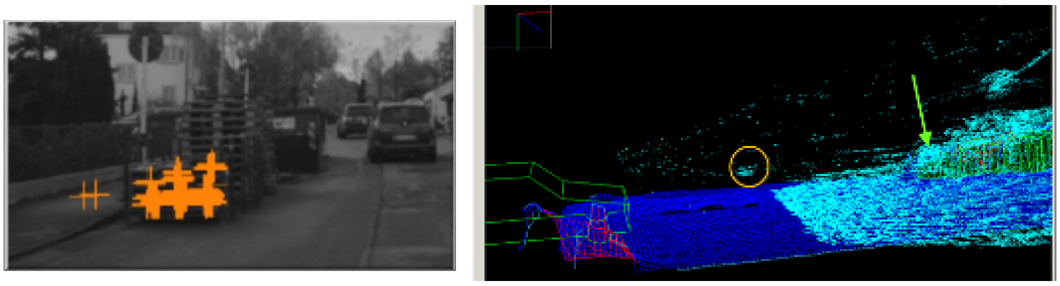


Fig. 5.15. A cluster of about 100 3D points (crosses on the left image) on a repetitive pattern, hovering in front of the car, inside the orange circle (virtual camera image). The green arrow shown where these 3D points should be located, if stereo matching was correct

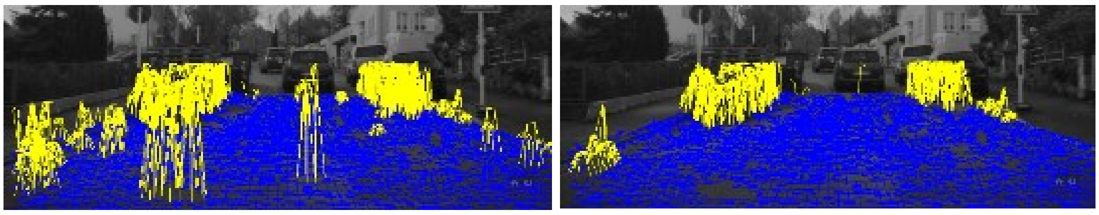


Fig. 5.16. False obstacle clusters are generated in the current frame grid if stereo artefacts are not filtered (left image), most of the false obstacle clusters are removed when stereo errors are filtered (right image)

When the current frame grid is built, beside the elevation value, the number of 3D points contained by each cell is stored. After the grid is built, an analysis of the density of 3D points in the grid is carried out in order to remove the stereo artefacts. The key idea is that stereo artefacts that float above the road should have lower densities of 3D points due to the gap present between the artefact and the road. Contrary, an obstacle of similar height should have significantly more 3D points (assuming it has enough texture for quasi-dense stereo matching).

The stereo system model is taken into account for the density analysis. First, for each grid depth, the size in pixels of the unit surface, which is vertical and fronto-parallel with the camera, is computed. A square of 1 mm x 1 mm are projected onto the image plane using the canonical stereo geometry, and the image area of the square is computed. A look-up table of densities for the unit surface is obtained.

Given an obstacle cell, or a group of obstacle cells, its fronto-parallel area (in mm^2) is approximated as a bounding rectangle (Fig. 5.17): average width x average elevation. Then, the expected number of 3D points in the cell (or group of cells) is obtained by multiplying its rectangle area with the unit surface density at the cell's depth (or average depth for a group of cells).

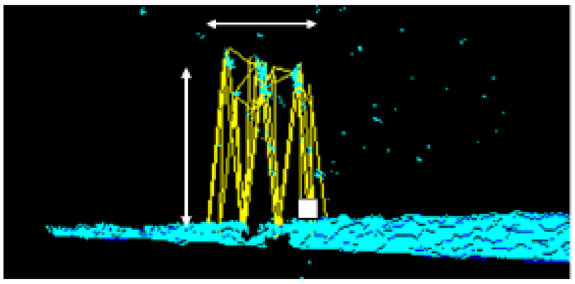


Fig. 5.17. A drawing of the square fronto-parallel unit area (the white square – drawn much larger for viewing purposes), and the features used for evaluating the obstacle's fronto-parallel area: the average width and the average elevation

By comparing the actual density of individual obstacle cells or of small clusters of obstacle cells with the expected density, most the stereo artefacts can be removed from the grid. Results are shown in Fig. 5.16, the right image.

5.2.2. Temporal integration of the classified grid

In order to keep some backward compatibility with the project 1 implementation, the bilinear interpolation scheme is applied only if the current cell grid is road and the four closest correspondent cells from the previous global grid are road too.

In the context of the current approach for temporal integration, the age of a cell in the global grid represents the number of frames where the class of the cell was re-confirmed (the same class assigned as in the previous frame) by the 3D stereo data.

For each current grid cell G_c, choose the correspondent global grid cell G_g (previous frame, after ego motion compensation), using a nearest neighbor interpolation:

1. If G_c has data, then:

    a) If G_c and G_g have the same class (road / obstacle) – update with weighted average, increase age.

    b) If G_c is road and G_g is obstacle – keep G_c. This is the typical scenario where a moving obstacle must have its past cells overwritten by road, or emptied, unless they are re-detected from the current frame stereo data.

    c) If G_c is obstacle and G_g is road – keep G_c if G_g is older than 2 frames, otherwise keep G_g and set its age to 1 frame (make sure that this cell will not cancel out an obstacle cell in the next frame, if re-confirmed by the stereo measurements). This step is a complementary measure to filter out remaining stereo artefacts.

2. If G_c has no data, then:

    a) If G_g is road – keep it by replacing G_c, basically all road locations are accumulated over time.

b) If G_g is obstacle – keep it if it is in a 3D region invisible for stereo reconstruction. This is explained next.

Sub-step 2.b deals with obstacles in the global grid that are no longer visible in the current frame for the stereovision. In this case, these obstacles must be preserved in the grid (dynamic obstacles will be stored at their last measured position). The region not visible to stereo is composed by two sub-regions, given by the horizontal and the vertical field of view of the stereo camera.

With this new approach for temporal integration, false trails from dynamic obstacles are filtered out despite the fact that the reasoning is done mainly at cell level. A relevant example is shown in Fig. 5.18.

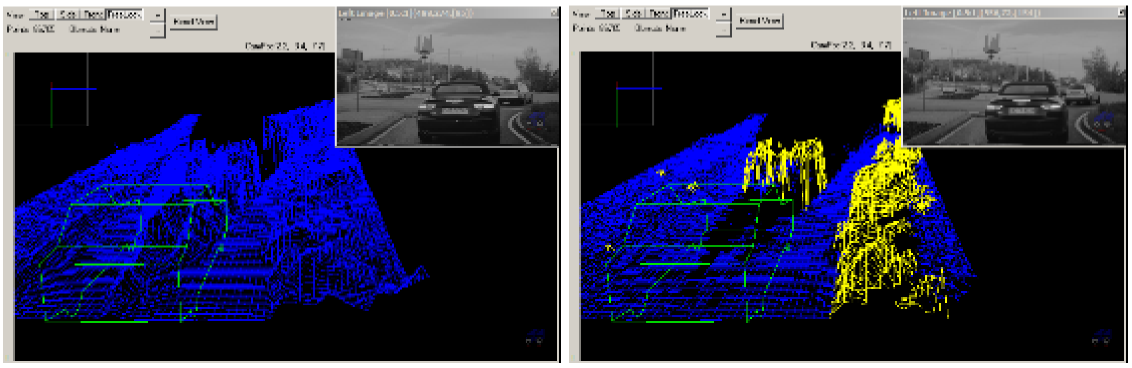


Fig. 5.18. The temporal integration from project 1 (previous sub-chapter), based only on combining elevations, leaves large false trails behind dynamic obstacles (left image). The new temporal integration scheme, that takes into account the class of the cell (road - blue, obstacle – yellow), successfully removing dynamic obstacle trails (right image)

5.2.3. Large delimiters detection (obstacle type)

The initial approach developed for delimiters detection was based on identifying significant patches that exhibit the same elevation variation (increasing towards left, or right). A map of the lateral elevation gradient was computed and cells were clustered based on their gradient magnitude and orientation. This approach worked fine for off-road scenarios with natural road side delimiters (vegetation, fences, etc.). However, for vehicles in front of the ego, it always provided a split classification of each vehicle in three parts (Fig. 5.19): left side, right side, and central side (as drivable/road). This classification of vehicles was not well suited for 3D lateral lane delimiters detection.

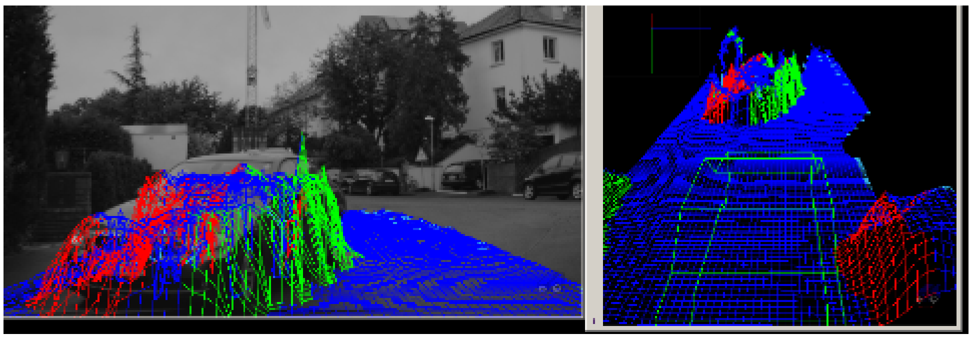


Fig. 5.19. The initial approach for classifying delimiters caused multiple labels on obstacles placed in front of the ego car (grid projected on the left image and viewed with a virtual camera). Left side delimiters – green, Right side delimiters – red

Vehicles in front are more relevant as lateral delimiters based on their relative position to the ego's trajectory. Starting from this idea and from the fact that the global grid already has the cells labelled as road or obstacles, the following steps are applied for detecting obstacle like delimiters:

1. Obstacle cells in the global grid are clustered together based on their direct adjacency.
2. For each obstacle cluster, the following features are computed and used for analysis: area A (in cells), mean elevation |E|, and mean absolute magnitude |ΔE| of the elevation gradient (only on the lateral direction). Additional filtering is applied based on these three parameters, to remove false obstacles.
3. Each cell of the obstacle is classified relative to the ego car trajectory (instantaneous circular trajectory) as *left-side* or *right-side*. The obstacle receive the majority label, which is stored in its grid cells.

In step 2, stereo artefacts are again removed: if the obstacle area and its average slope are small (so it cannot be a pole-like obstacle), then its cell are labelled as road (if their average elevation is close to the road) or emptied.

The results with the trajectory based approach for obstacle-like delimiters detection are show in Fig. 5.20. In this scenario, the ego car was taking a right turn, so the car in front is correctly classified as a left side delimiter (green).

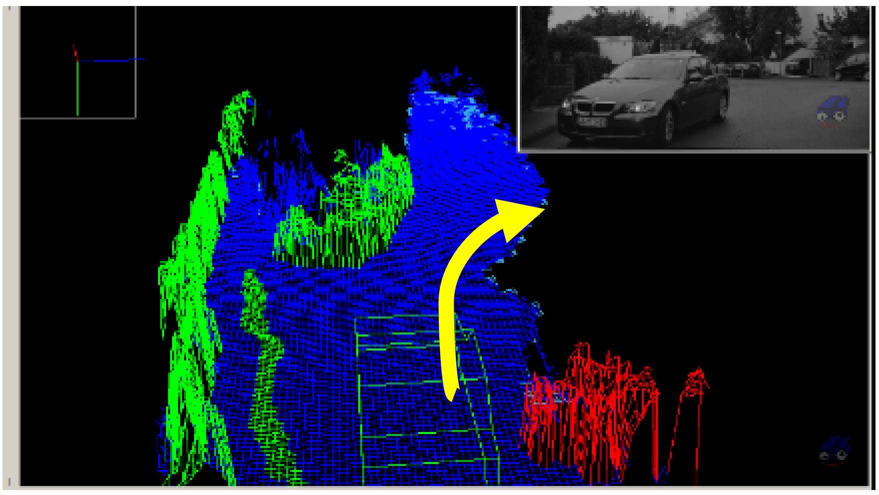


Fig. 5.20. The vehicle in front is classified as left-side delimiter when making a right turn. Left side delimiters – green, Right side delimiters – red, the grid is viewed with a virtual camera placed behind and above the ego car

5.2.4. Continuous delimiters detection (curb type)

Continuous delimiters are also detected trough a clustering approach. Grid areas that exhibit the same lateral gradient orientation and lateral gradient magnitude above a threshold, are searched in those global grid regions that are not already labelled in the large delimiters detections step (previous section).

Considering that areas with much lower slopes are searched, the risk of labelling road areas as delimiters is increased. Therefore, specific criteria will be used for filtering, adapted to the stereo uncertainty model: the gradient magnitude (slope) threshold will increase linearly with the depth and the clustering process can start only if a starting point is found at small or medium depths. The continuous delimiters detection has the following steps:

1. Scan the global grid and select a starting point that was not already classified, and has its gradient magnitude above the gradient threshold.
2. Grow a region around the starting point: iteratively add new cells to the region if they are adjacent to the current region, they were not already labelled, have a gradient greater than the threshold, and have the same gradient orientation as the starting point.
3. If the current region is greater than a minimum size, then validate it as a continuous delimiter, and label its grid cells as left-side or right-side, depending on the gradient orientation of the region.

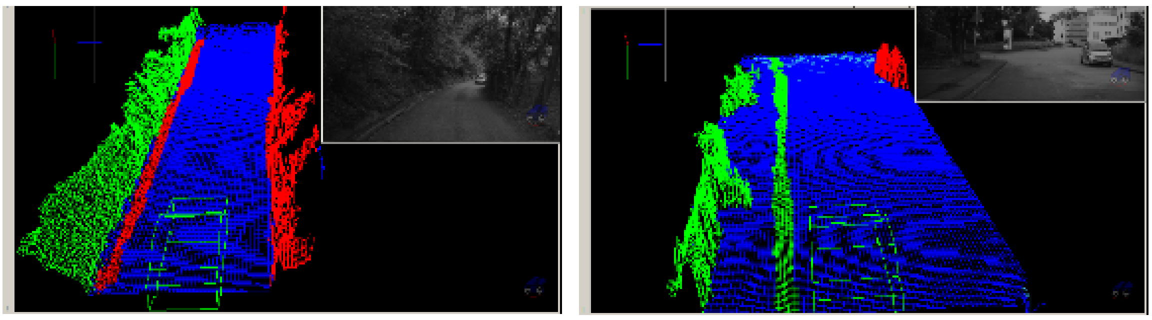


Fig. 5.21. Continuous delimiters detection: ditch in the left image, and curb in the right image. Left side – green, right side - red

Some results are show in Fig. 5.21, as curbs and ditches that are detected with the proposed method. Unlike the obstacle-like delimiters, the label shows how the continuous delimiters are sloped relative to the ego car (the ditch wall on the left side of the road, in the first image, has a decreasing elevation from right to left).

