UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXVII Ciclo

3D Vision-based Perception and Modelling techniques for Intelligent Ground Vehicles

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Alberto Broggi

Dottorando: *Mario Sabbatelli*

Gennaio 2015

Alla mia famiglia,
alle amicizie oltre confine
ed al mio eterno caos, Roberta.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In recent years, the road safety research has made significant evolutionary leaps, especially in the last decade. The ultimate aim of the scientific research in the automotive is based on a fundamental idea: the autonomous driving. Ensure a high safety level to the passenger and to the surrounding environment beyond the human reflexes via advanced perception and control technologies represents the main goal of this research. In the this decade successfully experiments of autonomous driving on intelligent ground vehicles are presented and performed by different international car makers and research groups [1–4], as shown in Figure 1.1. Nowadays there are states in America (Nevada, Florida and California) that allow and legally regulate the autonomous vehicles in public roads.

The most important feature for an intelligent ground vehicle is represented by the perception system. A full and reliable reconstruction and modelling of the environment is required by any low and high intelligent trajectory planning systems.

Since 1985 the early perception experiments on mobile robots and vehicles have been performed using different sensor classes (e.g. LiDAR, radar and camera). The combination of different sensors in order to sense the environment surrounding the ego-vehicle is known as *sensor fusion*. For the most of famous car makers, the re-

search in this field has brought to reconsider their industrial design, optimizing the integration of perception technologies, such as camera, radar and LiDAR, in terms of efficiency and cost. Nowadays the research in stereoscopic vision system has made giant leaps and represents a really all-round solution for automotive and other fields also (e.g. mining, industrial safety, surveillance). The main aim of this system is to provide a full tridimensional reconstruction of the environment. Let us to assume to generate a reliable set of information of the world, the next challenge is how to model the collected data in order to provide an efficient representation of the environment. For each kind of intelligent ground vehicle, the lowest knowledge level for planning systems is represented by the set of traversable paths and the detection of moving and standing obstacles around the ego-vehicle.

(a) *BRAiVE*- VisLab, University of Parma

(b) *Bertha* - Daimler.

(c) *KITTI* - Karlsruhe Institute of Technology (KIT).

(d) *Google Car* - Stanford Artificial Intelligence Laboratory (SAIL).

Figure 1.1: Examples of fully autonomous ground vehicle.

1.2 Motivation

Many works have been presented in literature on perception and modelling for intelligent ground vehicle. Part of these techniques are successfully implemented and tested for ad hoc scenario. In this thesis will be analyzed and tested these techniques in order to provide a perception and modelling system of the surrounding environment for intelligent ground vehicles that it is independent of the application scenario.

1.3 Outlines of the thesis

In this manuscript will be analyzed the most important modelling techniques in terms of *obstacle* and *terrain* for intelligent ground vehicle based on input derived by a *stereo image processing*. The thesis is detailed by the following sections:

Chapter 2 - *Perception for Intelligent Ground Vehicles using Stereo Vision*: in the first part of this section a brief overview of the stereoscopic model will be detailed. Afterwards, the techniques to reconstruct the depth information by images will be described.

Chapter 3 - *Vision-based Environment Modelling*: in this chapter the developed approaches for the terrain and obstacle estimation will be described.

Chapter 4 - *Tests and results*: in this chapter will be detailed each test performed for each developed approaches. Next the obtained results will be analyzed and discussed in terms of performance and processing times.

Chapter 5 - *Conclusions and Future Works*: in this final chapter the conclusions and the derived contributions will be highlighted and future works as well.

Chapter 2

Perception for Intelligent Ground Vehicles using Stereo Vision

People experience a great delight in colour, generally.
The eye requires it as much as it requires light.

– Johann Wolfgang von Goethe

2.1 Stereo Vision

The goal of stereo vision is to recover the three-dimensional (3D) coordinates of real-world points seen through a binocular camera system. A binocular camera system consists of two cameras mounted on a baseline, see Figure 2.1. There are different techniques in literature to compute the depth information [5–8]. In this thesis a passive stereo approach is applied in order to obtain a 3D reconstruction that is based on finding corresponding point pairs. This process consist of two image points belonging to one real world point, one in the left camera and one in the right camera. Figure 2.1 the reconstruction process of the P_W point assuming to know the image position of the corresponding points of a world point in both images and the stereo

camera parameters.
The process of passive stereo will be described in more detail in the next subsections. The problem of finding corresponding point pairs for every point visible in both cameras, known as *disparity estimation*, is discussed in 2.1.5.

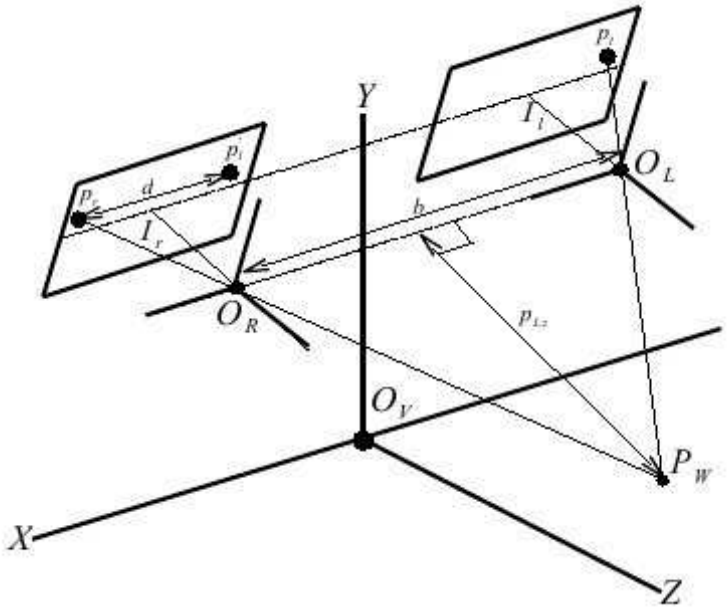


Figure 2.1: Stereo camera model

2.1.1 Passive Stereo

The Computer Vision field that deals with understanding and modelling of the geometry of multiple cameras is called multi-view geometry. In the last decades the presented methods in literature have reached a mature level. The theory behind these approaches are detailed in deep by Hartley and Zisserman in [9]. In the following sections a brief overview of theory and methods that apply to passive stereo will be described.

The process of passive stereo can be subdivided in 4 step procedure.

1. Calibration: estimation of the intrinsic and extrinsic parameters of the stereo

camera system.

2. Rectification: procedure based on the transformation and alignment of the images by a fix of the extrinsic parameters as shown in Figure 2.1.

3. Matching: as shown in Figure 2.1 the goal is to find the points p_l and p_r that correspond to the same physical point P_w for every point visible in both images.

4. Depth estimation: by horizontal image coordinates of the points p_l and p_r it is possible to the disparity d between them. The 3D coordinates of P_w relative to the origin O_v of the vehicle coordinate system are computed using the corresponding disparity value d and the rectified stereo camera parameters.

2.1.2 Camera model

The camera model is represented by the intrinsic and extrinsic parameters. The intrinsic parameters describe how a point is projected on the image plane I. The extrinsic parameters describe the position and the orientation of the camera's focal point O_c according to a world reference frame. Let us assume the orientation as rotation matrix $\mathbf{R}$ which describes the rotation between the world coordinate system and the camera coordinate system and the vector $\mathbf{T}$ as translation vector that describes the translation between the world coordinate system and the camera coordinate system. The relation between the world coordinates and the camera coordinates is detailed in Equation 2.1:

$$P_c = \mathbf{R} \cdot P_w + \mathbf{T} \tag{2.1}$$

where P_w represents a vector of coordinates $(P_{w_x}, P_{w_y}, P_{w_z})^T$ in the world coordinate system and P_c is the resulting vector of coordinates $(P_{c_x}, P_{c_y}, P_{c_z})^T$ in the camera coordinate system after the conversion from image coordinate system (2D) into the 3D coordinate system of camera. In order to process the P_c it is necessary to introduce the projection model to convert a three-dimensional point to a two dimensional (2D) image. The pinhole model, shown in Figure 2.2, is a linear, thus distortion free, model that captures the process of light falling through a lens onto an image plane. The lens is modelled as one single point O_c, known as pinhole. The perpendicular

line from O_c to the image plane I is known as optical axis. The segment from O_c to I represents the focal length f. The image coordinates p_{i_x} and p_{i_y} is computed from the camera coordinates of point P_c using the formula shown in Equation 2.2.

$$p_i = \begin{pmatrix} p_{i_x} \\ p_{i_x} \end{pmatrix} = \begin{pmatrix} f\frac{P_{c_x}}{P_{c_z}} & f\frac{P_{c_x}}{P_{c_z}} \end{pmatrix}^T \tag{2.2}$$

where the x and y coordinates of P_c relative to O_c can be computed from the image coordinates when P_{c_z} is known using the Equation 2.3.

$$P_c = \begin{pmatrix} P_{c_x} \\ P_{c_y} \\ P_{c_z} \end{pmatrix} = \begin{pmatrix} \frac{P_{c_x} p_{i_x}}{f} & \frac{P_{c_y} p_{i_y}}{f} & P_{c_z} \end{pmatrix}^T \tag{2.3}$$

The most of commercial cameras are equipped with a glass lens system in order to replicate the pinhole model that could add several types of distortions to the images. In [10, 11] the authors show different methods to remove the distortion caused by the lens. An easy estimation approach of the camera parameters is presented in by Heikkila et al. in [12].

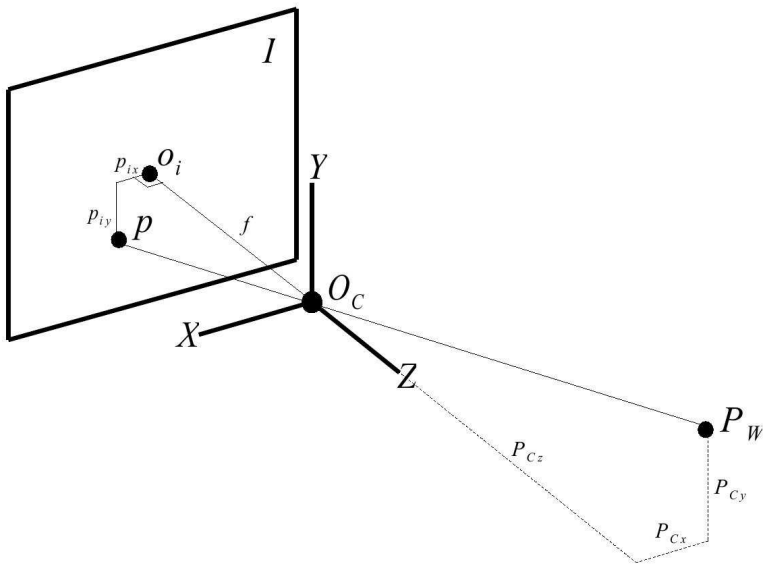


Figure 2.2: Pinhole camera model.

Camera calibration is usually performed using several markers printed on a board (e.g. chessboard, blobs) in a flat area. The size of the board and relative position of these markers is known beforehand. After taking several images of this calibration plane under various orientations and extracting the image coordinates of the markers, it is possible to estimate the intrinsic camera parameters. The next step is the definition of the transformation from the complex model to the pinhole model that can be applied to the images in order to represent the captured data by the sensor as a pinhole (linear distortion free) camera. This process is usually known as *undistortion*. In all other sections it will assume that the used cameras behave according to the pinhole model. Let us to analyze the complex model in order to represent the images using the pinhole model.