5.2.5. Performance analysis

The localization error of the delimiters was evaluated by comparing the results with lidar measurements. For natural static side delimiters, the lateral localization error for the delimiters is mostly below 1 cell, with some exceptions of 2-3 cells error.

After the development of the road profile estimation, there is no need for the least square fitting of the quadratic road model (legacy from project 1). This allowed a significant reduction for the memory required. The total memory decreased to 3.7 MB for the delimiters detection algorithm. The processing time is about 8.5 ms, on an Intel I5-2500 processor, 3.3 Ghz.

5.3. Near road features detection

The research work done by the candidate in the third project RB-2 (2015) will be presented in this sub-chapter.

The results/framework from the previous two projects were used as a starting point. The main novelty obtained is related to: method for modelling roads with complex geometry (no global model assumed, and complex lateral and longitudinal geometries are accepted), algorithms for the detection of near road interest obstacles: potholes, curbs, and lane grooves. To our knowledge, except for the detection of curbs, no methods were available in the literature for the detection of the near road interest obstacles.

5.3.1. Detection of road surfaces with complex longitudinal – lateral geometries

The road detection algorithm proposed in project 2 (previous sub-chapter) was not designed to cope with roads having a complex lateral geometry. The longitudinal (along the depth) geometry was free of a global shape, the only constraint being that, on small depth intervals, the road can be considered planar.

The goal in this project was to extend the road detection to cope with complex lateral geometries, starting from the initial road profile (project 2 approach, vertical longitudinal profile). Even when the road presents a lateral curvature or roll, the project 2 method is still able to extract a longitudinal road profile, but the labelling of the grid based on this profile is not entirely correct (at each grid depth, some road cells might be mislabeled as obstacles, see Fig. 5.22.a). However, even with laterally curved roads, the longitudinal road profile helps identifying at least some of the roads cells at each depth (except when occlusions are present).

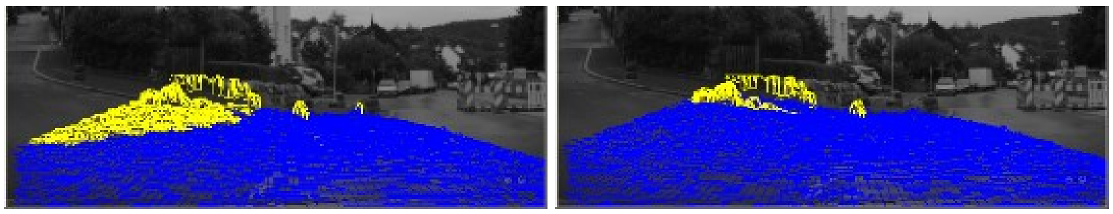

a. b.

Fig. 5.22. a. Example of road mislabeling when a strong lateral curvature / slope is present, toward the left. b. Results after the lateral road refinement step. Grid is projected onto the image (blue – road, yellow – obstacle)

The solution proposed starts from the partial labelling of the road cells in the grid, by extending the road area on each grid row (depth). In addition, it deals with missing data / true obstacles.

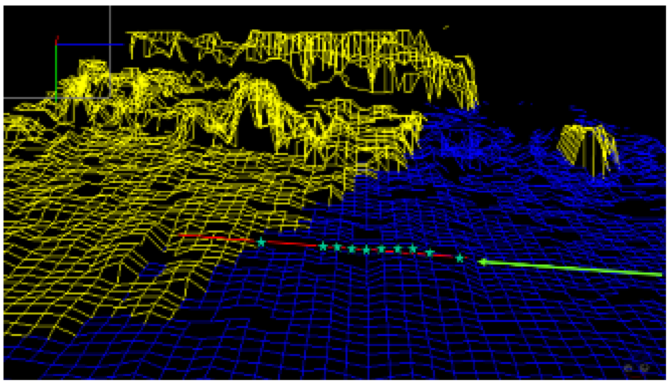


Fig. 5.23. Scanning from right to left a grid row: a subset of road cells (green stars) are used to compute the local lateral road profile (red line, lsq-fit). Neighboring obstacle cells from the same row will be re-labelled as road if they are close to the local lateral road profile. Grid is viewed with a virtual camera (blue – road, yellow – obstacle)

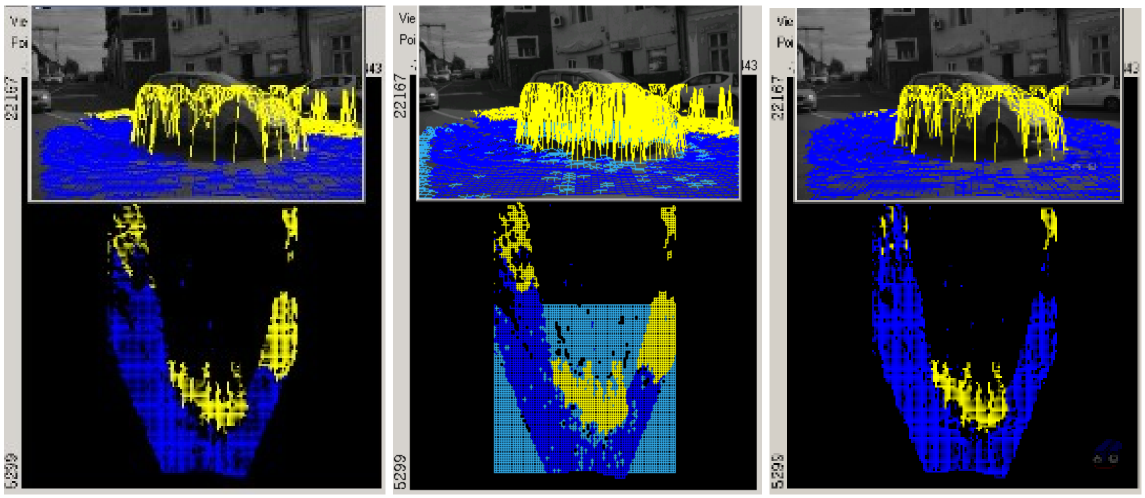
Each grid row is scanned in both directions (left-right, and right-left, see Fig. 5.23):

1. Scan the row starting from one of the lateral extremities, and store road cells in a queue. Stop if the queue is below a certain size *Pat_Size*. Remark: road cells do not have to be strictly adjacent.
2. If the queue has *Pat_Size* cells, then compute the local road profile by fitting a line

(lsq-fit) to the cells in the queue. Continue scanning, and for each grid cell do the followings:

- If the current grid cell is *Road*, then the cell is added to the queue
- If the current grid cell is *Obstacle*, then
  - Compute the road elevation (from local road profile) at the cell location
  - Compute the absolute difference between the road elevation and the cell elevation
  - If the elevation difference is below a threshold, then the cell is relabeled as *Road* and added to the queue
- If the current grid cell is *Empty*, then the cell is labelled temporary as *Road* (*Road_Fill*) and its elevation is set to the local road profile elevation. Remark: the reason for this will be explained later (longitudinal road refinement step).
- Re-computing the local road profile: this is done only if *New_Points* cells have been added to the queue, from the moment of the last re-computation. In this situation, the oldest *New_Points* cells in the queue are removed (a sliding window behavior). This is intended as a safeguard against climbing on obstacles (the most recent cells added do not contribute immediately to the local road profile).

The results of the lateral road refinement are promising (typical performance shown in Fig. 5.22.b), except for scenarios where the road surface is not dense or quasi-dense (the lateral refinement performs poorly if the 3D road data is scarce), or there are large occlusions in front of the ego vehicle (example in Fig. 5.24.a). The lateral refinement does not work because there are not enough road cells to compute the local profile, or these road cells are too far away from the obstacle patches that require relabeling as road.



a. b. c.

Fig. 5.24. a. A large occlusion in front (the large vehicle) makes the lateral road refinement less effective, b. The lateral road refinement fills the grid with temporary data (light blue, *Road_Fill*), to be used for longitudinal road refinement. c. The result after the longitudinal road refinement, and the removal of temporary road cells (*Road_Fill*). Grid is projected onto the image, on top row, or a top view is shown on the bottom row (blue – road, yellow – obstacle)

The solution to this issue is to repeat the refinement step, but applied longitudinally. In order to have a better support for the local least square fitting, we need to compensate for the field of view (or other occlusions) road invisible areas. This is why previously, in the lateral road refinement step, empty grid cells were labelled temporarily as *Road_Fill* (Fig. 5.24.b).

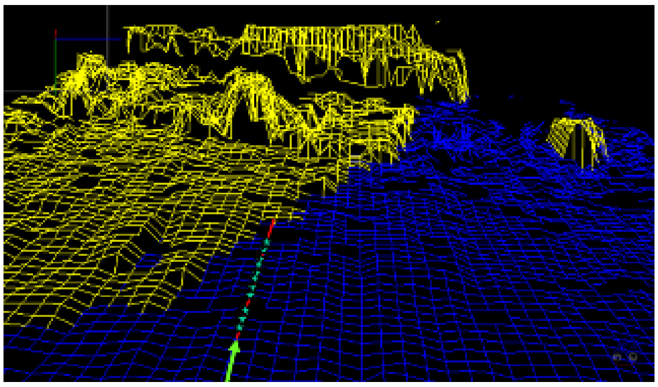


Fig. 5.25. Scanning from close to far a grid column: a subset of road cells (green stars) are used to compute the local longitudinal road profile (red line, lsq-fit). Neighboring obstacle cells from the same column will be re-labelled as road if they are close to the local longitudinal road profile. . Grid is viewed with a virtual camera (blue – road, yellow – obstacle)

Each grid column is scanned in both directions (far-close, and close-far, see Fig. 5.25), with the same approach as described for the lateral refinement, but applied the columns. Finally, the temporary Road_Fill grid cells are marked as empty. Results are shown in Fig. 5.24.c.

The road refinement is applied to both directions, so the resulting algorithm can cope with more complex road shapes, both laterally and longitudinally.

5.3.2. Potholes detection

In this section, the pothole detection approach will be presented. The initial approach (briefly presented in the next sub-section) was based only on the local elevation gradient but it had several problems. After that, the approach was modified completely by using a second order derivative, and this will be presented in detail.

5.3.2.1. Initial pothole detection approach

The first method proposal for pothole detection started from the pothole general 3D appearance as a patch that is depressed. Therefore, pothole patches should have stronger gradients, compared to the nearby road, on their borders and inner slopes. Specific thresholds will not be mentioned in this sub-section, as this approached was just an initial attempt.

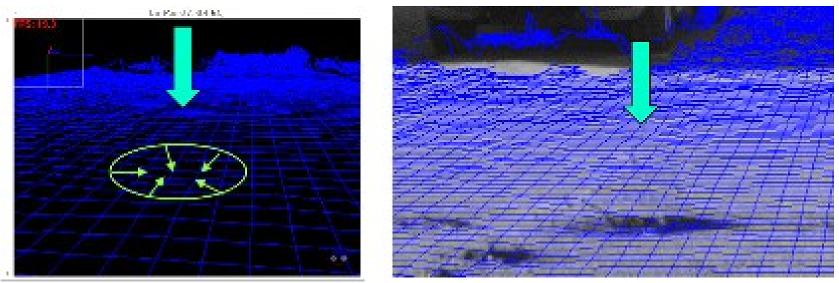


Fig. 5.26. Using a pothole model – a patch (free shape) with strong gradient on its border, and the central area depressed

The first step is to detect patches (see Fig. 5.27) with a relevant gradient magnitude (in a certain interval, adjusted with depth). Each patch is labelled / extracted separately (using breadth-first search) for further analysis.

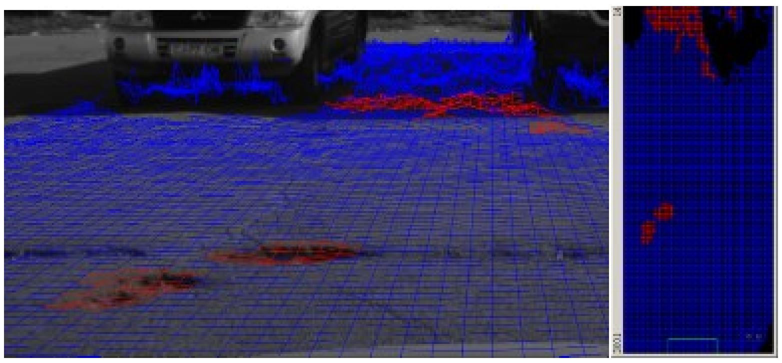


Fig. 5.27. Connected patches (red) of grid cells that have a relevant gradient magnitude are considered as pothole candidates (grid projected onto the left image, and a top view)

The next step is to analyze each candidate patch, to see how close it is to a pothole shape:

- Find the lowest and highest cells inside the rectangle circumscribing the patch => estimate the pothole height *H*
- Mark those patch cells that have an elevation around the lowest cells within *H*/3 – as inner cells (see Fig. 5.28, with green labels)

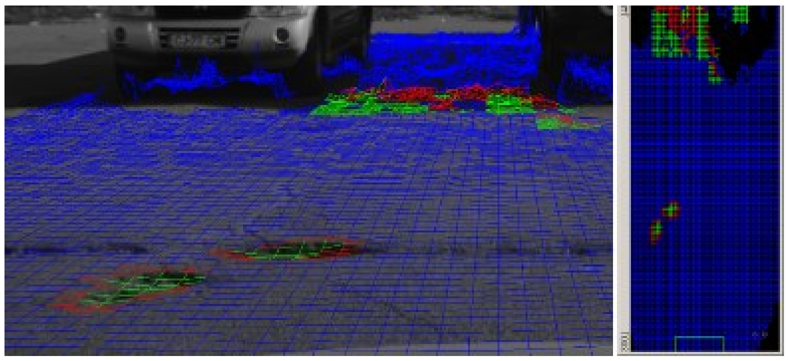


Fig. 5.28. Candidate patches processing: green cells are the inner pothole cells, while the red ones show the border of each pothole

- Count how many inner pothole cells have a road cell as neighbor versus how many have only border cells as neighbors – this empirical condition requires that a pothole is made mostly of inner cells that are enclosed by the border cells

➢ Based on the ratio of the two counters (and other criteria related to the pothole height, size, etc.), confirm or reject the pothole (see Fig. 5.29 for results)

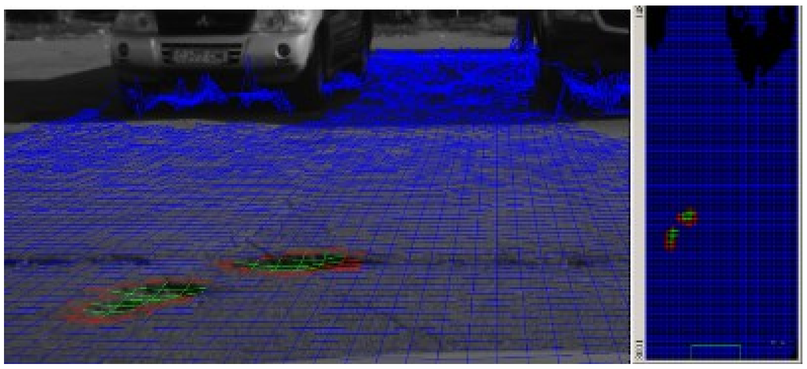


Fig. 5.29. Example of good results for the initial approach

After testing the method on several sequences, a high rate of errors (false / missed potholes) was noticed. Most of the errors are due to a local non-null pitch or roll around the pothole resulting in a relevant gradient on the road nearby the actual pothole, so the candidate patches are not always limited to the pothole position. Selecting an appropriate gradient threshold, even when adapted to the depth, is difficult. This happens on scenes where the road is not close to flat.