The applied camera model is based on a 3 step procedure:

1. estimation of normalized image coordinates p_n using the Equation 2.4;

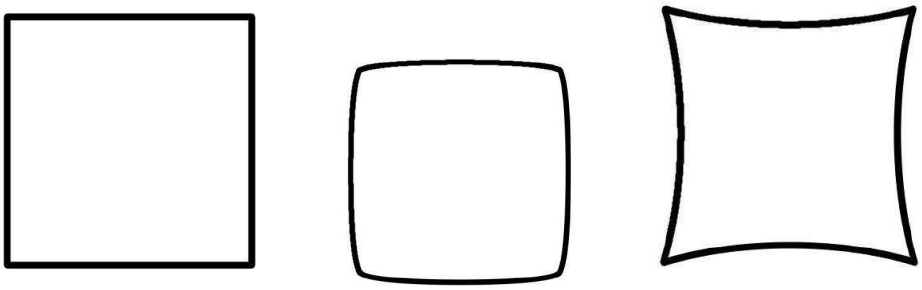
$$p_n = \begin{pmatrix} p_{n_x} \\ p_{n_y} \end{pmatrix} = \begin{pmatrix} f\frac{P_{C_x}}{P_{C_z}} & f\frac{P_{C_y}}{P_{C_z}} \end{pmatrix}, \quad f = 1 \tag{2.4}$$

2. definition of the radial distortion model in image corodinates p_d by the k_1 and k_2 parameters as shown in Equation 2.5 where r is the radius from the image centre point O_i; the most common radial distortion models are represented in Figure 2.3;

$$\begin{gathered} r^2 = x^2 + y^2 \\ p_d = \begin{pmatrix} p_{d_x} \\ p_{d_y} \end{pmatrix} = (1 + k_1 r^2 + k_2 r^4) p_n \end{gathered} \tag{2.5}$$

3. the estimation of un-normalized image coordinates p_d are computed using the Equation 2.6 where cc_x and cc_y represent the coordinates of the principal point O_i that it is the projection of the optical centre O_C onto the imaging plane I. The focal length is modelled by f_x and f_y.

$$p_d = \begin{pmatrix} f_x p_{d_x} + cc_x \\ f_y p_{d_y} + cc_y \end{pmatrix} \tag{2.6}$$



(a) (b) (c)

Figure 2.3: Examples of radial distortion: undistorted (a), barrel distortion (b), pin-cushion distortion (c)

2.1.3 Epipolar geometry

Epipolar geometry refers to the geometry between two cameras that it directly depends on the camera pinhole model (see Section 2.1.2) and adds information about the relative position and orientation between the two cameras. Figure 2.4 show a general setup of a stereoscopic camera system where the optical axis of both camera's are not parallel to each other. The rectification process cited in Section 2.1.1 lets to obtain a parallel alignment between the optical axis. Also the baseline connecting the two optical points will run parallel to the image lines in both imaging planes. This step is required in order to search easily the corresponding points between the captured images. Figure 2.5 shows the rectification process result.

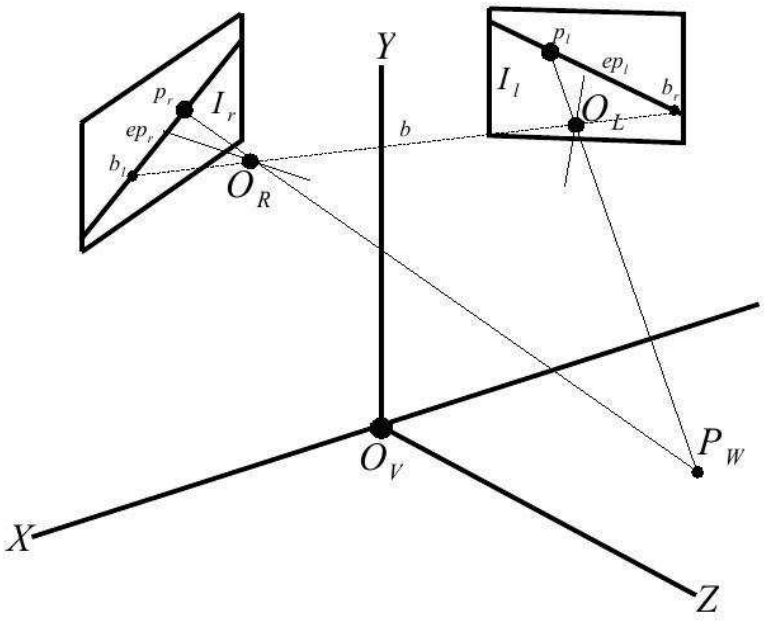


Figure 2.4: Unrectified stereo camera geometry based on pinhole model.

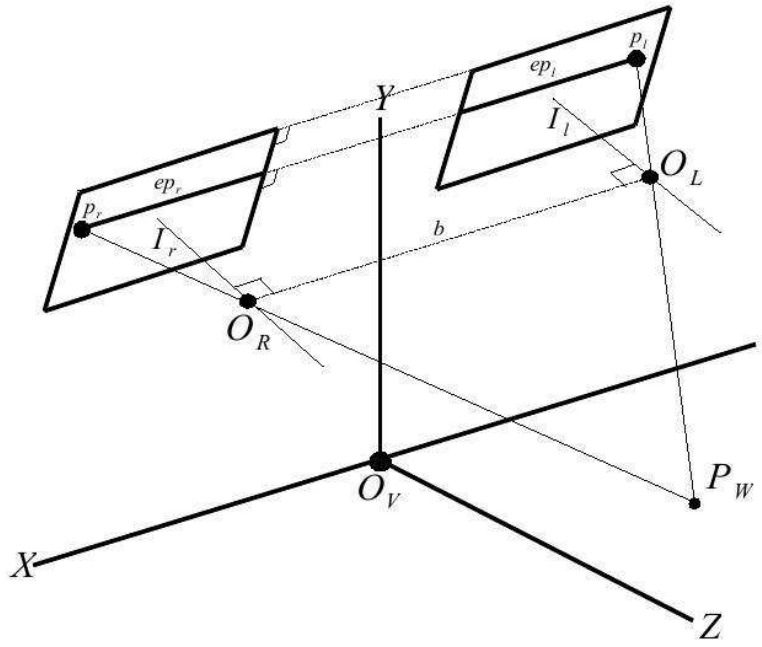


Figure 2.5: Rectified stereo camera geometry based on pinhole model.

This procedure let to estimate the intrinsic and extrinsic parameters of the stereo camera system. Using these parameters it is possible to construct a transformation from the real cameras alignment to a parallel cameras system. In this section it will explain in detail the epipolar rectification process. As shown in Figure 2.5 the epipoles b_l and b_r represent the points of intersection of the baseline b , the line joining the optical centres O_l, and O_r, with the image planes I_l and I_r. An epipole point is a simply projection of a image point in one carmera in regarding to the optical centre of the other. Let us to assume an epipolar plane as an imaginary plane defined by the real point in the world P_W and the optical centres O_l and O_r. As shown in Figure 2.5 the epipolar lines ep_l and ep_r represent the straight lines of intersection of the epipolar plane with the image planes I_l and I_r. All epipolar lines intersect at the epipole but for parallel stereo cameras the epipoles will be at infinity.

Using this calibration process for the stereo camera, it is possible to obtain the

intrinsic and extrinsic parameters of both cameras by a similar method cited in Section 2.1.2. The intrinsic parameters describe how points are projected onto the imaging planes of both cameras. It includes parameters such as the cameras focal length, principal point and the lens distortion. The extrinsic parameters describe the transformation from the left camera coordinate system to that of the right one. These parameters are represented by a rotation matrix and translation vector. With these information the epipoles can be estimate in both images by computing the intersection of the baseline with both imaging planes. Thus it is possible to compute an efficient search of corresponding points from a 2D search to a 1D search along the epipolar lines indeed By epipolar rectification of the images the epipolar lines will become parallel to the image lines and the search of corresponding point of p_l can be performed along the same horizontal scan-line in I_r, as shown in Figure 2.5.

Epipolar rectification first transforms the projection matrices of both cameras to a generic pinhole model (un-distortion) where the focal length will be the same for both cameras. The next step is to fix the orientation between the imaging planes of both cameras in order to obtain coplanar planes. Then the offset between the vertical positions of the focal points is removed. All that remains is a horizontal offset between the optical centres known as *baseline*. Finally, the transformations applied for epipolar rectification can be used to warp the images and compute the stereo reconstruction of a observed point in the world coordinate system.

2.1.4 Depth estimation

The geometry behind the stereo reconstruction of image points is explained in this section and is shown in Figure 2.6. Let us to assume the pinhole model and the images from the cameras are un-distorted and rectified as a parallel stereo camera system as cited in Section 2.1.3. In order to reconstruct the 3D coordinates of image points, a depth estimation of these points from the camera is required. This distance can be computed using Equation 2.3.

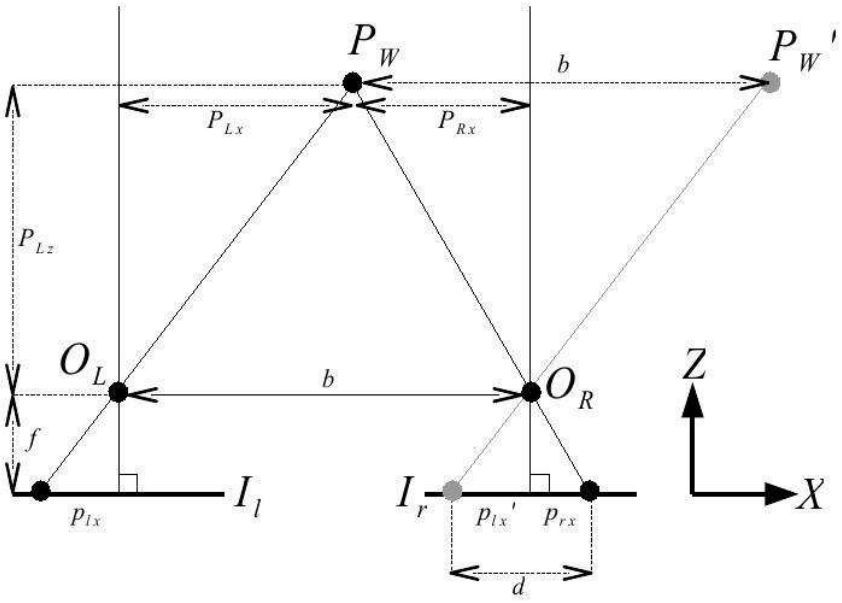


Figure 2.6: Stereo geometry

As shown in Figure 2.6 the distances P_{Lz} and P_{Rz} are equal and the difference between P_{Lx} and P_{Rx} corresponds to the baseline b, which leads to Equation 2.7.

$$\frac{P_{Lz}}{P_{Lx}-P_{Rx}} = \frac{f}{p_{lx}-p_{rx}} \rightarrow \frac{P_{Lz}}{b} = \frac{f}{d} \tag{2.7}$$

The depth of P_W relative to O_L can be computed with Equation 2.8.

$$P_{Lz} = \frac{fb}{d} \tag{2.8}$$

The final step is to transform the 3D coordinates from the left camera coordinate system to the vehicle coordinate system applying Equation 2.9 where $\mathbf{R_v}$ and $\mathbf{T_v}$ describe respectively the orientation and the translation between the left camera and the vehicle.

$$P_v = \mathbf{R_v} P_{Lc} + \mathbf{T_v} \tag{2.9}$$

2.1.5 Disparity estimation

In order to obtain a tridimensional reconstruction of the observed environment using a stereo camera system, as discussed in Section 2.1.1 finding correspondence points and the disparity between them is required. In this section it will be analyzed the field of stereo algorithms that is the techniques that enable disparity estimation from stereo image pairs taken by a stereo camera system, seen in Figure 2.5. A stereo image pair consists of one image from the left camera together with one image from the right camera assuming that both images are taken at the same time and are also un-distorted and rectified (see Section 2.1.3).

The aim of stereo algorithms is based upon the construction of a *disparity image* $d(x,y)$ that is to find for each pixel at image coordinates (x,y) of a camera, one corresponding point in the image from the other camera. After computed the corresponding points by the matching process from a rectified stereo image pair, the disparity value $d(x,y)$ can be obtained. Let us assume that $I_l(h,w)$ and $I_r(k,w)$ are correspondence points in a rectified stereo image pair, the disparity value $d(h,w)$ can be computed using Equation 2.10.

$$d(h,w) = h - k \tag{2.10}$$

The search for correspondence points is one of the key research topics in Computer Vision and is known as the correspondence problem or for more complex problems *feature matching*. For the most of common stereo camera, where the images are un-distorted and rectified, the search complexity is $O(H,W^2)$ because the epipolar lines are co-parallel. In order to optimize this process, the search of the corresponding points is restricted to a maximum bound max_d on the disparity: $O(H,Wmax_d)$.

The search space complexity is not the only issue in disparity estimation problem. Many hardware and environmental aspects can interfere with the matching process and therefore with the disparity estimation. Points on image patches with bad signal to noise ratio are usually prone to errors that it can be caused by the absence of texture. Repetitive textures on the other hand, pose difficulties due to the great similarity between sub-regions in the texture. This problem is also known as *sub-pixel accuracy*.

Also the difference in camera gain and bias, perspective distortion and occlusions make the search challenging. In [13, 14] the authors present a general overview of recent research into disparity estimation. In order to provide a reliable reconstruction and modelling of the terrain and the obstacles around the vehicle, several works in literature about the disparity estimation for intelligent vehicle have been analyzed [5, 7, 8, 15].

A method that provides a full reconstruction in terms of disparity by a stereo camera, is known as *dense stereo*. A different approach exists in literature where the number of locations is limited to certain image features for instance edges: *sparse stereo*. This technique was popular during the early experiments of Computer Vision when the computation power was limited. Nowadays a dense stereo approach has become feasible at real-time frame rates proving a perfect solution in several application fields (e.g. automotive, mining, agricultural, industrial safety). In the remainder of this thesis, dense stereo will be applied in Chapter 3 in order to reconstruct and model the terrain and the obstacles. For this thesis the perception model will refer to produce a dense disparity map $d(x,y)$ also known as *Disparity Space Image* (*DSI*).

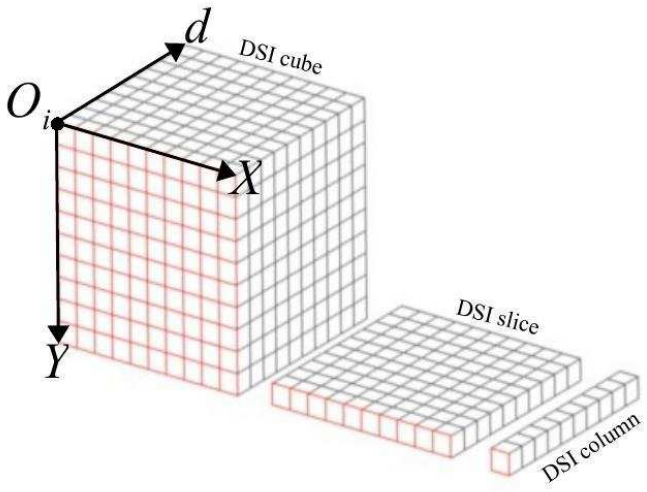


Figure 2.7: Disparity Space Image

A *DSI* (x,y,d), as shown in Figure 2.7, is referred for every position to a certain similarity measure. This value let to describe the confidence that the points $I_l(x,y)$ and $I_r(xd,y)$ relate to the same physical observed point in a world coordinate system. The disparity estimation process based on *DSI* basically can be describe in a 3-step procedure:

1. fill the *DSI* with the output of a single pixel similarity function S for every possible value of x, y and d using Equation 2.11

$$DSI(x,y,d) = S(x,y,d), \quad x \in [1 \dots H] \;\; y \in [1 \dots W] \;\; d \in [1 \dots max_d] \tag{2.11}$$

2. aggregate the output of similarity function as in Equation 2.12 where $K(x,y,d)$ gives a set of coordinates inside the *DSI* that can influence the similarity value of $DSI(x,y,d)$.

$$DSI(x,y,d) = \sum_{k \in K(x,y,d)} DSI(k_x,k_y,k_d), \quad x \in [1 \dots H] \;\; y \in [1 \dots W] \;\; d \in [1 \dots max_d] \tag{2.12}$$

   An example of similarity function can be represented by the following formula:

$$K(x,y,d) = [(x,y,d),(x-1,y,d),(x+1,y,d),(x,y-1,d),(x,y+1,d)]$$

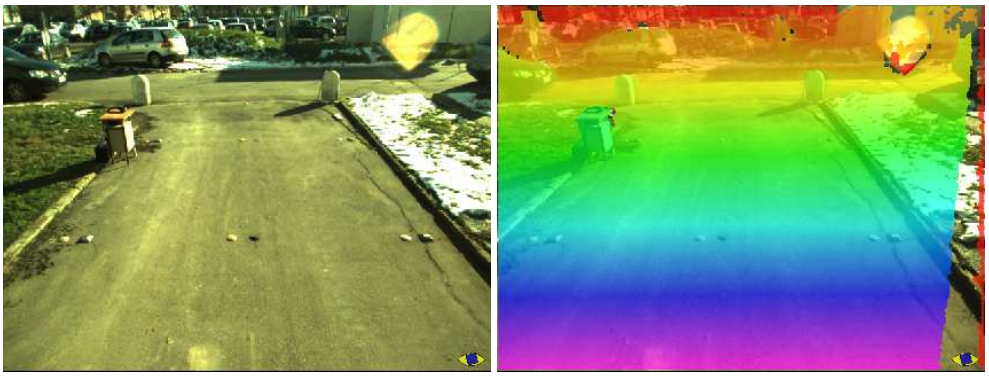
   where the effect is that neighbouring pixels influence each others similarity value and eventual will influence each others disparity estimate.

3. as optimization step, estimate for every point $I_l(x,y)$ the best value for d so that the overall disparity map matches the true disparity most likely.

A distinction between optimizations methods based on the portion of the DSI is required. Local methods optimize the disparity for a pixel solely based on its *DSI* column. Otherwise scan-line methods optimize the disparity for a pixel based on its *DSI* slice. Indeed global optimization methods optimize the disparity for a pixel based on the entire *DSI* cube.

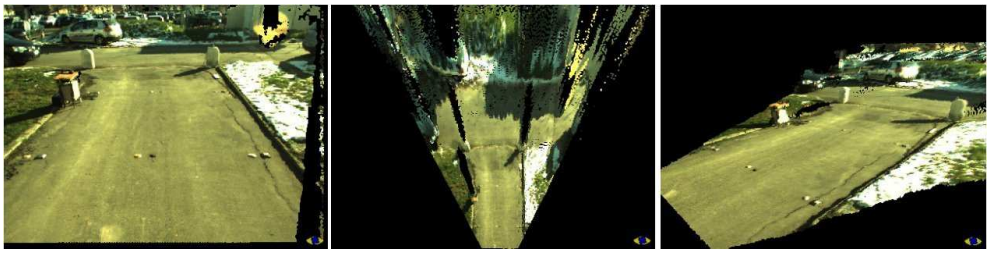
The 3D points engine used for this thesis, as shown in Figure 2.8, is based on the processing of a dense *Disparity Space Image* (*DSI*). To perform the stereo reconstruction in terms of computational costs for the matching and similarity algorithms, a modified *Semi Global Matching* (*SGM*) algorithm [5] with a multi-resolution analysis scheme [16] has been implemented. In order to reduce computational weight, a multi-threaded SIMD processing scheme has been devised, exploiting the parallel processing capabilities of the hardware platform.

The engine has been extensively tested during *VIAC*, the *VisLab Intercontinental Autonomous Challenge* [1, 16], in a variety of scenarios and in different conditions.



(a) Right input image.

(b) DSI.



(c) 3D virtual camera perspectives: front, top, lateral.

Figure 2.8: DSI and Dense Stereo 3D point cloud in several viewpoints.

Chapter 3

Vision-based Environment Modelling

Few people have the imagination for reality.

– Johann Wolfgang von Goethe

In this chapter a real-time tridimensional modelling approach for intelligent ground vehicles of 3D data, computed by stereo reconstruction (see Chapter 2), will be introduced.

In order to provide a medium level of knowledge of the surrounding environment, the proposed strategy has been divided in 2 sub-systems as follows.

1. Terrain estimator: the 3D input points are filtered in order to estimate a shape of the ground using a 2.5D map, as detailed in Section 3.1.

2. Obstacle detector: the discarded 3D points at the previous step are marked as obstacle candidates and processed in order to represent them by a full 3D clustering approach, detailed in Section 3.2. Also a linear Kalman filter is applied to discriminate moving and static obstacles.

The algorithm block diagram of the proposed strategy, shown in Figure 3.1, is

pipeline oriented, where each block can be implemented independently, as far as the data flow is respected. The block diagram of this traversability mapping algorithm: first a 3D point cloud is generated; then, a fully derivable surface model of the terrain is build, able to segment points into terrain inliers and outliers; finally, slope and occupancy information are fused together to build a traversability cost map.

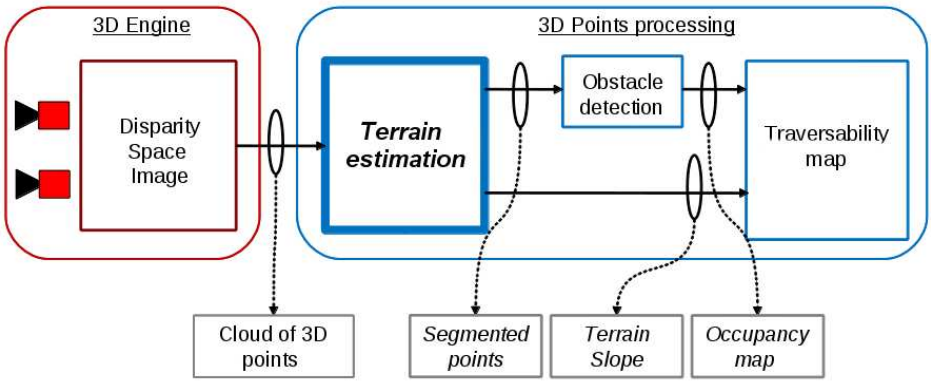


Figure 3.1: Pipeline of terrain estimation and obstacle detection system.

3.1 Terrain estimation

Reliable perception of terrain slope and terrain traversability is a key-feature for any ground vehicle designed for extreme and urban environments. This is often achieved processing 3D points clouds coming from high-quality dense depth maps, as seen in Section 2.1.5.

3.1.1 Relative works

The majority of the approaches project depth information into digital elevation maps [17] or into various types of Cartesian grid maps, containing cells of uniform size. Cells typically store information about the corresponding world portions, depending on the algorithm approach and type of sensor used. The so called 2D grid maps just con-

tains *occupancy* information (traversable, not traversable); these are useful for basic obstacle avoidance on flat terrains, based on simple range sensors. When cells store also height information, the subject refers to the 2.5D grid maps [18]. More complex grid maps embed full 3D information using adjacent stack of cells [19], or octrees connected cubes [20], and are able to represent objects at multiple heights located at the same range and azimuth. Elevation maps, as well as cubes grids, do not provide an immediate classification of their cells as belonging or not the terrain. Some authors [21] enrich cells' geometric characteristics with visual information, like color and texture, to understand if a given cell is part of the terrain, applying machine learning techniques on the basis of a priori knowledge. When it is possible to use assumptions on road model and on vehicle pose, successful solutions of geometry-based obstacle/terrain segmentation have been proposed, in some cases working in image coordinates by using v-disparity space [22]; in other cases applying RANSAC fitting to segment the 3D points in road inliers and outliers [23].