The first thing that needs improvements is the candidate cells detection. Most of false areas are due to local pitch / roll and they do not present curvature variation for the elevation profile. The idea is to use this feature to rule them out. Using the local curvature can help for a better detection of potholes, and, starting from this idea, the algorithm described in the next sub-section was developed.

5.3.2.2. Pothole detection based on the second order derivative

The algorithm for pothole detection, developed in this project, is presented next. The main steps are:

1. Compute the Laplacian for each grid cell, where elevation data is available.
2. Mark the *candidate pothole cells*: those grid cells that have a negative Laplacian value, with an absolute Laplacian value higher than a threshold (adapted to the depth). Concave patches (as seen from above) are labelled in this way.
3. Select pothole candidates (patch size should be at least *MIN_PHOLE_SIZE_CELLS* = 5 to be valid, and at most *Max_Nr_Potholes* = 250 are selected). Scan the grid and, if a pothole candidate cell P_c is found, then start a breadth-first search (queue based, that stores the grid location and label: *inner* or *border* pothole cell) to grow the region of the pothole. Place P_c in the queue. Repeat the following steps until the queue is empty:
    a. Extract the first cell P_f from the queue.
    b. If P_f is an *inner* cell then scan all of its neighbors in the grid (4-point neighborhood) and:
        - add to the queue those neighbors (not already in the queue) that

are labelled as *candidate pothole cells*, and label them as *inner* cells.

- add to the queue those neighbors (not already in the queue) that are not labelled *as candidate pothole cells*, (have a Laplacian value outside the interval of *inner* cells, a zero crossing – the curvature changes to flat or convex) and label them as *border* cells.

4. Various features are computed for each pothole candidate, features that will be used for estimating the confidence: pothole lateral / longitudinal limits, the near road plane used for computing the pothole height, roll and pitch angles, etc.

5. Perform tracking between frames: each pothole candidate is associated with candidate detections from the previous frame, and a life-time is computed for each pothole.

6. Compute the confidence of each pothole as the product of the confidence values computed for each pothole feature. Those potholes with a confidence values higher than *MIN_PHOLE_CONF* (=0.5) are considered valid detections and provided in the ADTF output. At most *Max_Nr_Potholes_Out* (=5) potholes are outputted.

Additional explanations are provided next.

Sensing the local curvature – pothole candidates

Sensing the local curvature can be done with any operator based on second order derivatives. Two alternatives were investigated, the Laplace operator and the second order directional derivative.

The Laplace operator is a differential operator computed as the sum of the second order derivative on each of the two main axes:

$$\frac{\partial^2 I}{\partial x^2}+\frac{\partial^2 I}{\partial y^2}$$

The second order directional derivative involves computing the derivative of the gradient on the direction (n) of the local gradient. It is equivalent with:

$$\frac{\partial^2}{\partial n^2}(I)=\frac{\frac{\partial^2 I}{\partial x^2}\left(\frac{\partial I}{\partial x}\right)^2+2\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\frac{\partial^2 I}{\partial x\partial y}+\frac{\partial^2 I}{\partial y^2}\left(\frac{\partial I}{\partial x}\right)^2}{\left(\frac{\partial I}{\partial x}\right)^2+\left(\frac{\partial I}{\partial y}\right)^2}$$

When applying these formulas to the grid, the coordinates system is represented by the two axes: depth and lateral position, and the grid elevation is the value of the function to be derived. The derivative discrete nucleus is [-1 0 1] – lateral, $[1\ 0\ -1]^T$ – depth, applied on each direction.

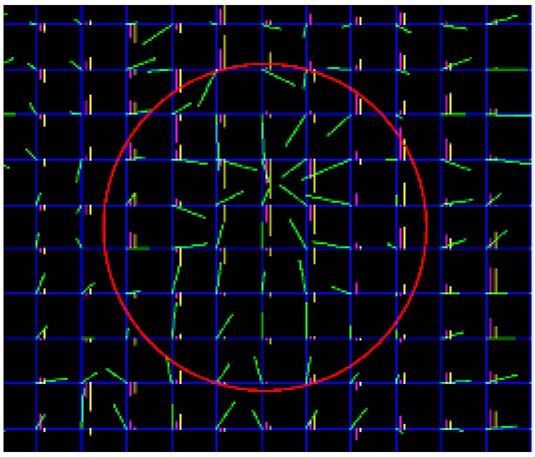


Fig. 5.30. Various derivative values displayed on a top-view pothole region of the grid (the same pothole as in Fig. 5.26): the gradient vector – green lines, the Laplace value (yellow) and the second order directional derivative (magenta) are displayed as vertical lines (upward for positive, and downward for negative values) near each grid vertex

Depressed areas have a negative value for the second order derivatives (same curvature, concave if seen from above) but, close to their borders, the second order derivative presents a zero crossing (the value is small in magnitude and outside the depressed areas it becomes zero or positive).

The Laplace operator was selected because it is less strict (provides more cells in the pothole region), faster to compute, and is defined everywhere (the second order directional derivative cannot be computed where the gradient magnitude is null). First, we select those cells with a negative value for the Laplacian and the magnitude of Laplacian greater than a threshold (variable with the depth, related to the stereo elevation uncertainty). Candidate cells are selected for further analysis (see Fig. 5.31).

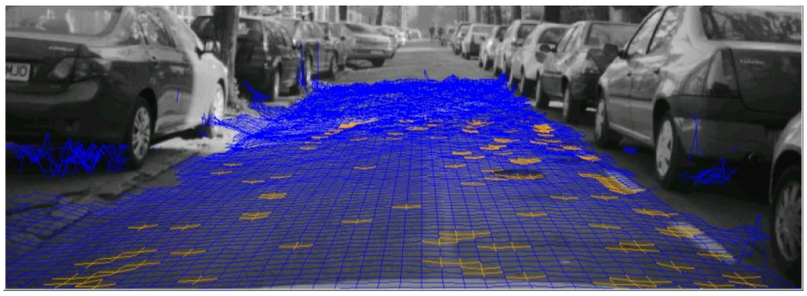


Fig. 5.31. Candidate cells (orange) are selected with the Laplace operator

The border of each potential pothole is searched. Each region of connected candidate cells is expanded (region growing, breadth-first search) with potential cells that are placed on the border of the region, and have smaller magnitude but still negative for the Laplacian, or zero, or the first encountered positive Laplace (stop condition for the current location in the breadth first search). The result of this step consists of several pothole candidates, each made of a set of inner cells and border cells (see Fig. 5.32).

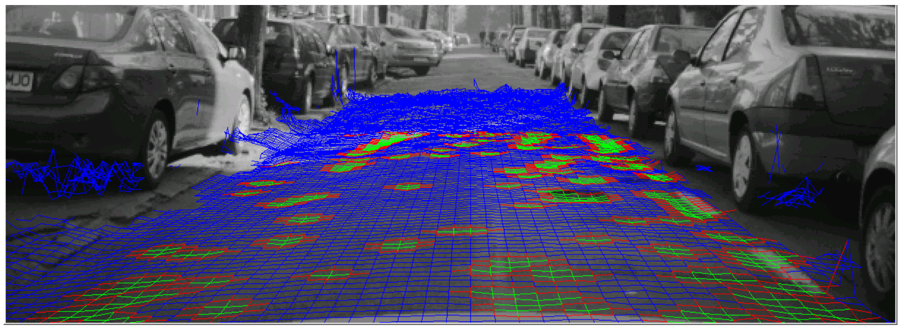


Fig. 5.32. Pothole candidates: connected regions made of inner cells (green) and border cells (red)

The next step is to evaluate various features for each pothole candidate, and, based on these features, to decide which potholes are real or not.

Estimating the pothole features

The first features to be evaluated are the pothole limits on the depth and lateral direction.

Another feature that is evaluated for each pothole is the ratio between the mean of the absolute value of the second order derivative along the depth and the mean of the absolute value of the same derivative along the lateral direction. This is because we expect a certain equilibrium between the curvatures of the pothole along the two axes.

The pothole height was estimated by evaluating the plane that would "cover" the pothole. Initially, the plane was fitted (lsq-fit) to the cells that are the borders of the pothole, but, in this case, the height was often under-evaluated. A better solution is to fit the plane to the road cells that are adjacent to the pothole border. Then, the distance from each inner pothole cell to the plane is evaluated, and the height of the pothole is the maximum value of this distance. This method of evaluating the pothole height works better because it cancels out the effect the local roll and pitch of the road. The roll and pitch angles of this plane are evaluated.

A tracking scheme for potholes

Given the high rate of false positives for detecting potholes, another feature that can help is the temporal persistence of the results: the lifetime of a pothole is the number of consecutive frames for which the pothole was detected.

A tracking scheme was developed, that can also deal with fragmentation or joining between potholes:

- A global set of potholes is maintained: each with a life time (in frames) and features (position) updated with the ego motion
- Each candidate pothole detected in the current frame is associated with one (or many, or none) global pothole. The global set is updated (lifetime increased for potholes with a correspondent).
- Association takes into account fragmentation or joining between potholes:
    - If both potholes (previous / current frame) have their cells stored, then the measure of overlap is computed as the number of cells that overlap.

- If the coordinates of the cells are not available (for large pothole), then the depth / lateral limits of each pothole are used to compute the overlap area.
- If a current frame candidate pothole overlaps more potholes from the previous frame, then the overlap area is the sum of the individual overlaps.
- Association is valid is the overlap ratio is larger than a threshold

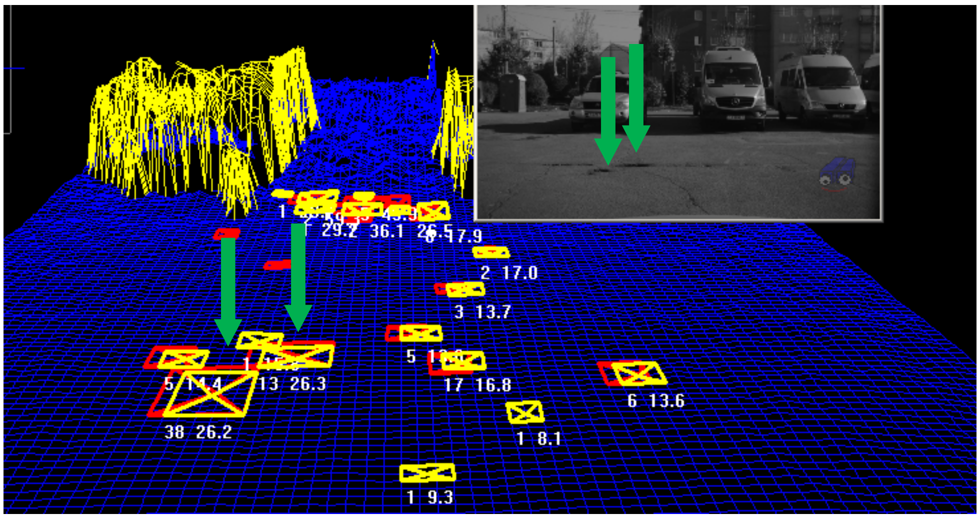


Fig. 5.33. The association process: with red, the candidate potholes from the previous frame (ego motion compensated, with yellow the candidate potholes from the current frame. For each candidate in the current frame, the lifetime and height are displayed. For the two potholes depicted by the green arrows, they have a lifetime of 38, respectively 13 frames (it is a very low speed sequence), and a height of 26.2, and 26.3 mm.

The pothole confidence score

A confidence score is estimated for each pothole based on the difference between its computed features and their expected values. The following features are taken into account: lifetime, area, height, pothole plane roll / pitch angles, aspect ratio, distance, filling ratio. Potholes having the confidence score above a threshold are considered valid. Example of results in Fig. 5.34.

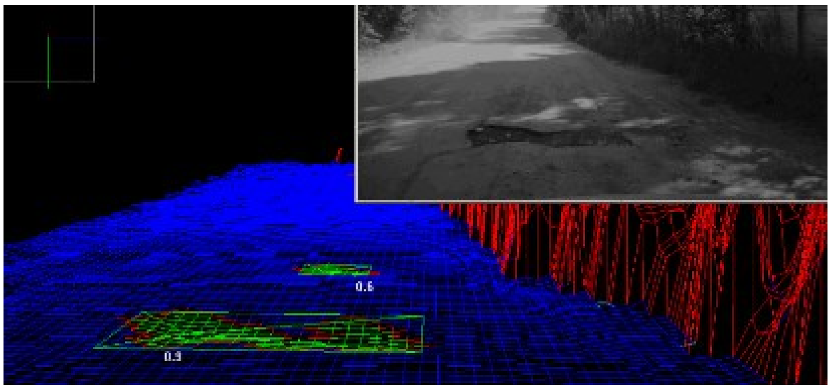


Fig. 5.34. A real pothole (the large one) with a confidence of 0.9, and a false pothole caused by the light / shadow artefacts in stereo (confidence 0.6)

Still, the discrimination between real potholes and potholes from stereo artefacts is not robust, unless a high confidence (0.8-1) is used => some real potholes are missed, or detection is done only for larger potholes (size and height) => adjust confidence scores.

5.3.3. Curb detection

Existing curb detection methods, which are stereovision based, rely mainly on fitting a global parametric model to the 3D data, in order to extract the location and the shape of the curb. The simplest model proposed is a line [50], [51]. A polygonal model was proposed in [52] that models curbs as a chain of segments (several meters each). This was a first step towards modelling curved curbs. Later in [45] and [53], a cubic polynomial model was proposed. This model worked better for curved curbs and it even allowed curvature variation (as long as it fits a cubic polynomial), but it was not appropriate for long curbs with a more complex geometry. A more general model was proposed in [54], as a cubic-spline. Still, it required the curb to follow the shape of a cubic polynomials on selected intervals. Approaches presented in [52], [53] and [54] were the some of the candidate's achievements during his PhD stage.

In this project, a curb detection method was developed that does not rely on any global parametric model, it provides curbs located with increased accuracy and also estimates confidence values for each curb location. The curb detection algorithm has better localization for curbs, initially at grid cell level – thinner curbs, 1-cell width, and then at sub-cell accuracy. This will require a different representation for curbs, as poly-lines, instead of elongated patches in the grid (as proposed in the previous sub-chapter). Furthermore, in order to achieve sub-cell accuracy, lines will be fitted to sub-chains of curb cells to compute the sub-cell lateral position / height for each curb location.

Each algorithm step was developed with various measures to cope with the stereo noise currently present in the 3D data. In Fig. 5.35, different perspective views of the grid data (with temporal integration, no surface filtering in order to avoid curb widening) at various distances are presented, when the curb is shadowed (poor quality of the stereo data).

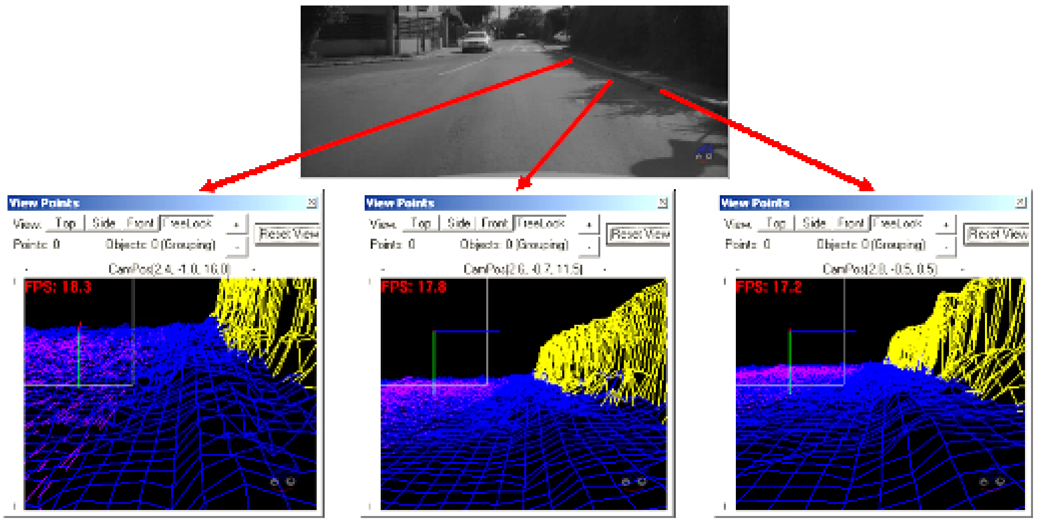


Fig. 5.35. Perspective view of areas of the grid along a curb, at various depths (farthest on the left). Far depths present large noise in the elevation data

Next, the main steps of the curb detection algorithm will be presented.

5.3.3.1. Detecting curb candidate locations

Curb candidate locations will present a significant elevation variation in comparison with the nearby non-curb grid locations (road / sidewalk). This variation will be detected with the local elevation gradient:

- Compute the lateral gradient of the elevation, using a convolution nucleus applied along each grid row. The sign of this gradient will be specific to each type of curb: + for left side curbs, and - for right side curbs (the elevation stored in the grid is the value of the vertical location, and the vertical axis is oriented downwards).
- Select those cells with a gradient magnitude above a threshold (see Fig. 5.36.a). Evaluate the threshold as a function of the stereo configuration and depth. Additionally, an upper limit is imposed on the gradient, to avoid the selection of obstacle boundaries.

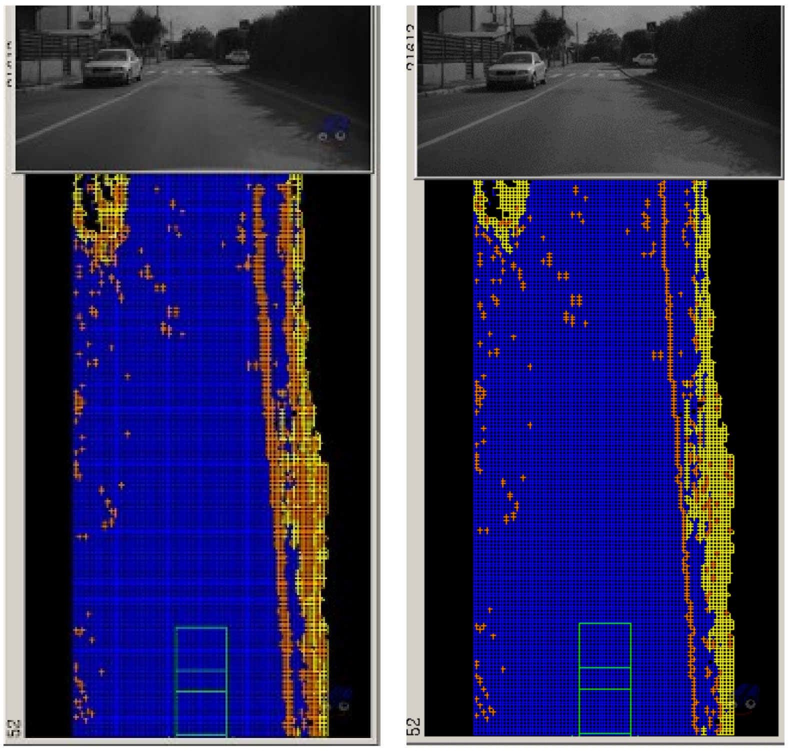

a. b.

Fig. 5.36. a. Grid cells were the lateral gradient is above a threshold (with orange). b. After performing non-maximum suppression, and the removal of large gradients from obstacles, the curb profile becomes thinner

The next step is to remove those cells where the gradient does not present a local maximum. This is done by performing non-maximum suppression: only cells with a gradient magnitude greater than their neighbors remain (see Fig. 5.36.b).

5.3.3.2. Curb chains extraction

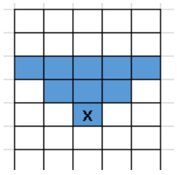
This step consists of extracting chains of candidate cells. The noise along the curb profile and potential gapes must be taken into account.

The extraction is a breadth first search: the coordinates of the grid curb cells (row, column) are extracted in a queue. Scan the grid, from close to far, and repeat the following steps until no candidate curb cells remain:

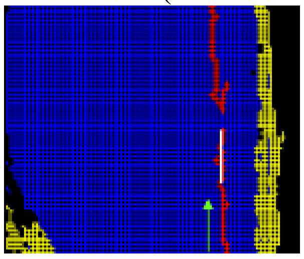
1. Find a candidate curb cell, added to the queue. Left side curb candidate cells are valid if they are placed in the left half of the grid and have an elevation decrease from left to right (test the gradient sign). Right side candidate curb cells…the other way around.

2. Iteratively, while the queue is not empty, add new curb cells to the chain:

a. Select the first cell P(*row*, *col*) from the queue.
b. Search in its grid neighborhood for others candidate curb cell (not already selected). The neighborhood is oriented towards greater depths (see below, with light blue, X is the current position), including cells from the next two rows, *row*+1, and *row*+2.



c. Add to the queue those candidate cells (from sub-step b.) that have a similar road side elevation profile with cell P. The road side profile of a curb cell, in this sub-step, is provided by the elevations of the next 5 grid cells on the road side of the curb, on the same row. This condition helps when there is a lot of noise in the stereo data around the curb (it avoids climbing on the sidewalk).



d. If the queue is empty, then there might be a gap in the curb 3D data (figure above):
   - Compute the local linear profile of the curb (RANSAC and least squares fitting) on the last cells added to the queue
   - Search new curb candidate cells, on the line direction, up to a maximum distance

Chains of candidate curb cells that are longer than a threshold are considered for future refinement. So far, we have extracted chains of grid cells (locations with integer grid coordinates for row / column), that also have the curb type (left side or right side). We have to compute a more precise curb profile (3D – longitudinal, lateral, vertical, height)

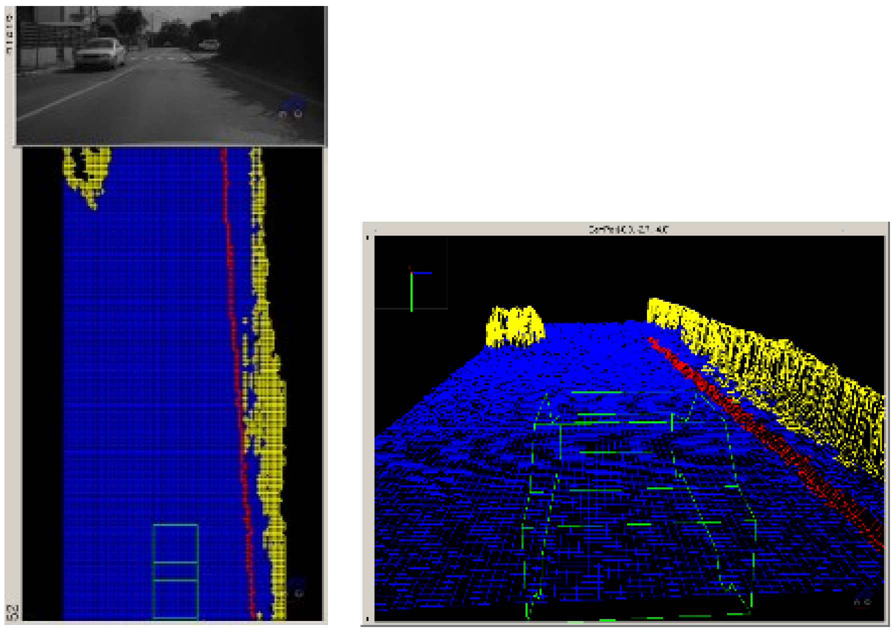


Fig. 5.37. A chain of curbs cells (red) extracted from the grid (top view and perspective view)

5.3.3.3. Curb lateral - longitudinal profile refinement

Better accuracy is only possible with certain assumptions regarding the model of the curb. There must be a tradeoff between a global model (ex. a high order polynomial) and the basic chain of curb cells (integer grid locations).

The assumption made is that the curb can be approximated locally by a linear model (quadratic also tested, but large errors are not filtered well). This allows averaging measurements and getting a better localization. Also, small gaps where curb cells are missing will be filled.

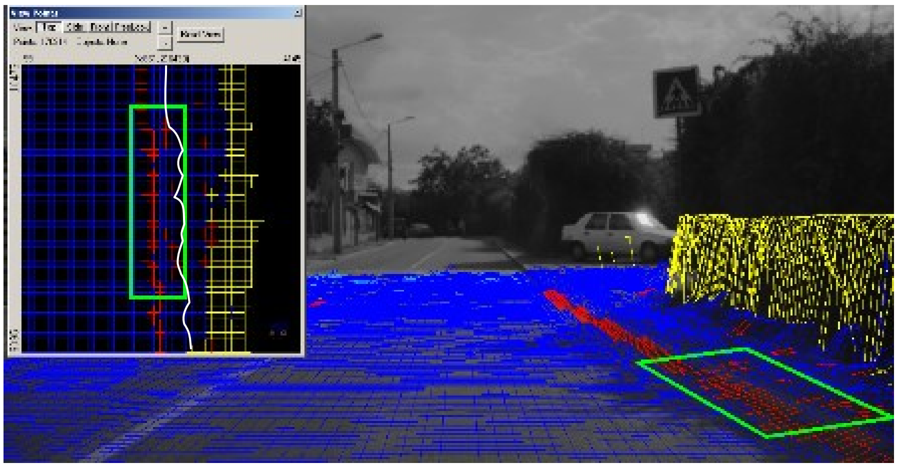


Fig. 5.38. If the curb profile is fuzzy and not close to a step, then false curb candidate points are added to the queue (green polygon area). The white path shows the curb candidate cells, for the current curb chain, that are the closest to the road side

In the early development of this step, least squares fitting of a line was applied on the local patch of the curb (a symmetric interval in the queue of candidate cells, centered on the current cell). In curb areas where noise exists (multiple curb candidate cells on the same grid row), the approximation will sometimes be biased by the noise. As a countermeasure, RANSAC was used, but the results were not always good, especially when too much noise was present in a certain curb patch (a small zigzag effect, up to 5-10 cm, on the refined lateral position).

However, at least on the available sequences, the noise was most of the time on the sidewalk side of the curb. So, before applying the line fitting, if there are multiple curb candidate cells on a grid row (Fig. 5.38), only the cell closest to the road is considered for line fitting.

For a candidate curb chain, the lateral – longitudinal refinement works as follows (actually, for each grid row/depth a refined lateral position of the curb is computed):

- For each grid row, the lateral position of the curb is extracted, as a column value, on rows where at least a candidate curb cells exists (for the other rows the position is considered invalid). If there are multiple curb cells on a row, then the curb cell that is the closest to the road side of the curb is selected.

- The refined lateral position is computed by fitting lines (lsq fit) to a window centered on each position (invalid lateral positions are not considered in the lsq fitting). The sub-cell position is computed from the line equation, for each curb location. A confidence score is computed for each curb cell, showing the confidence of the lateral positioning.

The result of this step is the lateral – longitudinal curb profile (Fig. 5.39), consisting of sub-cell accuracy lateral positioning for the curb on each grid row where the curb data is available.

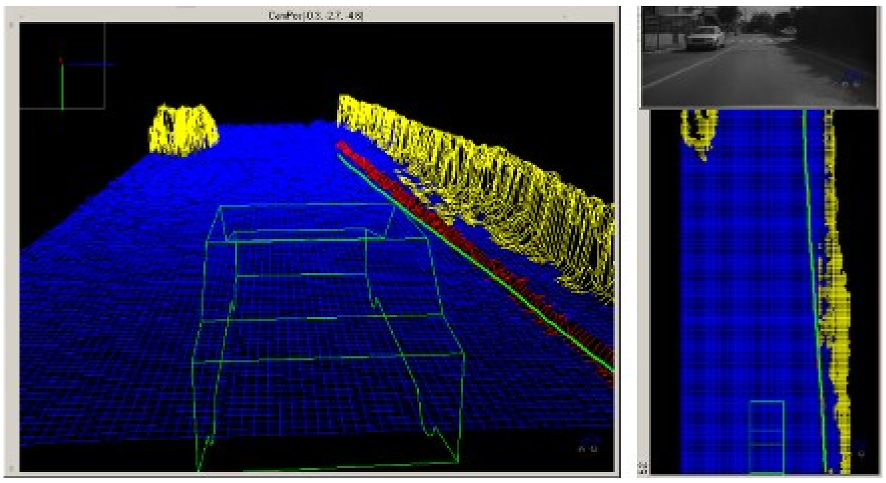


Fig. 5.39. The lateral – longitudinal curb profile (the green polygonal line), consisting of sub-cell accuracy lateral positioning for the curb on each grid row where the curb data is available. The vertical profile is not computed – that is why the projection on the perspective view is not precise

5.3.3.4. Curb vertical profile - road / sidewalk. Extraction and refinement

Each vertical profile, for the road and the sidewalk side, is computed independently. First, the road elevation and the sidewalk elevation is extracted, and then, these profiles are refined with line fitting.