In [24] Broggi et al. presented a geometry based technique able to segment 3D data points into terrain inliers and outliers, without any constrains on vehicle pose and surface roughness. The segmentation is made on the basis of *traversable terrain* concept: *any surface where the vehicle can drive on*. In other words, *any surface not too rough and steep for the vehicle capabilities*. Ants Colony Optimizations (ACO) was used by the authors to fit points into a certain number of longitudinal and lateral 2D models, then fused together to obtain a 3D Cartesian model of the terrain. ACO works very satisfactorily, but it is computationally demanding (as well as RANSAC based techniques), since it requires many iterations for each model to reach optimal results. Moreover, subdividing the world in several 2D models, not constrained with each other but just merged at the final stage, leads to a noisy and, occasionally, unstable results.

3.1.2 Overview

In this work the candidate proposes a new technique to build a real-time map of the *traversable terrain*, fitting 3D points into a rational B-Spline surfaces [25] [26]. Oppositely to the ACO based approach, made of a mesh of several 2D models, B-Splines

provide a full 3D elevation map model, where each sample represents a constrain for the others, improving robustness; at the same time, B-Splines are locally controllable by control points, allowing the same ACO's sensitivity to localized terrain slopes. Finally, the proposed iterative re-weighted fitting method performs terrain's inliers and outliers segmentation with very few iteration, compared to ACO and RANSAC, ensuring computational efficiency in a wider range of scenarios.

Example of the final result is shown in Figure 3.2 where the terrain slopes with a synthetic grid and obstacles are represented with their original color and enriched with 3D info, while a picture of the scene is also shown to compare the results.

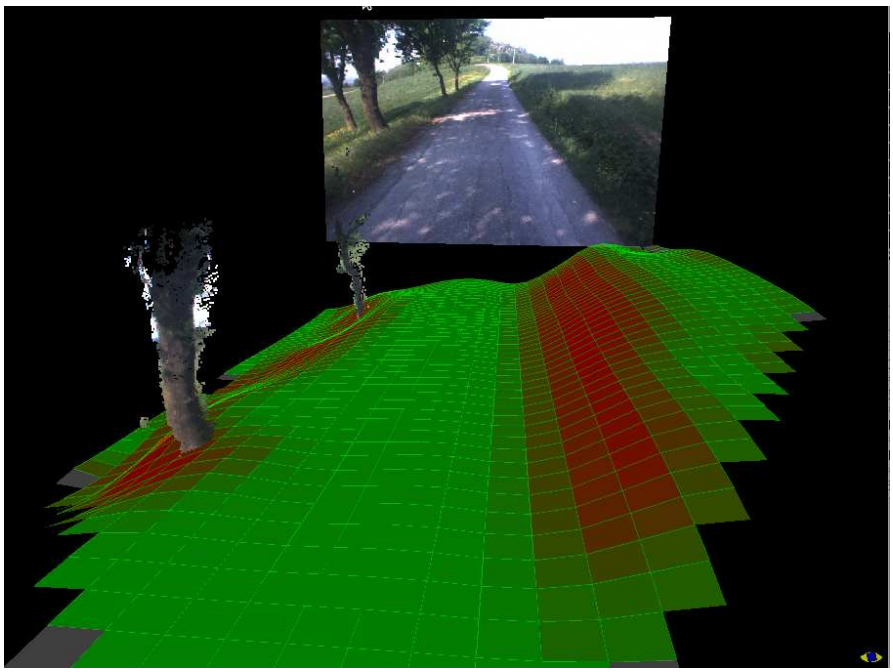


Figure 3.2: Example of terrain estimation result.

The approach can be described as a 6 steps procedure:

1. *3D points sampling*. A data set of 3D world points (x_i, y_i, z_i) is generated. The data set must provide enough points to support the desired resolution over the

area of interest. Implementation details in Section 2.1.5. Each 3D point is projected onto its corresponding cell, that will contain every point that belongs to its volume, regardless they represent terrain, an object, or spurious noise. The number of cells and their size are defined by the region of interest and the required resolution. Resulting map has $m_0 \times m_1$ Cartesian cells, each one characterized by (x,y) coordinates and z height, hereinafter called $p(x_i, y_i, Z_i)$. Details on sampling method are described in Section 3.1.3.

2. *B-Spline surface fitting and ground estimation*. The goal is to extract the *main terrain surface* from the sampled points grid $p(x_i, y_i, Z_i)$: a 3D surface representing the terrain where it is possible to drive safely and where objects stand on. Section 3.1.4 shows how repeated least square fitting and equalization of sampled points, with different patterns of B-Spline control points, order, and weights, leads to the desired segmentation in an efficient way.

3. *Slope estimation*. Once the terrain surface is available in B-Spline form, terrain slope can be easily computed for any (x,y) location contained into the interest area just by derivation.

4. *Obstacles detection*. Discarded points (x_i, y_i, z_i) at previoius step that not belonging to the terrain are represented as obstacles candidates.

5. *Traversability cost*. Traversability cost Tc for each (x,y) location is a function of corresponding slope and occupancy information: the steeper the slope, the higher the cost; similarly, the more occupied the terrain portion, the higher the cost of its traversability. The traversability cost is detailed in Equation 3.1 where w_{occ} is the occupacy cost, w_{slope} is the steepness cost, $isOcc(x,y)$ is a normalized likelihood of the cell occupancy (the max value is returned if there is an obstacle on the cell), and $S^{'}(x,y)$ is the B-Spline first derivative.

$$Tc(x,y) = w_{occ} \cdot isOcc(x,y) + w_{slope} \cdot S^{'}(x,y) \tag{3.1}$$

3.1.3 3D points engine

The main focus of the sampling processing, applied among the 3D world points obtained by the disparity map, is simplifying the traversable terrain reconstruction, highlighting, in the 3D cloud, all points belonging to the terrain slope and, at the same time, attenuating the spurious noise contribution. Moreover, the grid quantization allows to reduce the problem of computational complexity, transforming the dense stereo depth map into a sparse grid representation.

Each world point (x_j, y_j, z_j) is projected, according to its position, on a $m_0 \times m_1$ 2.5D grid, whose cells are used to locally accumulate the points.

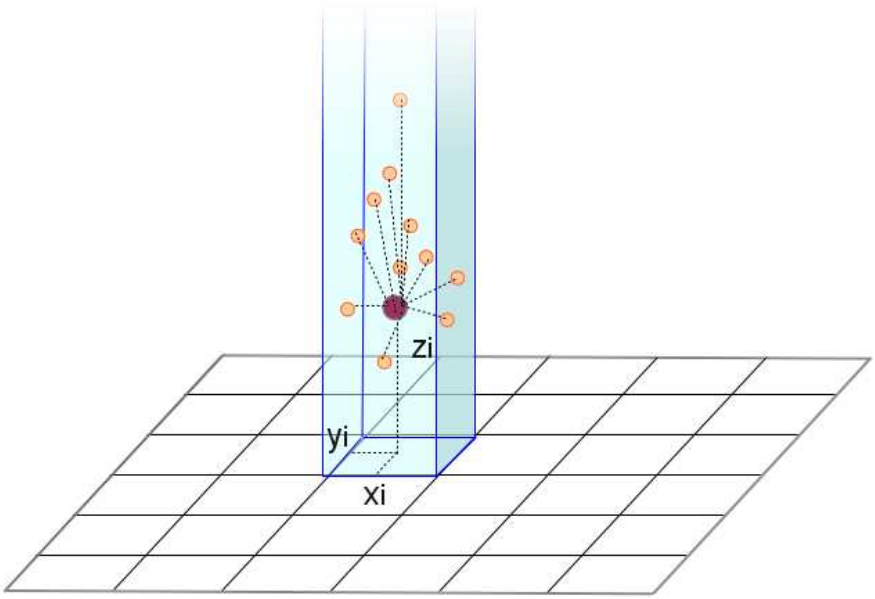


Figure 3.3: 3D world points quantization processing.

As shown in Figure 3.3, for each cell c_i all 3D points, belonging to its volume, are condensate in a single point $p(x_i, y_i, Z_i)$, with x_i and y_i equal to the cell centre coordinates (p_x, p_y), and Z_i calculated as:

$$Z_i = max(min(z_j), mean(z_j) - \sigma) \tag{3.2}$$

where:

- $min(z_j)$ is the height of the lower world point in c_i;
- $mean(z_j)$ and σ are, respectively, the average and the standard deviation of the 3D points heights belonging to the volume of each cell c_i.

When none of 3D points belongs to a given cell, its corresponding Z_i is computed averaging valid neighbours' heights. If no valid neighbours are available, cells are marked as *INVALID*, and $Z_i := 0$. Section 3.1.4 explains how B-Spline fitting handles invalid cells.

This mapping, through the local evaluation of the 3D points distribution, allows to maximize the contribution of the 3D points characterized by low height values, that are likely to represent the terrain slope. The determination of the mapped Z_i is more influenced by low z_j values, regardless of whether the corresponding world points heights vary in a small or in a large range. The standard deviation evaluation is used to attenuate the outliers contribution. Moreover all mapped heights are maintained in the range defined, locally, by the highest and lowest world point.

3.1.4 B-Spline surface fitting

A rational B-spline tensor product surface of order d_0 and d_1 is a $R^2 \rightarrow R$ function defined in Equation 3.3 as follows:

$$\mathbf{S}_{\mathbf{R},\mathbf{Q}}(x,y) = \sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} \frac{\mathbf{R}_{i_0,i_1} N_{i_0,d_0}(x) N_{i_1,d_1}(y) \mathbf{Q}_{i_0,i_1}}{\mathbf{R}_{i_0,i_1} N_{i_0,d_0}(x) N_{i_1,d_1}(y)} \tag{3.3}$$

where: $\mathbf{Q}$ is a 2-dimensional array of $(n_0+1) \times (n_1+1)$ control points; d_0 and d_1 are Spline's degrees along the two axes, with $1 \leq d_0 \leq n_0$ and $1 \leq d_1 \leq n_1$; $\mathbf{R}$ is the corresponding control points' weights matrix.

The set of $N_{i,d}(t)$ are the Spline's basis functions and are detailed in Equation 3.4 where t is a non-decreasing sequence of scalars t_i for $0 \leq i \leq n_{0,1} + d_{0,1} + 1$ known as *knot vector* [25]. In this section, the vectors (x,y) are world coordinates, while $\mathbf{S}_{\mathbf{R},\mathbf{Q}}(x,y)$ are the corresponding terrain's height z.

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t) \tag{3.4}$$

with

$$N_{i,0}(t) = \begin{cases} 1 & t_i \leq t < t_{i+1} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

Figure 3.4 shows an example of surface fitting using B-Spline basis along x and y directions.