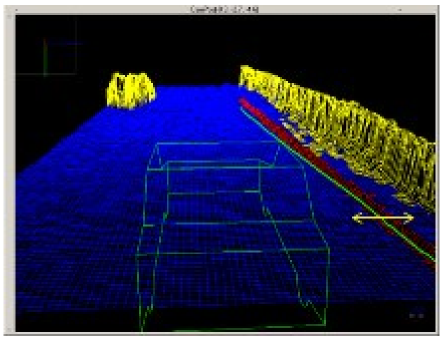
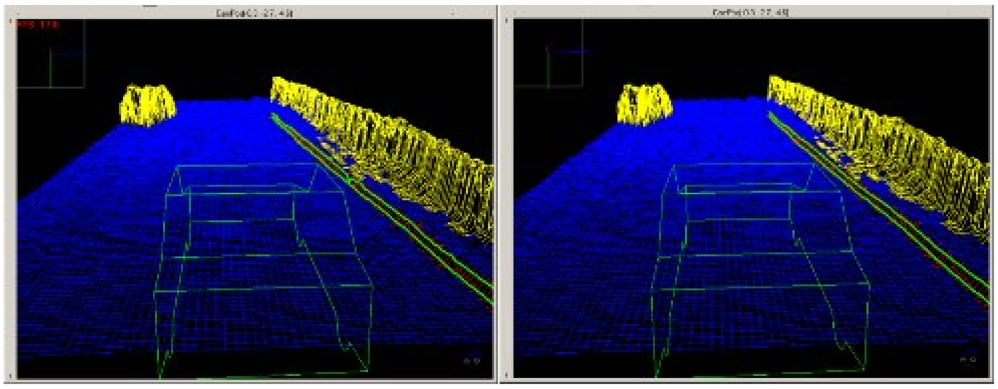


Fig. 5.40. The vertical profile of the curb (road and sidewalk sides) is extracted from the grid elevation, on each side at a certain lateral distance from the curb position

For each curb cell, the road/sidewalk elevations are extracted from the grid cells placed laterally, on each side of the curb, at 2(or 3) cells distance (Fig. 5.40).



a. b.

Fig. 5.41. a. The road and sidewalk elevation profiles extracted from the grid (noise is present in some locations), b. The road and sidewalk elevation profiles after refinement with line fitting (most of noise removed)

The vertical profiles have noise (Fig. 5.41.a), or might have no elevation (invalid) for some locations. The refinement is similar to the one performed for the lateral-longitudinal profile: the refined elevation is computed by fitting lines (RANSAC + lsq fit) to a window centered on each position in the curb chain. The elevation values for each curb position are recomputed (Fig. 5.41.b) based on linear approximation (invalid values are replaced also).

During the vertical profile refinement, two confidence scores are computed for each cell, for the road and sidewalk profile. For both profiles, the confidence score depends on: the ratio between the size of the consensus set (inliers of the RANSAC solution) and the number of points for RANSAC, and the ratio between the local curb height and the elevation uncertainty from stereo at that location.

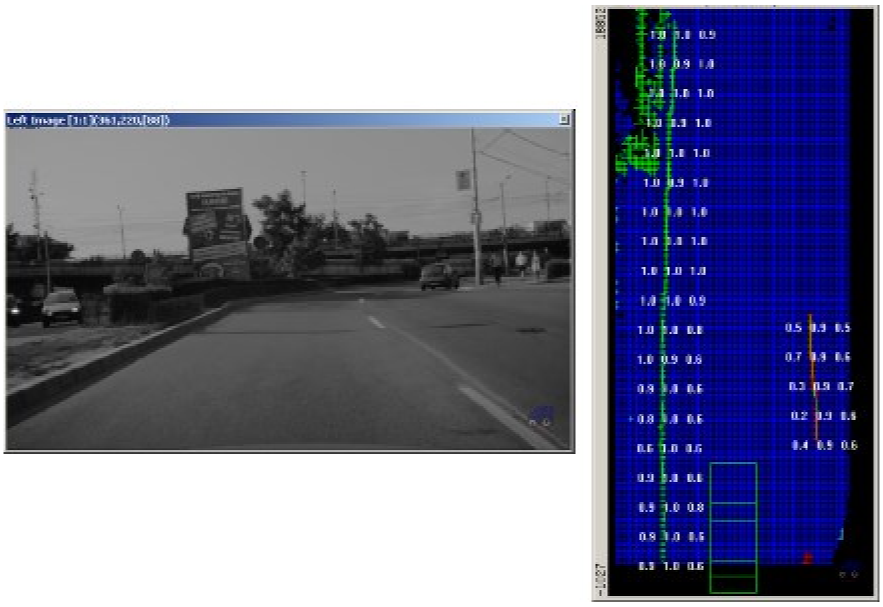


Fig. 5.42. Confidence scores for the curb profile: three values are displayed (only for every 1 of 10 cells, in this image) for the sidewalk elevation, lateral-longitudinal position, and for the road elevation. The curb on the left is real, while the curb on the right is a false detection

After this final step, curbs are described completely with the computed profiles. Sample results are presented in Fig. 5.43. Evaluation on various scenarios showed that false positives have a low confidence score.

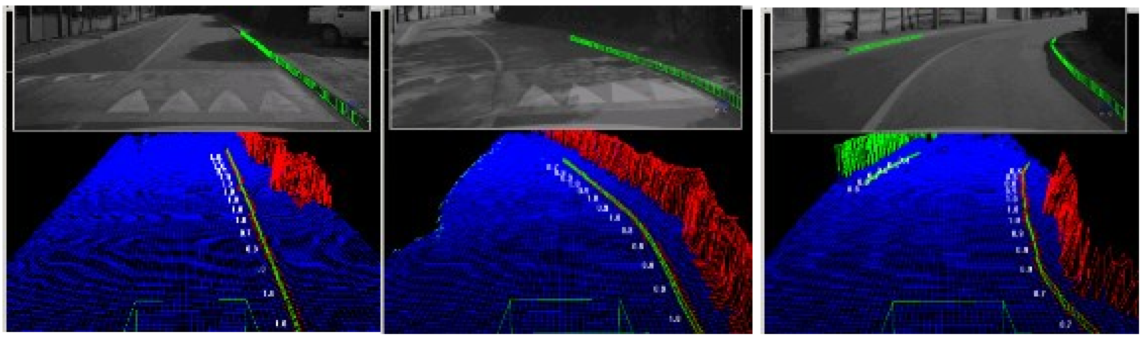


Fig. 5.43. Sample results for curb detection (road profile confidence is displayed)

5.3.4. Lane grooves detection

Lane grooves have a specific lateral elevation profile. This profile should be encountered over a large interval of depths, for the lane groove to be relevant.

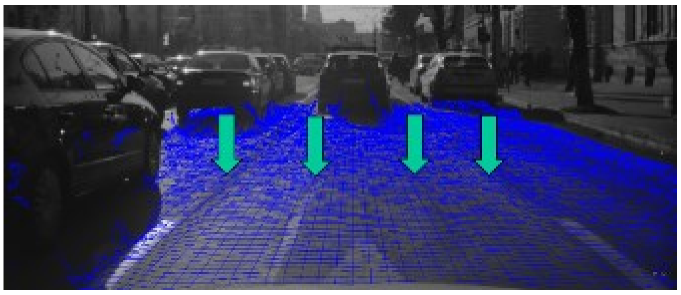


Fig. 5.44. The lateral sides of the two lane grooves are marked with the green arrows.

A linear lane groove model was assumed, oriented close to the ego car’s longitudinal axis.

Lane groove grid cells are labelled based on their lateral gradient of the elevation. Then, grooves are detected through a voting scheme based on the vertical projection of the labelled grid (along the columns). The steps of the lane grooves detection approach are:

1. Find the cells from the lane groove sides by thresholding the lateral gradient: if the cell gradient is within a specific interval, then mark it as a lane groove side. Depending on the gradient sign, each cell is marked as a *Left_Side* or a *Right_Side* cell (Fig. 5.45.a and b-top).

2. The vertical projections (along the grid columns)/histograms for each type of candidate cells are computed (Fig. 5.45.b-bottom).

3. Relevant peaks (Fig. 5.45.b-bottom, the peaks marked with a blue vertical stripe at their base) are detected in each of the two vertical projections/histograms. A relevant peak represents the borders of a lane groove, and has the following properties in the vertical projection: it must be larger than a threshold, must be a local maximum and, also, a sharp maximum.

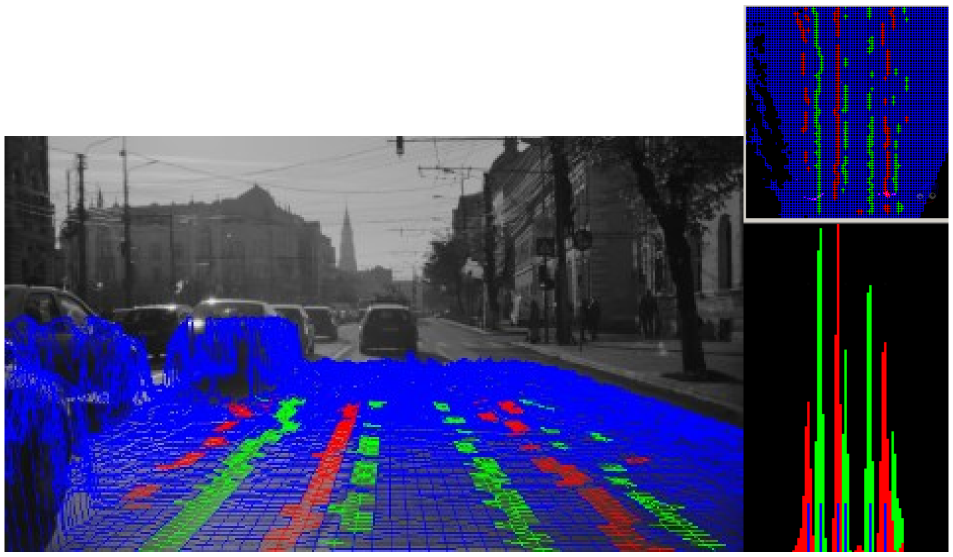

a. b.

Fig. 5.45. a. Image projection of the grid with the candidate lane groove sides (**left**/**right**, depending on the lateral slope), b. top: the top view of the grid, bottom the vertical projections, overlapped, green for the left side cells, and red for the right side cells. Peaks that are relevant have a dark blue stripe at their base

4. Lane grooves are extracted (as pairs of consecutive relevant peaks having a (*Left_Side* - *Right_Side*) configuration. The left lane groove is searched from left to the middle of the grid (ego car's longitudinal axis), among the relevant peaks, until the first pair of relevant peaks is found, if any. The right lane groove is searched similarly, but from the right to the middle of the grid, in the set of relevant peaks. A lane groove is valid only if its width is within a specific interval.

5. In the output grid, the lane grooves are labelled using particular labels (see Fig. 5.46).

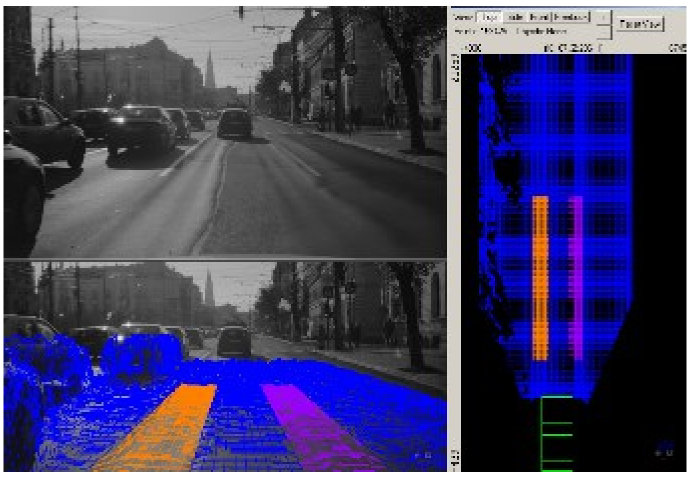


Fig. 5.46. The two lane grooves (the left one with orange, and the right one with magenta) are labelled in the grid, as rectangular areas bounded by their relevant peaks positions on the lateral

5.3.5. Performance specifications

Quality of the results was evaluated on various scenarios, and the strengths and limitations of each algorithm were identified. The memory and processing time specifications (all the steps/methods described so far) are the following:

- Memory size: less than 7 MB
- Processing time per frame is very low (all algorithms active): 10-13 ms (Intel Core i5-2500 CPU), out of which: building the current frame + full road modelling 2 ms, potholes detection 0.5 ms, curb detection – 1.5 – 2.5 ms, lane grooves detection – 0.2 – 0.3 ms

6. Lidar and camera based environment perception

A multi- lidar and camera system is used for environment perception in the ongoing Horizon 2020 project UP-DRIVE. The candidate research work aims at developing a real-time obstacle detection method with multiple synchronous lidars. The main contributions of the author are the development of a method for road surface detection from the multi-lidar data, and the detection of oriented obstacles using a voxel representation of the multi-lidar 3D measurements. So far, an original low-complexity method for computing the orientation of 3D obstacles was published at an international conference (*ICCP 2018*, Table 1.1, first row). This contribution will be presented in this chapter.

6.1. Detection of oriented obstacles

A low complexity approach for computing the orientation of 3D obstacles, detected from lidar data, is presented in this sub-chapter. The proposed method takes as input obstacles represented as cuboids without orientation (aligned with the reference frame). Each cuboid contains a cluster of obstacle locations (discrete grid cells). First, for each obstacle, the boundaries that are visible for the perception system are selected. A model consisting of two perpendicular lines is fitted to the set of boundary cells, one for each presumed visible side. The main dominant line is computed with a RANSAC approach. Then, the second line is searched, using a constraint of perpendicularity on the dominant line. The existence of the second line is used to validate the orientation. Finally, additional criteria are proposed to select the best orientation based on the free area of the cuboid (on top view) that is visible to the perception system.

6.1.1. Introduction and Related Work

In the field of autonomous driving, the sensorial perception system is one of the key elements. Common sensors used are cameras, mono or stereo, radars, or lidars, often in combination to sum up the strengths of each sensor type.

Lidar sensors have one or multiple layers used for scanning the surrounding environment. A number of accurate 3D measurements are provided for each layer, with a certain angular resolution. Recently, lidar sensors with 16 or 32 layers have been introduced, as an intermediate solution between the common 1-, 2-, 4- layer and the expensive 64-layer lidar.

Various obstacle models are proposed in the literature, the cuboid (parallelepiped) being the most common one. It allows a compact representation and fast higher level processing (such as tracking). Other potential models are simplifications of the cuboid model, such as lines or rectangles (on the bird's eye view of the environment), or more complex such as poly-lines and clusters of grid cells/voxels. The cuboid is a good compromise for autonomous driving applications, as the interest obstacles (vehicles, urban structures, pedestrians etc.) can be delimited with this simple geometric shape. The cuboid model is less appropriate for obstacles with irregular shape, and, when the obstacle axes are not aligned with the reference system. For the latter issue, the solution is to find the right orientation of the cuboid in order to align the cuboid sides with the obstacle boundaries.

Most lidar obstacle detection systems rely on the L-shape model for detection, and this model is usually applied in the polar coordinates representation, along the same layer of a lidar. In [40], lidar 3D measurements are first clustered by scanning along the same layer and grouping points in the same cluster, if they are close in angular distance. Then, for each cluster the L-shape is extracted by initially fitting a line to the extreme points of the cluster. Iteratively, the line is split until an L-shape is obtained, which provides both the location and the orientation of the obstacle. Tracking is further employed to smooth the results and to estimate dynamic features. 4-layer lidars are used, that are placed in the front bumper, and oriented quasi parallel with the road.

The approach from [41] formulates the fitting of the L-shape as an optimization problem, without using the polar representation. A measure is proposed, in a least-squares sense, for how good the L shape fits a cluster of points representing an obstacle. The best fit of the L-shape is then searched by computing the fitness measure for each possible orientation. The method required about 4 ms of processing time for each obstacle, mainly due to the exhaustive search needed.

Points are clustered in potential obstacles in [42] by using angular and range thresholds, directly in polar coordinates. Each cluster is fragmented into edges by applying the iterative end point algorithm, initially proposed in [43]. For each segment, the line is computed with generalized least-squares fitting and used as primitives for tracking.

Either a line or a rectangle is fit to the cluster of points from an obstacle, with least-squares, in [44]. The rectangle fit is preferred if available. Tracking is then used for temporal filtering and smoothing. 1-layer lidars are used.

In [45] the lidar scan data, from one layer, is first segmented into obstacle clusters using the adaptive point break algorithm proposed earlier [46]. For each cluster, two perpendicular lines are computed with least-squares minimization. Each point from the cluster is considered as pivot (the common point of the two lines) and the best pair of lines is found. The point scan order must be known, and the average processing time is 1.5 ms for each obstacle cluster.

A low complexity approach for computing the orientation of cuboidal obstacles was developed by the candidate. The proposed contributions are as follows: a model consisting of two perpendicular lines representing the two potential visible sides of rectangular (or quasi-rectangular obstacles), model that is fitted to the obstacle boundaries using a RANdom Sample Consensus approach (RANSAC, [47]); a strategy to validate/choose the best orientation based on the occupied area of the cuboid in the bird's eye view (top view).

The method that will be presented does not require the original scan order of the points and the processing of each lidar layer individually for computing the orientation. Thus, it is a more general approach, suitable for lidars with many layers. Instead of fitting the two line model with least squares, iteratively for each pivot point, as in [45], a random consensus approach will be used: the first line is computed, and then, the second line is searched on the perpendicular direction. This allows a lower computational complexity for the method, as it will be later discussed in the results section.

Next, an overview of the proposed approach will be presented, the main steps will be detailed, and, finally, results and evaluation of the performance are presented and analyzed.

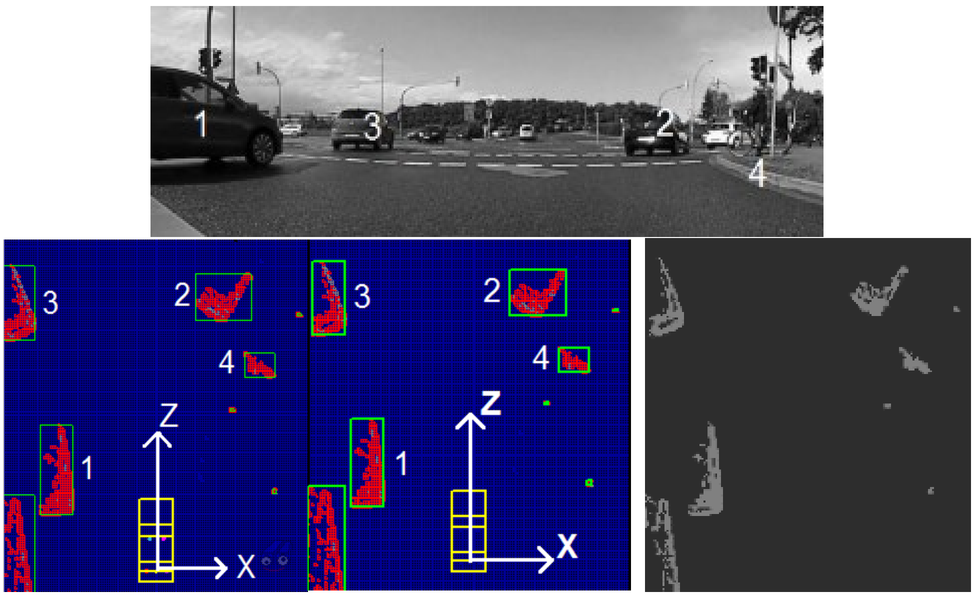


Fig. 6.1. Input of the proposed approach: a set of detected obstacles, with un-oriented cuboids (green). Top image: a camera view in front of the ego vehicle (yellow in the top view), with four relevant obstacles marked, 1,2,3 - vehicles, 4 - bicyclist. Bottom-left: the result of obstacle detection (red areas), a bird-eye view, bottom-right: occupied grid cells are shown with a medium grey shade

6.1.2. Overview of the Proposed Approach

The perception system is similar to the one described in [53] consisting of multiple lidars of 32 layers, each placed close to one of the four corners of the ego vehicle roof. Obstacle are detected using a grid based approach derived from [44].

The approach for computing the orientation of the obstacles starts from a set of un-oriented cuboids, each cuboid having its set of occupied grid locations in the horizontal plane (top view of the 3D environment, Fig. 6.1). A grid location/cell represents a square of 10 x 10 cm in the horizontal plane.

The reference system used to represent the top view will have Z as longitudinal axis and X as lateral axis (relative to the ego vehicle). For all the images in this chapter that show a top view of the scene, or the occupancy grid, the vertical axis corresponds to Z, and the horizontal to X.

The model consists of two perpendicular lines, as in Fig. 6.2. The first line L (without any relation with the L-shaped model, just a notation) is considered the dominant orientation (on the side with the most boundary cells), while the second line L_P is used to validate that the obstacle has two visible sides for the perception system.

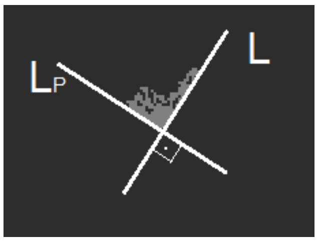


Fig. 6.2. The two lines used to model oriented objects: L will be fitted to the dominant visible side (the longest visible side of the obstacle), while L_P will model the other visible side, if any

Each step of the approach was designed having in mind the need for low computational complexity, and most of the computation is done in the discrete grid representation where the obstacles were initially detected. The main steps of the approach for computing the orientation of each obstacle are:

- *Computing the visible boundaries*: the idea of visibility refers to those obstacle cells that are directly observable from the ego vehicle (for example, in Fig. 6.1, for vehicles 1, 2, and 3, only sides perceived directly by the perception system are representative for the rectangular shape of these obstacles). Only these cells will be used for fitting the two line model.
- *Fitting the two-line model to the boundaries*: a random sample consensus approach will be applied, initially for the first line, and then for the second line on the remaining visible cells.
- *Validation/selection of the best orientation*: based on the strength (consensus score from RANSAC) of the two lines and additional criteria, the best orientation will be selected.

6.1.3. Detailed Method Steps

6.1.3.1. Computing the visible boundaries

First, the boundary cells of the obstacle are selected by labelling all obstacle cell that are adjacent to at least one grid cell that is not obstacle.

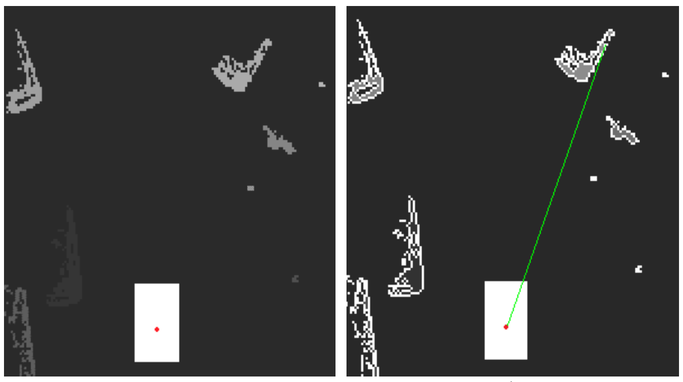


Fig. 6.3. a. Scenario from Fig. 6.1 with different labels for each obstacle, the ego vehicle is the white rectangle and the red dot represents the reference point for the perception system, b. Each obstacle patch has its boundary cells marked (white points), and an example of line of sight from the perception system to a boundary cell is shown as the green line

Next, the visible cells are selected. The condition for visibility is checked with ray tracing. For each boundary cell, the line of sight is computed from the ego vehicle. The center of the car is considered the origin of the perception system (view point), as it is equally distanced in the horizontal plane from the four lidars that perceive the obstacle (depending on the obstacle position, it is likely that only some of the four lidars provide 3D measurements). A more accurate (but more costly) verification would be to consider each lidar as view point.

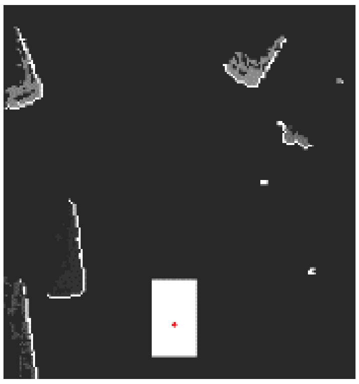


Fig. 6.4. Each obstacle patch has its visible boundary cells marked (white cells).

Normally, a boundary cell is visible if its line of sight does not intersect any other (current) obstacle cell. Given the discrete representation of the obstacle locations (grid cells), the line of sight for each boundary cell is approximated in the grid space using a Bresenham interpolation (between the perception system origin and the boundary cell). For obstacle sides that have a relative angle with the line of sight lower than 45 degrees, the discrete line of sight of some boundaries cells will likely intersect other boundary cells (such an example is shown in Fig. 6.3.b, the green line). Therefore, a boundary cell is considered visible (results in Fig. 6.4) if no obstacle cells are present on its line of sight, or if its line of sight intersects only some boundary cells from the obstacle. In this way, most of the visible sides of each obstacle are selected.

6.1.3.2. Fitting the two-line model to the boundaries

The first line, L, is represented with the general line model with 3 free parameters, as this allows representation of all possible orientations:

$$ax + bz + c = 0 \tag{1}$$

Given two points (x_1, z_1) and (x_2, z_2), the line parameters are computed as follows:

$$(z_1 - z_2)x + (x_2 - x_1)z + (x_1 z_2 - x_2 z_1) = 0 \tag{2}$$

The second line, L_p, given the perpendicularity constraint with L, will have the following equation:

$$bx - az + c_P = 0 \tag{3}$$

Once the main line L is computed, the direction of L_p will be perpendicular and only the free parameter c_P has to be estimated. If a point (x_3, z_3) belongs to L_p then:

$$c_P = az_3 - bx_3 \tag{4}$$

Computing the two lines for each obstacle, on the set of visible boundary cells, is described next.

The first line is computed in a RANSAC standard fashion: *k* samples of two random cells are selected from the set of visible boundary cells. The number of samples k=log(1-p)/log(1-w^S)=52 was computed by considering a success probability p=0.9999 and the ratio of good data of w=0.4, sample size s=2. The percentage of good data is a conservative value: assuming a vehicle with two of its sides visible, the side corresponding to the length will contain more than half of the visible boundary cells.

For each sample, the parameters of L are estimated as in equation (2). The consensus set is computed for each sample: the visible boundary cells that verify the line are counted (within a maximum distance threshold T_d=0.75 cells, this allows a band of 1.5 cells width around the line position). The line having the largest consensus score (number of inliers) is considered as solution for L if its number of inliers is at least the number of expected good cells (*w* times the number of visible boundary cells). If selected as solution, the parameters of L are recomputed with least squares fitting on the consensus set.

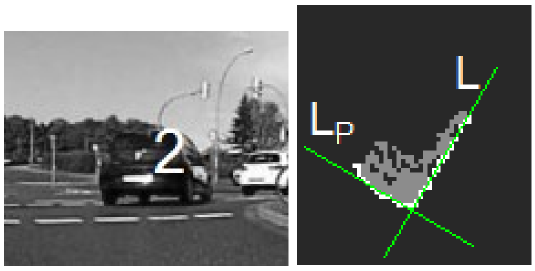


Fig. 6.5. The two lines computed for the vehicle 2 are shown (the scenario presented in Fig. 6.1).

The outliers, those cells that do not belong to the dominant line L, are selected for computing the second line L_P. In this case, one point is enough to compute the free term c_P. The approach is also based on selecting random samples: each sample consists of one cell, and the line with the largest number of inliers is selected. In this case, a larger ratio of good data is assumed (cells from the first line are not taken into account) w=0.6. As a consequence, the number of random samples is reduced to k=10 (beside a greater *w*, the sample size is *s*=1). The results for L and L_p for one of the vehicles from Fig. 6.1 are shown in Fig. 6.5.

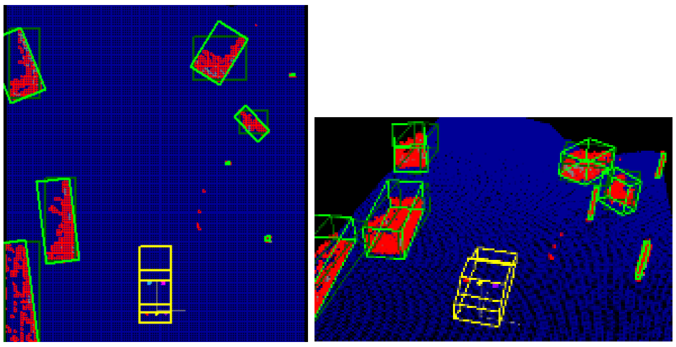


Fig. 6.6. The oriented cuboids for the scenario presented in Fig. 6.1 are show with bright green, in top-view (left) and a perspective view of the scene (on the right)

Based on the dominant line, the oriented cuboid is computed for each obstacle. This is done by projecting the obstacle cells onto the two lines and finding the extreme projections on each line as the limits for the oriented cuboid. The scenario presented in Fig. 6.1 is a typical example where the proposed approach is efficient, with the oriented cuboids shown in Fig. 6.6. However, there are many situations where additional validation is required for the orientation, and this issue will be discussed next.

6.1.3.3. Validation/selection of the best orientation

First of all, the orientation is not validated for obstacles that have a small top view print, such as poles or pedestrians (normal standing pose). This is done by imposing a small absolute threshold (=8 cells) on the number of inliers of the first line L.

When dealing with obstacles that have two sides visible to the perception system, both lines are computed and the orientation is computed reliably (details in the Results section). In real scenarios, obstacles have various positions and orientations, and often only one obstacle side is visible. This can apply not only for rectangular obstacles such as vehicles, but also for structures that exhibit only one visible side for the perception system: building, fences (see next figure) etc.

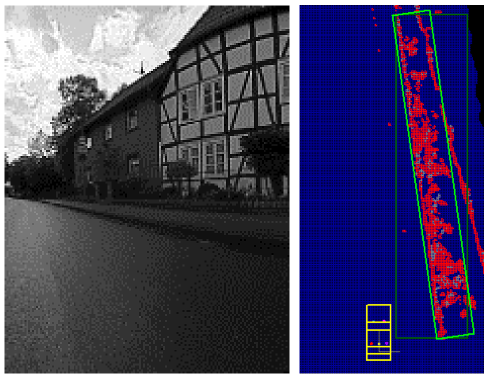

a. b.

Fig. 6.7. a. Scene with a large fence + building on the right side of the road, b. Top-view of the oriented (bright green) and un-oriented obstacle (dark green) detected from the 3D data of the fence and nearby vegetation

Two different situations must be distinguished. The first one is when the dominant line L has a strong support and the orientation can be considered valid (example in Fig. 6.7), regardless of the second line. The second situation is when L is computed, but its score is not sufficient to rely only upon one side. These two situations are solved with hysteresis (double thresholding) and reasoning based on the strength of the second line L_p.