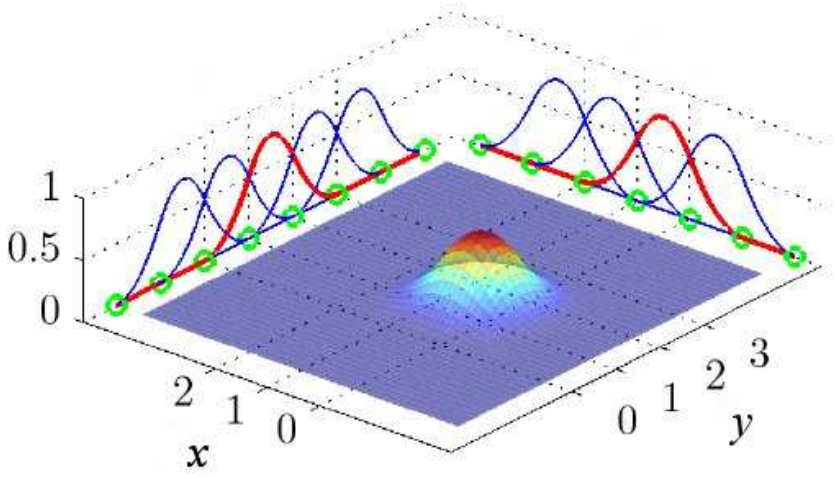


Figure 3.4: Example of surface fitting with basis in x and y

Control points $\mathbf{Q}_{i_0,i_1}$ are typically obtained by *Least Squares Fitting*, from a set of $m_0 \times m_1$ sample points $\mathbf{P}$ with known z. In particular, the proposed fitting module starts with *Weighted Least Squares Curve Fitting* [25] along the X world coordinate, where to each sample is assigned a proper amount of influence $\mathbf{W}(z_i)$ over the parameter estimates. Then the resulting control points are fitted across the Y direction, to compute the final $\mathbf{Q}_{i_0,i_1}$ control points matrix. As mentioned, here the weights refer to *sample points*, instead of control points, and are used to increase or decrease the influence of a given sample on the final square error.

3.1.5 Ground points extraction

Basically the algorithm consists of a repeated least-squares B-Spline fitting, using surfaces with a number of control points increasing iteration after iteration, and where control points' weights and samples' weights, at a given iteration, are adapted on the basis of the previous surface.

The idea is to fit sample points with a very simple B-Spline, not weighed and not rational, with very few control points, in order to understand the principal terrain slope and shape; then, the closer a sample point is to this surface, the higher its weights will be at the next iteration. Moreover, to speedup the procedure, samples are also *equalized*: they are moved towards the surface, the higher the distance, the higher the equalization. This procedure ends when the maximum number of control points n_{max} (always $\leq min(m_0, m_1)$) is reached or when no point has been equalized in the last step. Algorithm 3.1.5.1 summarizes the process in pseudo-code.

Algorithm 3.1.5.1 Iterative least-squares fitting of **S**

$\mathbf{W} \leftarrow \texttt{identity\_matrix}(\texttt{m}_0 \times \texttt{m}_1)$
$\mathbf{R} \leftarrow \texttt{constant\_matrix}(1, (\texttt{n}_0 + 1) \times (\texttt{n}_1 + 1))$
procedure **Terrain**($\mathbf{P}$)
do
$\mathbf{P}_{\mathbf{prev}} \leftarrow \mathbf{P}$
$\mathbf{Q} \leftarrow \texttt{spline\_fitting}(\mathbf{P}, \mathbf{W}, d_0, d_1, \delta n_0, \delta n_1)$
$\overline{e} \leftarrow \texttt{linear\_mean}(\|\mathbf{P} - \mathbf{S}_{\mathbf{R},\mathbf{Q}}\|)$
$\mathbf{W} \leftarrow \texttt{compute\_weights}(\mathbf{P}, \mathbf{S}_{R,Q}, e)$
$\mathbf{R} \leftarrow \texttt{remap\_weights}(\mathbf{W}, \delta n_0, \delta n_1)$
$\mathbf{P} \leftarrow \texttt{equalization\_filter}(\mathbf{P}, \mathbf{S}_{R,Q}, \overline{e})$
$\delta n_i \leftarrow \delta n_i + 1$
while $(\delta n_i < n_{max\_i}, \quad i = 0, 1) \wedge (\|\mathbf{P}_{\mathbf{prev}} - \mathbf{P}\| < \varepsilon)$
$\mathbf{Q} \leftarrow \texttt{spline\_fitting}(\mathbf{P}, \mathbf{W}, d, n_0, n_1)$
end

As mentioned $\mathbf{P}$ is a $(m_0 \times m_1)$ samples matrix containing the Z_i values of each sample, while $\mathbf{R}$ and $\mathbf{W}$ are control points and sample points weights matrices respec-

tively. The **R** are computed downsampling and interpolating the larger **W** matrix.

Function `compute_weights` in Algorithm 3.1.5.1 assigns to each sample point a fitting weight according to the following formula:

$$\mathbf{W}(Z_i) = 1 + \left(\frac{|\Delta z|}{th(\bar{e})} \right)^{-\alpha} \tag{3.6}$$

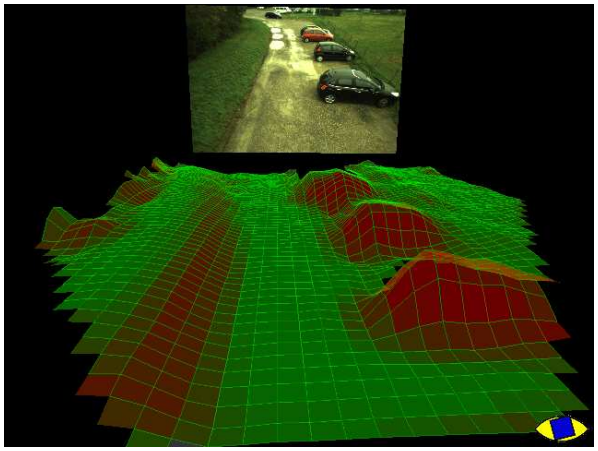
where $th > 0$ is the weighting threshold, proportional to the current average fitting error, while $\alpha \geq 1$ is the weighting speed, proportional to the current iteration number. Note how $\mathbf{W}(Z_i)$ is always greater than 1.

The `equalization_filter` in Algorithm 3.1.5.1 moves sample points towards the last computed surface, according to this logic:

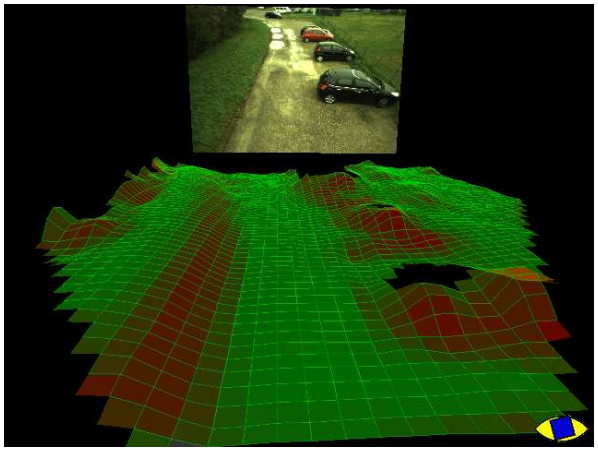
$$Z_i = Z_i - \Delta z \cdot (1 + \frac{1}{\mathbf{W}(Z_i)}) \tag{3.7}$$

Actually not all sample points are equalized: valid points (see Section 3.1.3) with $\Delta z < 0$ are not modified. The goal is to find a good *terrain* surface estimation, that must always be made of the *lowest* visible 3D points. Under this point of view, it is counterproductive to rise points to the current approximated surface. Note how adjustments are always smaller than Δz, hence points never overshoot the current surface.

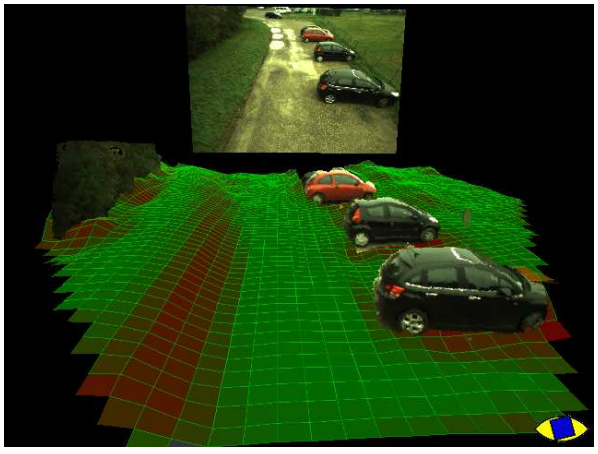
In Figure 3.5 it is shown the comparison between a simple B-Spline surface and the proposed iterative method, applied on the same samples: a simple B-Spline fitting (a) compared with the proposed iterative least-squares fitting (b) and the resulting obstacle segmentation (c). Note how the terrain is not affected by the equalization, while obstacles are noticeably lowered, allowing effective obstacle detection.



(a)



(b)



(c)

Figure 3.5: Examples of B-Spline processing.

3.2 Obstacle Detection

In Section 3.1 a terrain estimation system is presented from whom part of 3D input points, generated by the stereo reconstruction (see Section 2.1.5), are discarded. This section contains a general overview about the obstacle detection systems that are successfully performed on intelligent ground vehicles where a selected set of these ones have been also tested for this work achieving a solution based on the result obtained in Section 3.1: without assumptions on the observed environment. On basis of this theory, it is possible to deduce the concept of obstacle for an autonomous/intelligent vehicle: *a not traversable object that it stands or floats on the ground.*

In Section 3.2.1 a macro taxonomy of the obstacle detection systems, based on stereo vision for autonomous/intelligent vehicles, is presented in order to understand the motivation about the final solution. After this analysis, in order to provide a full and reliable 3D obstacle detection, the candidate proposes three solutions, detailed in Section 3.2.2, 3.2.3 and 3.2.4 as follows:

1. *probabilistic occupancy map approaches*
2. *voxel approach*
3. *stixel-based occupancy map approach*

3.2.1 Relative works

Obstacle detection (OD) is one of the main control system components for intelligent ground vehicles [27] since a reliable perception of the real world is a key-feature for any obstacle detection system for dynamic environments.

In last years, most of the historical approaches in literature have been readjusted in the framework of stereo vision and other 3D perception technologies (e.g. LIDAR) and important results have been provided by several experiments on autonomous ground vehicles [1, 3, 4]. In order to achieve a good performance, most of the OD algorithms needs some assumptions about the ground [28] or about the approximated free space on it [23, 24, 29].

The obstacle detection field is a very broad one and a lot of obstacle detection systems have been developed in the last years in this domain [30]. An algorithm can be considered reliable and accurate if it provides: a real-time output, a stable and reliable tessellation of the environment, a robust state estimation of the obstacle detected and working regardless of lighting and weather conditions. Stereo vision errors and general performance have been widely discussed in literature [7, 31]; Matthies et al. [21] show a practical approach to evaluate the performance of an obstacle detection algorithm.

In this section a brief description on obstacle detection algorithms based on stereo vision and other 2D/3D sensors is proposed. Each obstacle detection system is focused on a specif tessellation or clustering strategy, hence they have been categorized into 4 main models:

1. *probabilistic occupancy map*
2. *digital elevation map*
3. *scene flow segmentation*
4. *geometry-based cluster*

3.2.1.1 Probabilistic Occupancy Map

The main model of the probabilistic occupancy map (POM) is proposed by Elfes [32]: *occupancy grid mapping*. It is one of the most famous approaches in literature.

The world is represented as a rigid grid of cells containing a random variable whose outcome can be free, occupied, or undefined (not mapped). For a proper formalization of the problem, let us consider a regular lattice D of X_i Random Variables with outcomes in a finite set of labels. The Elfes model requires 3 possible states: free, occupied, unknown. Let us call Ω the Phase Space and Z_t the measurements performed at t time.

For a proper formalization of the problem, let's consider a regular lattice $\mathscr{D}$ where each lattice node is uniquely addressed by an index $i \in |\mathscr{D}|$ and bijectively associated

to each grid cell. Then a Random Variable X_i is associated to each of the above-mentioned nodes, with outcomes in L a discrete finite set of labels. The Elfes model requires $L = \{l_f, l_o, l_u\}$ with l_f free cell, l_o occupied cell and l_u undefined cell.