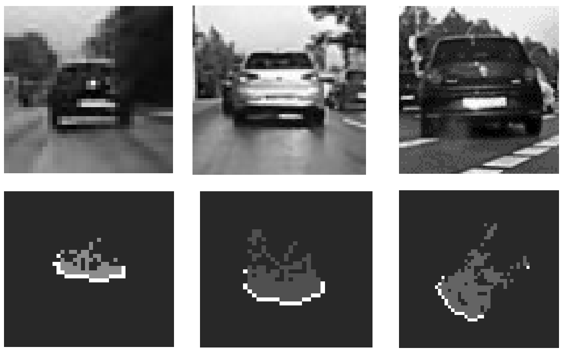


Fig. 6.8. Small passenger cars that are viewed from the rear (or front), without perceiving one of the lateral sides, often present curved visible boundaries that are less reliable to fit lines

For the first situation, a strong threshold T_S is imposed upon the consensus score of L. If the consensus score is above T_S then the orientation is validated. The value of T_S was established by considering the width of common vehicles (passenger cars). If such a vehicle is seen only from the rear or from the front, then its visible side has a width of 1.6 to 2+ meters. If the dominant line locks onto the rear or the front of a small vehicle, then in order to have a reliable fit, T_S is set to 15 cells (recall that a grid cell has a size of 10x10 cm). This threshold is set rather high with the following justification: if the dominant line fits the rear or the front of a small passenger car, then a good support is required for the line because this vehicle's side usually presents a curved profile (more or less depending on the model and shape of the bumpers, see Fig. 6.8).

If the consensus score of L is below T_S then the second line is considered. It might be either a vehicle with partially obstructed visible sides, or a smaller obstacle that can be represented with oriented cuboids. If line L_P has a consensus score of at least T_{SP} (= 10 cells), then the orientation is validated.

If both L and L_P have consensus scores below T_S, and T_{SP} respectively, then there are several ways to deal with the situation:

- Validate the orientation but assume it has a lower accuracy (it can be estimated from the consensus score of L), and rely on higher level processing (tracking) to get a better orientation through temporal filtering. From our experiments, for situations similar to Fig. 6.8, the orientation error is typically around several degrees (with extreme values of 10-12 degrees rarely observed), and tracking can smooth out these deviations.
- Use the context for reasoning: if only L was computed and with a poor score, then it's likely that the obstacle is (1) a vehicle without any lateral sides visible or (2) another obstacle that has only one significant dimension (example: a bicyclist, obstacle 4 in Fig. 6.1).

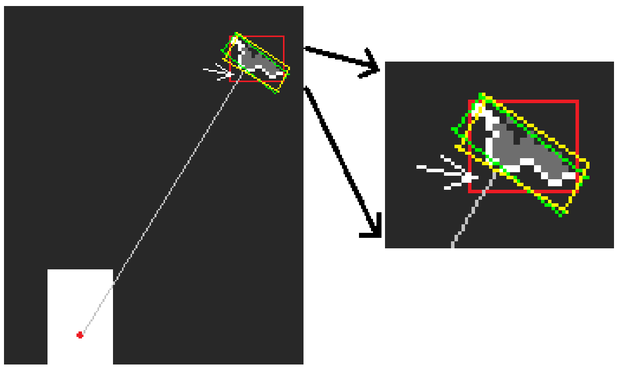


Fig. 6.9. When the dominant line L is less reliable, the line of sight can be used to infer the obstacle orientation, as the yellow cuboid. The un-oriented cuboid is shown with red, and the cuboid given by L with green. The line of sight to the obstacle center is shown. The white arrow depicts the free area, visible to the perception system, which is computed for each cuboid

For the second way, the position of the obstacle relative to the ego can be used to infer its orientation: if only one side is visible for a vehicle, then its orientation is likely along the line of sight with a certain freedom. However, if the obstacle is not a vehicle, but a thinner obstacle, then this "line of sight" orientation might be less accurate than the one given by L. To solve this issue, three possible orientations are considered (Fig. 6.9): the one given by L, the orientation given by the line of sight of the center of the obstacle, and the default null orientation (un-oriented). For each cuboid hypothesis, the free area of the cuboid that is visible to the perception system is computed and used as a measure of how well the cuboid envelopes the obstacle. The cuboid with the smallest unoccupied area is selected.

6.1.4. Results

The method for computing the orientation was implemented in C/C++ and integrated in the detection framework [48]. The approach is very fast: the running time for one obstacle (average vehicle sized) is 0.15-0.2 milliseconds. The processing speed is several times higher compared with existing methods. The processor used for testing was a Core I5, 2.5 Ghz. As comparison, for one obstacle, the processing time in [6] is about 1.5 ms on a similar processor (Core I7, C++ implementation).

The precision and accuracy of the orientation were measured on several scenarios were the real orientation of the vehicles in the scene could be estimated with a reasonable error.

The first scenario (Fig. 6.10) involved a sequence of several tens of frames where the ego vehicle was standing still, yielding the right of way to incoming traffic. The ego vehicle was positioned at an angle relative to the road axis, as it was leaving the main road. The angle between the ego vehicle reference frame and the road longitudinal axis was estimated manually to 14.4 degrees. The fixed road infrastructure (curbs) that is visible in the 3D lidar data was used to evaluate this angle.

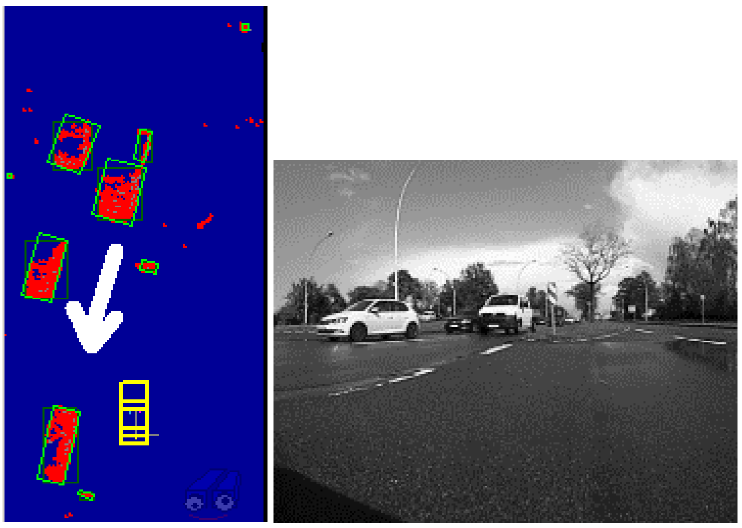


Fig. 6.10. Scenario 1 used for evaluation. Incoming traffic (the direction of motion shown with the white arrow) at an angle of 14.4 degrees

A total of 226 vehicle orientations, from 9 distinct vehicles, were kept for evaluation (some measurements were removed – vehicles that changed the lane, or small obstacles). The distance to the vehicles was up to 24 meters from the ego. The angle for each orientation was computed relative to the Z axis of the ego vehicle. The mean value obtained was 14.53 degrees, very close to the manual estimation of 14.4 degrees between the road axis and the ego vehicle. The standard deviation was 1.4 degrees, proving a robust evaluation of the orientation. The histogram of the angle values is shown in Fig. 6.11.

The second scenario used for evaluation was a sequence where the ego was driving straight on a multi-lane straight road, overpassed slower vehicles, and incoming vehicles were also present. A total of 375 orientations were computed for the vehicles (some up to 45 meters) that were moving straight (this was established from the video data). The mean angle value was 0.4 degrees (versus the expected value of 0 degrees) and the standard deviation 2.5 degrees.

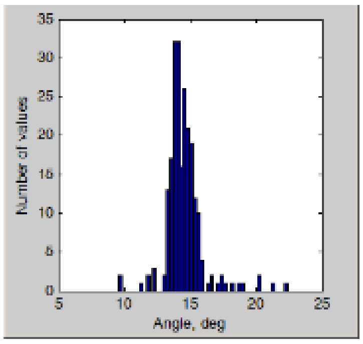
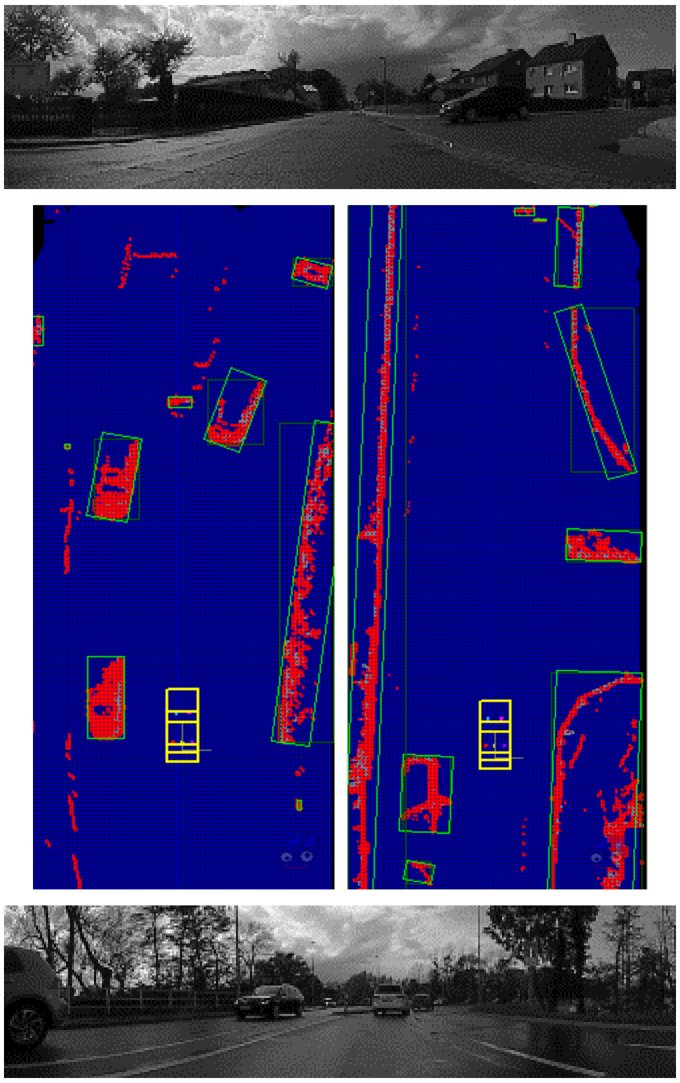


Fig. 6.11. Histogram of the angle values obtained for Scenario 1, mean value of 14.53 and standard deviation of 1.4 degrees

Evaluation was also performed on vehicles that were visible strictly from the rear or front, here the standard deviation was larger (4-6 degrees), when using the dominant line L. Tracking can be employed to increase accuracy. Other results are shown in . 6.12. Limitations of the cuboid model can be seen in one of the scenes: some obstacles are less suitable for a cuboid representation (curved fences).



. 6.12. Other results, including obstacles with small curvatures (fences). Top image shows the scene from the right top-view, bottom image shows the scene from the left top-view. The ego is drawn with yellow

A low complexity method for estimating the orientation of 3D obstacles, detected with multiple lidars, was developed. The method is very fast and it has high precision and accuracy, as proven by the quantitative evaluation performed.

7. Scientific, professional and academic development plans

In this chapter, future measures intended by the candidate to improve his scientific, professional and academic carrier will be discussed. These measures will be presented separately, oriented towards research and teaching (even though, in the academic environment, research and teaching often overlap naturally).

Since 2011, through his research work performed at IPPRRC (TUCN research center), the candidate has acquired extensive experience for environment perception with stereovision and lidar sensors. Mainly, starting from 3D point clouds, environment perception models and methods were developed by the candidate. This is an active research topic, with applications in driving assistance, robotics, etc., where the state of the art is continuously evolving and new issues require solutions. One of the most promising future improvements is the use of semantic information extracted from images to enhance the 3D point cloud. Within our research center, IPPRRC, there is already strong expertise with this topic. If images are not available, semantic segmentation is still possible, directly on the 3D cloud.

The candidate intends to pursue research in several directions, with the main objective of improving the state of the art for 3D environment perception. More robust and accurate segmentation methods will be developed when semantic information is available, with potential improvements for road/obstacle separation, obstacle grouping, obstacle classification etc.

- One important direction is developing innovative models and algorithms for perception from 3D point clouds, using traditional, probabilistic approaches. Multiple sensors of various modalities can be used for acquiring 3D data. This approach will handle only the 3D point cloud, in order to provide a solution independent of other sensors (cameras)
- A second direction, is developing innovative models and algorithms for perception from semantically enhanced 3D point clouds. Semantic information is obtained from images and it is registered with the 3D point cloud by assigning semantic information to the 3D points. Traditional, probabilistic methods will be developed
- Finally, a third promising direction will be the investigation and development of deep learning approaches (or other machine learning paradigms) for 3D perception. Solutions will be developed for both of the approaches mentioned above: when perception is carried out directly from the 3D point clouds or when it is done from semantically enhanced 3D point clouds.

One important aspect that will not be overlooked by the candidate is the continuous need to attract funds for future research, from industry and grant calls. This will provide better opportunities for young researchers to pursue a PhD and post-PhD carrier, and join the IPPRRC research center and the Computer Science Department. Implicitly, by maintaining a high level of expertise and competence, the contribution to the state of the art and the international visibility will increase.

Specifically, the candidate aims for the following measures in order to improve his research carrier and contribute to the long term research activity of the IPPRRC research center:

1. *Fundamental research*. From previous research work, the candidate has acquired extensive experience on topics as road and obstacle detection from 3D data, acquired with stereovision or lidar sensors. The solutions proposed so far can be extended furthermore by generalizing the models and algorithms proposed for environment perception (road surface, obstacles, curbs etc.). Solutions will be developed to exploit 3D point clouds enhanced with semantic information. In addition, potential approaches based on deep learning, for environment perception from semantic 3D point clouds, will be investigated.

2. *Increase the involvement in applied research.* In particular, the research topic based on mobile devices presented in this thesis is very promising. Smartphones with high processing power and high quality cameras are ideal processing platforms for computer vision applications. 3D reconstruction is possible even with a single camera (structure from motion), and this can be combined with powerful image sematic segmentation to have an advanced perception of the environment. Future funding for developing such a system will be searched in the next national grant calls. In addition, taking the solutions developed so far (stereovision and lidar based) even closer to a deployable system is a challenging task, as fundamental improvements for the accuracy will be required while keeping the computational complexity low.

3. *Involvement in existing and future research projects, as well as participation to future grant competitions.* The candidate is currently involved in two research projects (UP-DRIVE and SEPCA, see Table 1.3) where he's actively searching and developing solutions to key research issues. Original solutions for road and obstacle detection from multi lidar data were already developed in the UP-DRIVE project by the candidate. After future testing and improvements, these results will be submitted for publication in an international journal and/or conference(s).

4. *Continuous (and more frequent) dissemination of research results* in international journals and conferences, with the goal of increasing furthermore the international visibility of the candidate's work and the IPPRRC research center.

5. *Reviewer* (continued activity) for international journals such as IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Vehicular Technology, Sensors (MDPI) etc., and international conferences. This will guarantee permanent contact with the state of the art for the research topics approached by the candidate.