Thus it is possible to define

- Ω Phase Space for the grid, representing the set of all of the possible configurations
- $\sigma \in \Omega$ one of the possible configurations
- $X = \{X_i\}_{i \in |\mathscr{D}|}$ Random Variable composed of all of the local Random Variables
- Z_t Measurement performed by sensors at time t

The goal consists in computing $P(X, \{Z_\tau\}_{\tau=1,\dots,t})$ the associated joint distribution depending on a set of measurements carried out on a certain discrete set of time moments. Usually some assumptions are made in order to simplify the problem, namely *spatial conditional independence* and *temporal stochastic independence*.

The first kind of assumption usually made concerns spatial conditional independence among all of the variables hence leading to a factoring PDF over all of the nodes such as

$$P(X|\{Z_\tau\}_{\tau=1,\dots,t}) = \prod_{i=1}^{|\mathscr{D}|} P(X_i|\{Z_\tau\}_{\tau=1,\dots,t}) \tag{3.8}$$

Another assumption regards the temporal stochastic independence hence leading to

$$P(X|\{Z_\tau\}_{\tau=1,\dots,t}) = P(X|Z_t) \tag{3.9}$$

In order to simplify the notation let us consider m as the random variable associated to a generic cell. The occupancy value of a cell m is determined using a probability density function given measurements z_t:

$$p(m|z_1, \dots, z_t) \tag{3.10}$$

Equation 3.10 represents the state of the cell m given the measurements $z_1, \dots, z_t$. Maps can defined over high-dimensional spaces. Assuming a 2D occupancy grid

space and *static world*, namely the conditional independence among sensor reading given the knowledge of the map, the posterior density function in Equation 3.10 is reformulated in terms of log-odds as defined by Thrun [33]

$$p(m_{x,y}|z_t) = 1 - [e^{l_{x,y}^{(t)}}]^{-1} \tag{3.11}$$

with

$$l_{x,y}^{(t)} = l_{x,y}^{(t-1)} + \log \frac{p(m_{x,y}|z_t)}{1 - p(m_{x,y}|z_t)} - \log \frac{p(m_{x,y})}{1 - p(m_{x,y})}$$

and

$$l_{x,y}^{0} = \log \frac{p(m_{x,y})}{1 - p(m_{x,y})}$$

The log-odds regarding $p(m_{x,y}|z_t)$ in Equation 3.11 are recursively estimated through the Bayes rule, updating the cell value in different moments. More details are described in Thrun [33].

Figure 3.6 shows the representation of a depth sensor measure in a 2D occupancy grid. Grey cells have unknown occupancy values, white cells are free and black cells are occupied. The main advantages of this method are the following ones: it is easy to construct and it can be as accurate as necessary.

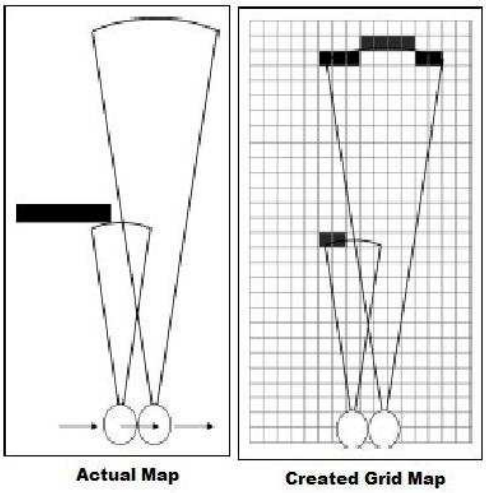
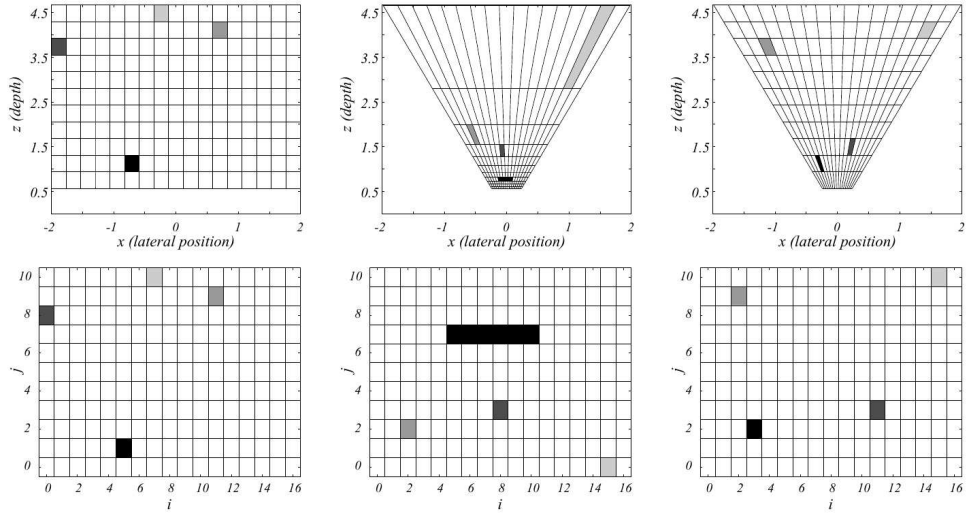


Figure 3.6: Example of the generation of occupancy grid map

A new set of stochastic occupancy grid models are detailed in Badino et al. [29]. Figure 3.7 shows a representation of these probabilistic maps along with the corresponding projections in world coordinates. The figures on the top show how the world points are discretized into a top-view representation while on the bottom show the corresponding occupancy grids. The cells have been marked with the same label of the corresponding world projections. In this work the authors illustrate an innovative way to map the measurements computed by stereo vision.



(a) Cartesian (b) Column/Disparity (c) Polar

Figure 3.7: Badino et al. occupancy grids for Disparity Space Image (DSI).

The disparity map represents the measurement processed by the algorithm to estimate an occupancy grid map. An estimation in world coordinates from the disparity map is implemented according to Equation 3.12 that is using a projection camera model based on the intrinsic and extrinsic parameters of the cameras.

$$p_k = P^{-1}(m_k) = \frac{baseline}{d} \cdot \begin{pmatrix} (u-u_0) \\ (v-v_0) \cdot \frac{fu}{fv} \\ fu \end{pmatrix} \tag{3.12}$$

where $m_k = (u, v, d)^T$ is a combination of (u, v) image coordinates and d the corresponding disparity computed by stereo, and $p_k = (x, y, z)^T$ the world point location of the m_k.

In the Badino's paper every cell of each grid maintains an *occupancy likelihood* $D(i, j)$ regarding the represented world area.

$$D(i, j) = \sum_{k=1}^{M} L_{ij}(m_k) \tag{3.13}$$

with M the number of measurements.

Equation 3.13 shows a definition of the function $D(i, j)$ where L_{ij} represents the occupancy likelihood for cell (i, j) given measurement m_k.

According to the Elfes model (see Equation 3.10) each occupancy likelihood function has been designed as a Gaussian probability density function G_{m_k}. A different function $L_{i,j}$ has been defined for each occupancy grid model.

In [29], the authors present 3 occupancy grid maps to tessellate the measurements by stereo:

1. *Cartesian grid*. The world is represented by a cartesian grid and mapped linearly to a grid of fixed dimensions (see Figure 3.7). Let us assume that cell (i, j) of the cartesian grid is centered at world coordinate (x_{ij}, z_{ij}). The likelihood function for cell (i, j) is represented in Equation 3.14.

$$L_{ij}(m_k) = G_{m_k}(P(p_{ij}) - m_k), \qquad p_{ij} = (x_{ij}, y, z_{ij})^T \tag{3.14}$$

   For each point $p_{i,j}$, the y is the triangulated measurements height obtained with Equation 3.12.

   The Gauss factor of the probability density function G_{m_k} is dependent on the difference between measurement and the reprojected cell position. Thus, the maximum likelihood factor is given to the cell which contains the triangulated measurement (see Figure 3.8a).

   In a normal implementation the authors declare that updating every cell of the grid could be time consuming hence not suitable for a real-time applica-

tion. They suggest to update only the cells significantly affected by the current measurement setting a proper distance threshold (e.g. Mahalanobis distance) between its projections.

2. *Column/Disparity grid*. The cells of the column/disparity grid correspond to discretized values of the u and d image coordinates. The occupancy grid criteria is based on mapping the measurements into (u,d) space assuming that a cell (i,j) corresponds to a coordinate $(u_{i,j},d_{i,j})$ as shown in Figure 3.7. In Equation 3.15 the likelihood function for the cell (i,j) is represented.

$$L_{ij}(m_k) = G_{m_k}((u_{ij}-u,0,d_{ij}-d)^T) \tag{3.15}$$

Figure 3.8b shows an example of the L_{ij} function.

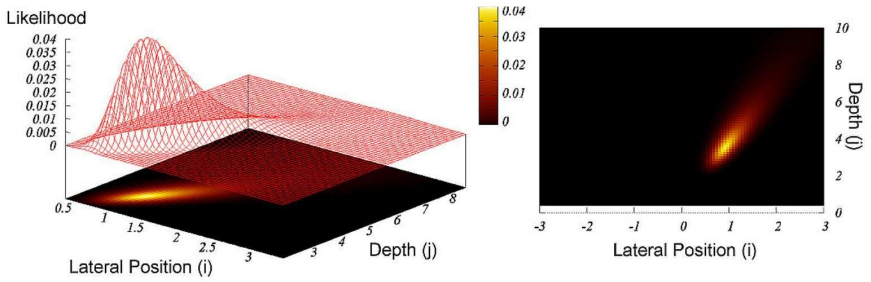
The disappearance of the v component in the measurement m_k regarding Equation 3.15 is due to the projection criteria onto the grid.

3. *Polar grid*. The mapping criteria of the polar occupancy grid is represented by the discretization of the values (u,z) where u corresponds to the column value in image space and z is the depth in the world coordinate system. The likelihood function for a generic cell (i,j) is detailed in Equation 3.16.

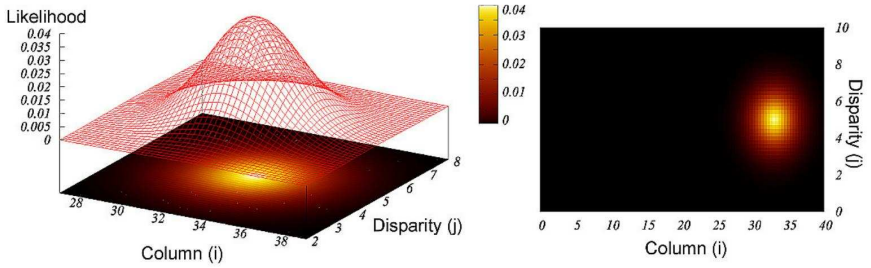
$$L_{ij}(m_k) = G_{m_k}((u_{ij}-u,0,\frac{f_u \cdot baseline}{z_{ij}}-d)^T) \tag{3.16}$$

Figure 3.8c shows an example of the L_{ij} function.

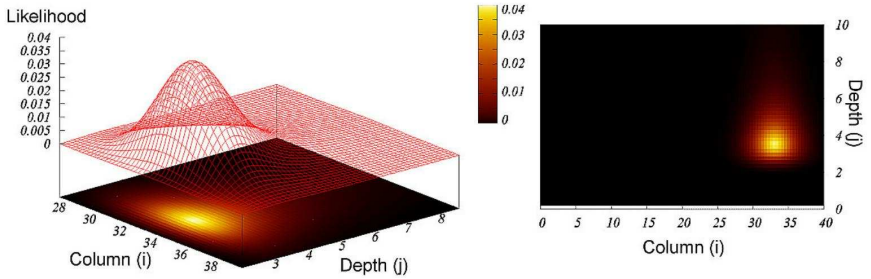
As suggested by the authors, this solution overcomes the problem of the column/disparity approach: the decreasing resolution to distant points. The result can be easily evaluated comparing Figure 3.7b and 3.7c.



(a) Cartesian Occupancy Grid.



(b) Column/Disparity Occupancy Grid.



(c) Polar Occupancy Grid.

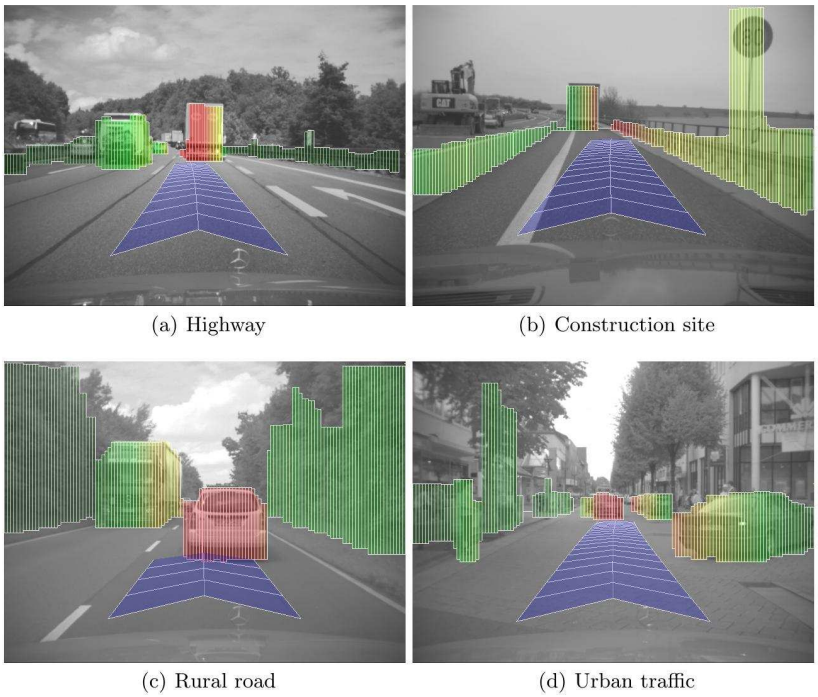
Figure 3.8: Examples of the likelihood function Li, j related to the same measurement for each occupancy grid map.

When it is possible to make proper assumptions about the road model and the vehicle pose, successful solutions regarding the obstacle segmentation problem have been proposed, in some cases working in image coordinates by using v-disparity space [22].

Most of the recent obstacle detection algorithms have been developed in the oc-

cupancy grid maps framework. Recently one of the major contribution has been given by Badino et al. [34] with the *stixel* representation.

Stixel tessellation. The basic concept of this approach is the world representation into a set of rigid clusters called *stixels* by the authors, each obstacle is hence described as a union of these elements.



(a) Highway (b) Construction site

(c) Rural road (d) Urban traffic

Figure 3.9: Stixel output in real scenarios.

A stixel based obstacle detector requires the following tasks to be performed:

1. a polar occupancy grid mapping of the measurements computed by stereo;
2. background and foreground of disparities using the previous polar grids;
3. height segmentation to estimate the heights of each stixels.

The last task is performed computing an upper boundary on the disparities cost image by means of dynamic programming. This part is detailed in Badino [34]. Real-time results are shown by the authors. They present a reliable obstacle detection algorithm that runs on an Intel Quad Core 3.00 GHz processor in 25ms. An evaluation of the stixel approach in different real world road scenarios is shown in Figure 3.9 where the color encodes the lateral distance to the driving corridor.

3.2.1.2 Digital Elevation Map

A *Digital Elevation Map* (DEM) is a height-based representation of the measurements into a map like a cartesian occupancy grid (see Section 3.2.1.1). This approach is widely applied mainly for terrain mapping [35]. A DEM can be computed with any 2D or 3D sensor (e.g. stereo vision, LiDAR and radar). Following the DEM-based approach, one of the major contributions for obstacle detection has been proposed by Oniga et al. [23]. The authors present a complete system for road surface estimation and obstacle detection in urban scenario.

In this work a DEM and two density maps are computed from the set of 3D points to obtain a compact representation with explicit connectivity between adjacent 3D locations. The road surface is fitted using a RANSAC approach to a small patch in front of the ego vehicle. To exploit this representation, the authors also propose an obstacle detection algorithm based on the density of 3D points per DEM cell (as a measure of the local slope). The density-based algorithm for obstacle detection is based on the density of 3D points: each DEM cell is classified as obstacle or road using a slope-based threshold criteria. Qualitative results are illustrated in Figure 3.10.

The authors claim that, due to use of software-specific C optimizations and the DEM representation, an average processing time of 22ms has been achieved for the whole algorithm (on Pentium 4 at 2.6 GHz). Furthermore, with the image acquisition and the dense hardware reconstruction, a sustained processing frame rate of 23 frames/s has been obtained. From the performance point of view, a set of false positive and negative results is also presented.

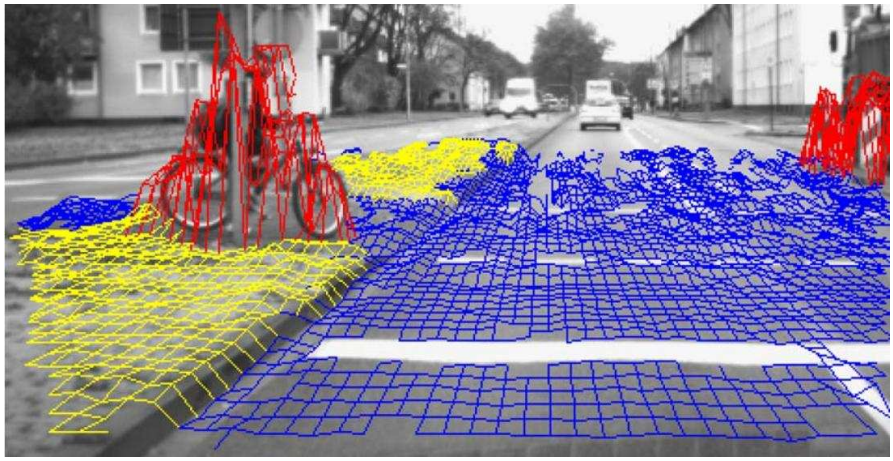


Figure 3.10: An example of Oniga's DEM-based approach: road (blue), traffic isles (yellow) and obstacles (red).

An important contribution has been brought by Danescu et al. [36], consists in applying the particle filter strategy to perform DEM tracking. The authors define this approach *Dynamic DEM*.

3.2.1.3 Scene Flow Segmentation

This technique, at first known as *optical flow*, is based on the temporal correlation to estimate the motion between two frames captured by camera at different times. In literature there are so many papers that show the implementation of an optical flow algorithm for obstacle detection but not many techniques guarantee a real-time processing [37], [38], [39]. Because of recent improvements in 3D reconstruction techniques by stereo, in the last years this approach has evolved into a new one: *scene flow estimation*.

Figure 3.11: Scene flow estimation of 6D vision

A notable 3D approach is the so called 6D vision [6], where each 3D point (computed by an FPGA stereo system) is tracked by means of an efficient GPU Optical Flow implementation. An important study has been presented by Lenz et al. [40] where the scene flow estimation is computed by means of a temporal correlation regarding two couples of frames acquired by stereo, this is essentially a *visual odometry* based approach following Geiger [8]. Qualitative results for the Franke and Lenz approach are shown in corresponding Figure 3.11 and 3.12. Figure3.12 show the Lenz's approach: (a) slowly moving and small objects such as the pedestrian in the first two frames are detected in a range up to 30 m. However, the track of such a small object is interrupted and it is not continuously tracked. Similarly moving groups of pedestrians are detected as one object since the scene flow difference is not unique and the number of detected interest points is to low. (b) Moving objects in this sequence are detected, but especially for the far range static objects are detected as well. (c) Turning cars and partly occluded objects, which where fully visible and observed in a previous frame, are detected.

Figure 3.12: Output of Lenz's approach in urban scenarios.

In [40] the authors present an algorithm that runs in Matlab processing at least one frame per second on one core of an Intel Core2 Duo with 2.4 GHz and 4 GB RAM computing grayscale images with a resolution of 1392×512 pixels and at least 2000 interest points have been detected for each image. An hybrid approach regarding moving obstacles has been presented by Rabe et al. [41]. This work is also developed in the framework of 6D vision project. The core strategy in this application relies on the principle of fusing optical flow and stereo information given in [6]. The basic idea is hence to track points with depth information determined by stereo vision over two or more consecutive frames and to fuse the spatial and temporal information using Kalman Filters exploiting also an egomotion information. The algorithm is tested on a 3.2 GHz Pentium 4 computing 2000 image points with a cycle time of 40-80ms. Figure 3.13 show an example of the approach described.

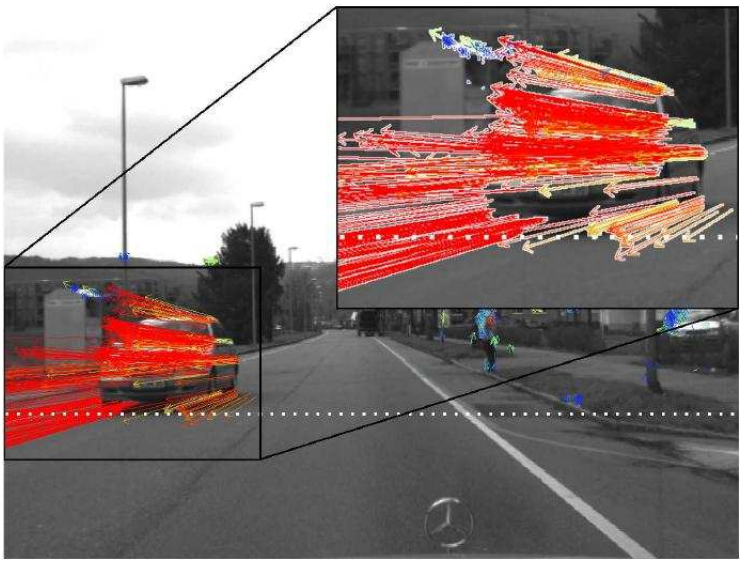


Figure 3.13: Rabe's moving obstacle detector: velocity estimation results for an oncoming car

3.2.1.4 Geometry-based Cluster

In this generic category it has been decided to present only the solutions in literature that provided real-time results. The strategy that can best describe this category is Manduchi et al. [27]. In this work, the authors have postulated the first obstacle detection approach for any dimensional model (3D also). Indeed they designed an algorithm based on a search method that clusters the sensors measurements using a double cone model (see Figure 3.14). The proposed 3-D obstacle search method using double cone that locates ground pixels (brown) and obstacle pixels (blue). The whole set of 3D points is projected on the frontal plane with respect to the camera (in order to simplify computations) then a scanning through the projected points set is performed. For each point, the double cone mask (projected it becomes a double triangle mask) is applied and if any other point is found in the region of interest they and the initial point are classified as an obstacle otherwise the initial point is classified as ground.

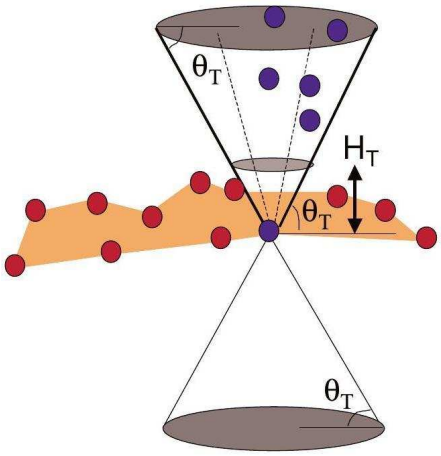


Figure 3.14: Manduchi double cone

The performance of this approach are widely evaluated in [16]. The authors described their stereo obstacle detection algorithm based on [27] showing the performance elaborated during an intercontinental experiment on their autonomous ground vehicle [1]. A similar approach was also used in PROUD test [4]. In order to achieve real-time processing, the authors have postulated a smart segmentation method along the disparities. The processing times are detailed in Table 3.1.