As a fulltime member of the Computer Science Department, the candidate will continue his teaching activities. An important activity will be advising bachelor and master students in their license and dissertation research activities. Special attention must be given to these students and their research topics, because those who are capable and research oriented might be the next PhD students. Should this habilitation be successfully defended, the candidate will also supervise PhD students. This will imply finding relevant research topics, with potential for a PhD thesis, permanent interaction and guidance for ensuring successful research results, obtaining funding for research activities, etc. More specifically, the candidate will continuously improve his teaching activities and the education provided for students by the following measures (some also involve the research part):

1. Involving students (bachelor/master and PhD students) in research projects. In order to provide such an opportunity for students, challenging but appropriate (as complexity) problems will be identified within the research projects the candidate will be involved in, or in related research topics. The same approach, but with a lower magnitude, will be attempted for the semester projects at the Image Processing and Pattern Recognition Systems courses. This will create a positive exposure for students to the latest developments in the domain of the attended courses.

2. The candidate will allocate more effort for searching and receiving funding for research, from various sources: industry or grants (national or European calls). In addition to the project guided research work, the additional funding from research will help future PhD students to focus fulltime on their thesis. This also allows recruiting the best graduates for advanced studies and increases their chances of following an academic carrier within the Computer Science Department. The academic human resource is one of the most important factors for a quality and successful teaching process.

3. Keeping the content of the Computer Architecture course up to date, while maintaining a gradual transition from basic to advanced specific topics. The main goal is to use basic and intermediate hardware notions for explaining modern processor architectures by carefully choosing the appropriate level of abstraction. Such a measure will maintain an increased interest level from students towards this course. Including additional explanatory figures and comments in the lecture slides will also help.

4. Permanently consult relevant publications (books and scientific articles) and online courses from relevant universities within the field of his taught courses. This will allow high quality content and explanations to be delivered to students during lectures and practical works.

5. In order to properly deliver information towards the students, the candidate will try to improve his communication and teaching skills. One important aspect is to continuously encourage students to ask questions during lectures, and to have a dialog with them about the topics discussed.

6. Using the knowledge acquired from research projects to propose new practical laboratory works. This will ensure high quality lab materials that are of interest for students. Given the nowadays accessibility to deep learning tools and networks, laboratory works related to deep learning can be added for the Image Processing and Pattern Recognition Systems courses.

7. Teaching or proposing new courses, related to the topics of this habilitation thesis, should such a task will be received from the Computer Science Department.

References

[1] B. Wallace, R. Scott and A. Spaans, "The DRDC Ottawa Space Surveillance Observatory," in *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, Hawaii, September 12-15, 2007.

[2] M. P. Levesque, "Automatic Reacquisition of Satellite Positions by Detecting Their Expected Streaks In Astronomical Images," in *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, Hawaii, September 1-4, 2009.

[3] M. Levesque, "Evaluation of the iterative method for image background removal in astronomical images," DRDC Valcartier TN 2007-344. Defence R&D Canada – Valcartier. http://pubs.drdc.gc.ca/PDFS/unc69/p52905 4.pdf.

[4] M. Levesque, "Image processing technique for automatic detection of satellite streaks," DRDC Valcartier 2005 TR-386. Defence R&D Canada – Valcartier. http://cradpdf.drdc.gc.ca/PDFS/unc64/p527352.pdf.

[5] H. Hirschmüller, P. R. Innocent and J. M. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *International Journal of Computer Vision,* vol. 47(1/2/3), pp. 229-246, April-June 2002.

[6] S. Nedevschi, et al., "High Accuracy Stereo Vision System for Far Distance Obstacle Detection," in *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 14-17, 2004.

[7] I. Haller, C. Pantilie, F. Oniga and S. Nedevschi, "Real-Time Semi-Global Dense Stereo Solution with Improved Sub-Pixel Accuracy," in *Proceedings of 2010 IEEE Intelligent Vehicles Symposium*, San Diego, CA, USA, June 21-24, 2010.

[8] H. Hirschmüller, "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR'05*, June 2005.

[9] L. Stefano, M. Marchionni and S. Mattoccia, "A fast area-based stereo matching algorithm," *Image and Vision Computing,* vol. 22, pp. 983-1005, 2004.

[10] C. Arth and D. Schmalstieg, "Challenges of Large-Scale Augmented Reality on Smartphones," in *ISMAR 2011 Workshop: Enabling Large-Scale Outdoor Mixed Reality and Augmented Reality*, Basel, Switzerland, 2011.

[11] F. Langguth and M. Goesele, "Guided Capturing of Multi-view Stereo Datasets," in *Eurographics*, 2013.

[12] J. Won, M. Lee and I. Park, "Active 3D Shape Acquisition Using Smartphones," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012.

[13] Q. Pan et al., "Rapid Scene Reconstruction on Mobile Phones from Panoramic Images," in *Proceedings of 2011 IEEE ISMAR*, 2011.

[14] A. Sankar and S. Seitz, "Capturing indoor scenes with smartphones," in *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST '12)*, 2012.

[15] C. Vancea and S. Nedevschi, "Analysis on Different Image Rectification Approaches for Binocular Stereovision Systems," in *Proceedings of 2006 IEEE ICCP*, Cluj-Napoca, Romania, 2006.

[16] E. Trucco and A. Verri, Introductory Techniques to 3D Computer Vision, Prentice Hall, 1998.

[17] J. Woodfill et al., "Data Processing System and Method". Patent U.S. Patent 6,215,898 B1, April 10, 2001.

[18] Federal Highway Administration, "Red-Light Running Report," U.S. Department of Transportation, http://safety.fhwa.dot.gov/intersection/redlight/, 2008.

[19] Y. Shen, U. Ozguner, K. Redmill and J. Liu, "A robust video based traffic light detection algorithm for intelligent vehicles," in *Intelligent Vehicles Symposium 2009 IEEE*, 3-5 June 2009.

[20] C. Cheng-Chin, H. Ming-Che, L. Hong-Sheng, A. Pratama and S. Wei-Cheng, "Detecting and recognizing traffic lights by genetic approximate ellipse detection and spatial texture layouts," *Int. Journal of Innovative Computing, Information and Control,* vol. 7, pp. 6919-6934, 2011.

[21] C. Wang, T. Jin, M. Yang and B. Wang, "Robust and real-time traffic lights recognition in complex urban environments," *International Journal of Computational Intelligence Systems,* vol. 4, pp. 1383-1390, December 2011.

[22] R. D. Charette and F. Nashashibi, "Traffic light recognition using image processing compared to learning processes," *Intelligent Robots and Systems,* pp. 333-338, 10-15 Oct 2009.

[23] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao and H. Chen, "The recognition and Tracking of Traffic Lights Based on Color Segmentation and CAMSHIFT for Intelligent Vehicles," in *Intelligent Vehicles Symposium 2010 IEEE*, 21-24 June 2010.

[24] M. Diaz-Cabrera, P. Cerri and J. Sanchez-Medina, "Suspended traffic lights detection and distance estimation using color features," in *In Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012.

[25] J. Roters, X. Jiang and K. Rothaus, "Recognition of traffic lights in live video streams on mobile devices," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 10, p. 1497–1511, 2011.

[26] H. D. Cheng, X. H. Jiang, Y. Sun and J. L. Wang, "Color image segmentation: Advances and Prospects," *Pattern Recognition,* pp. 2259-2281, 2001.

[27] R. Labayrade, D. Aubert and J. P. Tarel, "Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation," in *IEEE Intelligent Vehicle Symposium*, 2002.

[28] P. Lombardi, M. Zanin and S. Messelodi, "Unified stereovision for ground, road, and obstacle detection," in *Proc. IEEE Intelligent Vehicles Symposium*, 2005.

[29] R. Labayrade and D. Aubert, "A single framework for vehicle roll, pitch, yaw estimation and obstacles detection by stereovision," in *IEEE Intelligent Vehicles Symposium*, Columbus, USA, June 2003.

[30] R. Labayrade and D. Aubert, "In-vehicle obstacles detection and characterization by stereovision," in *Proceedings the 1st International Workshop on In-Vehicle Cognitive Computer Vision Systems*, 2003.

[31] X. Ai, Y. Gao, J. Rarity and N. Dahnoun, "Obstacle Detection Using U-Disparity on Quadratic Road Surfaces," in *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems*, The Hague, The Netherlands, October 6-9, 2013.

[32] S. Kubota, T. Nakano and Y. Okamoto, "A global optimization algorithm for real-time on-board stereo obstacle detection systems," in *Proc. IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, 13-15 June 2007.

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

[34] B. Hummel, S. Kammel, T. Dang, C. Duchow and C. Stiller, "Vision based Path Planning in Unstructured Environments," in *Proceedings of IEEE Intelligent Vehicles Symposium (IV 2006)*, Tokyo, Japan, June 13-15, 2006.

[35] Q. Yu, H. Araujo and H. Wang, "Stereovision Based Real time Obstacle Detection for Urban Environments," in *Proceedings of ICAR 2003, The 11th International Conference on Advanced Robotics*, Coimbra, Portugal, June 30 - July 3, 2003.

[36] Q. Yu, H. Araujo and H. Wang, "A Stereovision Method for Obstacle Detection and Tracking in Non-Flat Urban Environments," *Autonomous Robots,* p. 141–157, 2005.

[37] Z. Hu and K. Uchimura, "U-V-Disparity: An efficient algorithm for stereovision based scene analysis," in *IEEE Intelligent Vehicles Symposium*, Las Vegas, USA, June 2005.

[38] M. Wu, S.-K. Lam and T. Srikanthan, "Nonparametric Technique Based High-Speed Road Surface Detection," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, p. 874 – 884, April 2015.

[39] J. Woodfill, G. Gordon, D. Jurasek, T. Brown and R. Buck, "The Tyzx DeepSea G2 Vision System, A Taskable, Embedded Stereo Camera," in *Conference on Computer Vision and Pattern Recognition*, New York, June 2006.

[40] L. Matthies and P. Grandjean, "Stochastic performance modeling and evaluation of obstacle detectability with imaging range sensors," *IEEE Transactions on Robotics and Automation,* vol. 10, no. 6, pp. 783 - 792, Dec. 1994.

[41] M. Bertozzi and A. Broggi, "GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE Trans. Image Processing,* vol. 7, pp. 62-81, 1998.

[42] M. Cech, W. Niem, S. Abraham and C. Stiller, "Dynamic ego-pose estimation for driver assistance in urban environments," in *IEEE Intelligent Vehicles Symposium*, Parma, Italy, 2004.

[43] S. Kubota, T. Nakano and Y. Okamoto, "A global optimization algorithm for real-time on-board stereo obstacle detection systems," in *IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, 13-15 June 2007.

[44] F. Oniga and S. Nedevschi, "Processing Dense Stereo Data Using Elevation Maps: Road Surface, Traffic Isle, and Obstacle Detection," *IEEE Transactions on Vehicular Technology,* vol. 59, no. 3, p. 1172 – 1182, 2010.

[45] J. Siegemund, D. Pfeiffer, U. Franke and W. Förstner, "Curb reconstruction using conditional random fields," in *IEEE Intelligent Vehicles Symposium*, San Diego CA, June 2010.

[46] A. Wedel, H. Badino, C. Rabe, H. Loose, U. Franke and D. Cremers, "B-Spline Modeling of Road Surfaces With an Application to Free-Space Estimation," *IEEE Transactions on Intelligent Transportation Systems,* vol. 10, no. 4, pp. 572 - 583, Aug. 2009.

[47] H. Loose and U. Franke, "B-Spline-Based Road Model for 3D Lane Recognition," in *IEEE Conference on Intelligent Transportation Systems*, 2010.

[48] M. Vergauwen, M. Pollefeys and L. V. Gool, "A stereo-vision system for support of planetary surface exploration," *Machine Vision and Applications,* vol. 14, no. 1, p. 5–14, 2003.

[49] A. Broggi, C. Caraffi, P. P. Porta and P. Zani, "he Single Frame Stereo Vision System for Reliable Obstacle Detection used during the 2005 DARPA Grand Challenge on TerraMax," in *IEEE Intelligent Transportation Systems Conference*, Toronto, Canada, September 17-20, 2006.

[50] R. Turchetto and R. Manduchi, "Visual Curb Localization for Autonomous Navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA, October 2003.

[51] X. Lu and R. Manduchi, "Detection and localization of curbs and stairways using stereo vision," in *IEEE International Conference on Robotics and Automation (ICRA '05)*, Barcelona, Spain, April 2005.

[52] F. Oniga, S. Nedevschi and M.-M. Meinecke, "Curb Detection Based on a Multi-Frame Persistence Map for Urban Driving Scenarios," in *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems (ITSC'08)*, Beijing, China, 13-15 Oct. 2008.

[53] F. Oniga and S. Nedevschi, "Polynomial Curb Detection Based on Dense Stereovision for Driving Assistance," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, Madeira, Portugal, 2010.

[54] F. Oniga and S. Nedevschi, "Curb Detection for Driving Assistance Systems: A Cubic Spline-Based Approach," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, 5-9 June 2011.

[55] D. Kim, et al., "L-shape model switching-based precise motion tracking of moving vehicles using laser scanners," *IEEE Transactions on Intelligent Transportation Systems,* vol. 19, p. 598–612, 2018.

[56] X. Zhang, W. Xu, C. Dong and J. M. Dolan, "Efficient L-shape fitting for vehicle detection using laser scanners," in *Proc. IEEE Intell. Vehicles Symp. (IV 2017)*, 2017.

[57] M. Munz, K. Dietmayer and M. Mahlisch, "A sensor independent probabilistic fusion system for driver assistance systems," in *Intelligent Transportation Systems, ITSC'09, 12th International IEEE Conference on*, 2009.

[58] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut and R. Husson, "An Optimized Segmentation Method for a 2D Laser-Scanner Applied to Mobile Robot Navigation," in *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, 1997.

[59] R. MacLachlan and C. Mertz, "Tracking of moving objects from a moving vehicle using a scanning laser rangefinder," in *IEEE Intelligent Transportation Systems Conference*, Canada, 2006.

[60] X. Shen, S. Pendleton and M. H. Ang, "Efficient L-shape fitting of laser scanner data for vehicle pose estimation," in *IEEE Conference on Robotics, Automation and Mechatronics*, Cambodia, 2015.

[61] G. A. Borges and M.-J. Aldon, "Line extraction in 2d range images for mobile robotics," *Journal of Intelligent and Robotic Systems,* vol. 40, p. 267–297, 2004.

[62] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Graphics and Image Processing,* vol. 24(6), p. 381–395, 1981.

[63] R. Varga, A. Costea, H. Florea, I. Giosan and S. Nedevschi, "Supersensor for 360-degree environment perception: Point cloud segmentation using image features," in *IEEE Intelligent Transportation Systems Conference*, 2017.

[64] O. Hernandez, et al., "Performance instrumentation and compiler optimizations for MPI/OpenMP applications," in *OpenMP Shared Memory Parallel Programming*, Springer Berlin Heidelberg, 2008, pp. 267-278.