PROCESSING TIMES

Algorithm step	Processing Time [ms]			
	Intel® Core™ i7 920		Intel® Core™ 2 Quad Q9100	
	SGM	SAD	SGM	SAD
Preproc.	2.9	2.9	4.9	5.0
DSI	21.5	5.8	60.2	7.8
DSI filt.	2.8	2.8	6.5	6.7
Obstacle det.	32.1	25.2	59.2	54.1
Total	59.3	36.7	130.8	73.6

Table 3.1: VIAC processing times.

Figure 3.15 shows the qualitative results of the Broggi et at. [16] in different scenarios during the *VIAC* using a DSI computed at 128 disparities: in Figure 3.15(a) - (c) a busy motorway in Kiev, (d) - (f) country roads with woods and uphill sections, (g) a deserted mountain motorway in Kazakhstan, (h) a raindrop on the right camera and (h) an upcoming tractor.

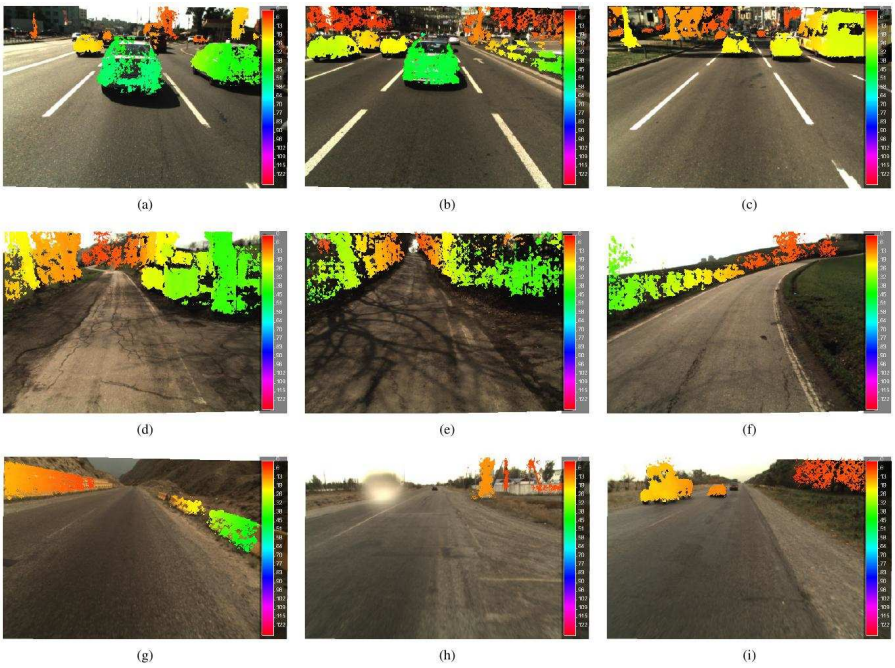


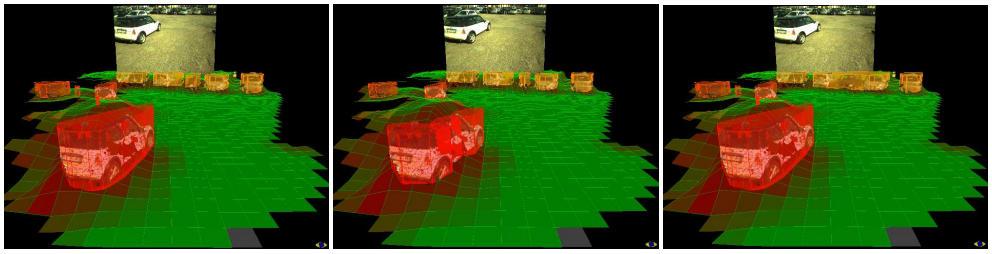
Figure 3.15: Obstacle detection results in VIAC.

3.2.2 Probabilistic occupancy map approach

On the basis of the theories cited in Section 3.2.1, in order to compute a fast and reliable information about the obstacles for an intelligent vehicle, a set of probabilistic occupancy map approaches is implemented following the Badino et al. theory [29].

The 3D input data are based on the discarded points generated by the terrain estimator as detailed in Section 3.1 and processed as described in Chapter 2. Figure 3.16 shows the output generated by each approach: (a) cartesian occupancy map,

(b) column/disparity and (c) polar grid map. The clustering strategy applied for each obstacle is based on a common convex hull algorithm: *gift wrapping*. The color cluster is used to suggest how far is an obstacle from the vehicle (e.g. red means close).



(a) (b) (c)

Figure 3.16: Output of probabilistic occupancy map approaches.

Tests and results will be detailed in Chapter 4.

3.2.3 Voxel approach

The approaches described in Section 3.2.2, do not exploit completely the 3D information provided by modern stereo matching algorithms. The complex cluster map shows full 3D information using adjacent stack of cells [19], or octrees connected cubes [20], and are able to represent objects at multiple heights located at the same range and azimuth. Full 3D clusters show the great advantage of adequately representing obstacles with non conventional shapes, like *concave* ones.

In this section the candidate introduces an obstacle detector based on a full 3D scene reconstruction to estimate both stationary and moving objects with minimum assumptions of the road model. A modelling of the 3D point cloud, derived from a disparity image, into an accurate voxel reconstruction is necessary to build complex clusters. The 3D input source is based on the same processing of the Disparity Space Image (DSI) cited in Chapter 2 and applied in Section 3.1. These 3D data contain the geometric and texture information and are perfectly suitable to make a segmentation based on a flood fill approach.

A vehicle pose estimation is needed to determinate the velocity and position of obstacles detected; for this reason it has been developed an *egomotion* estimator based on the *visual odometry* approach introduced in [8]. The developed approach for obstacle detection receives as input the point cloud from the 3D points engine and returns a list of obstacles, estimated in terms of position, volume and speed. This algorithm can be divided into three main steps. First, the positional 3D information of each point (x_i, y_i, z_i) is used to partition the cloud into voxels. Then, a color-based clustering algorithm is used to obtain the list of obstacles from the set of voxels. Finally, each obstacle is tracked, in order to strengthen the output and estimate the obstacle speed.

Through a temporal interpolation of previous 3D voxel reconstructions, the objects above the ground can be easily detected and their velocity and position can be estimated using a Kalman filter; more complex and similar approaches exist in literature [6].

The approach can be described as a 4 step procedure:

1. *Egomotion estimation*. A pose estimation of ego-vehicle is needed to distinguish between stationary and moving objects; this is achieved by a *visual odometry* approach, described in Section 3.2.3.1.

2. *Voxel-based partitioning*. The 3D point cloud is partitioned into voxels at a certain resolution. This structure will contain the points that belong to a voxel, regardless of whether they represent terrain, an object, or spurious noise. The Section 3.2.3.2 describes how this voxel structure quantizes at each frame the 3D point cloud using the *egomotion* information.

3. *Clustering*. In Section 3.2.3.3 a colour-space segmentation approach has been used to group together voxels with similar features, considering only those voxels that are above the ground plane; a subsequent refinement and cluster aggregation phase is performed to increase robustness and remove false positives. To each cluster is assigned a unique label C corresponding to their center of mass $(C_{x_i}, C_{y_i}, C_{z_i})$.

4. *Moving obstacles*. Finally in Section 3.2.3.4, a Kalman filter is applied to prune false positives estimating the predicted speed and pose of candidates. The evaluation of these data determines which of clusters are moving obstacles.

3.2.3.1 Visual Odometry

Visual Odometry (VO) is the process of incrementally determining the position and orientation of a vehicle by examining the changes that motion induces on sequential camera images taken by its onboard cameras [42]. VO implementation used for this work is based on a variation of Geiger's approach [8] that exploit a different feature descriptor.

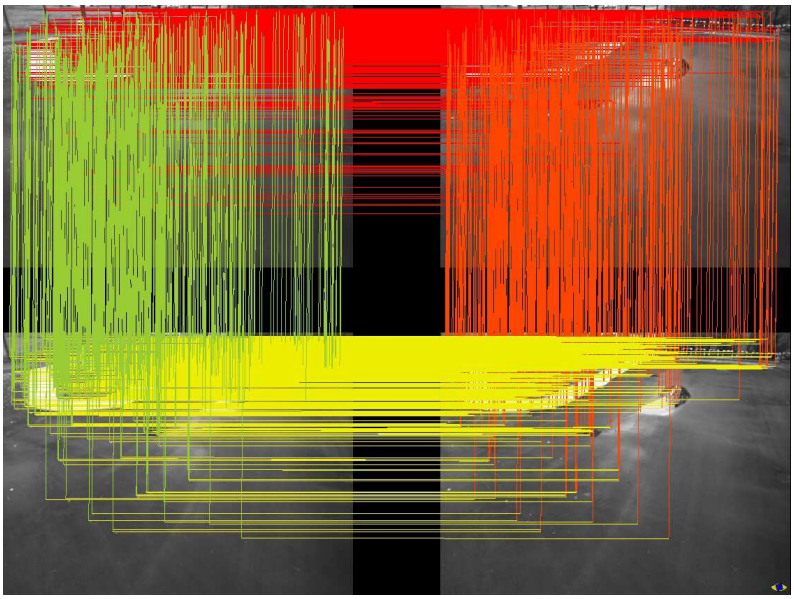


Figure 3.17: Feature matching using stereo images

Figure 3.17 shows the feature matching between two stereo images captured in different times. The top row shows two images of the last observed scene for the left and right camera; same way, the bottom row shows a frame of stereo images captured at the previous time. The similiarity metric algorithm tries to detect the

features comparing circularly the frame pair in the following order:

- (*last left frame* $\leftrightarrow$ *last right frame*);
- (*last right frame* $\leftrightarrow$ previous right frame);
- (*previous right frame* $\leftrightarrow$ *previous left frame*);
- (*previous left frame* $\leftrightarrow$ *last left frame*);

Finally the algorithm computes the rototranslation matrix that minimizes the reprojection error between the supplied correspondences via Levenberg-Marquardt nonlinear optimization.

3.2.3.2 Voxel-based space partitioning

The perception of the environment is based on the concept of *voxel*. A voxel is defined as an element of a partition of the 3D space in cube-shaped blocks, where each block is aligned along the directions of the axes of the Cartesian 3D coordinate system.

Each voxel in a grid is identified by its position and dimension (usually expressed with the center and the edge length, respectively). The set of 3D points spatially belonging to each voxel, as well as some additional information stored for convenience (the centroid of the points, the number of points, the color mean values and component variances, both in the RGB and HSL color spaces), is assigned to the same voxel. These data are stored using an efficient approach [20] saving only the needed voxels: thanks to *egomotion* information, new voxels are created only when a previously empty volume of the space is now occupied by a new 3D point; similarly, voxels are removed if no 3D points fall into it. Inserting and retrieval costs correspond to an un-ordered associative container; the implementation guarantees an $O(N)$ complexity for the worst case, $O(1)$ for the mean case. Moreover, this implementation allows the movement in the set at a constant cost (plus, eventually, the retrieving of the data), both in a Cartesian and a hierarchical (octree-based) way. This model could be extended to any n-dimensional space.

3.2.3.3 Color-based clustering

The obstacles have been assumed to be above the road plane. So, only the voxels with positive z have to be considered. The set of these voxels is used as the input of a *clustering* algorithm that segments the same set into clusters, each of them corresponding to an obstacle candidate. To each cluster is assigned the corresponding set and number of contained voxels, the color mean and variance (both in RGB and HSL color spaces), the overall bounding box and its density (where density of a cluster is defined in terms of the ratio between the number of contained voxels and the bounding box volume).

$$f(v,c) = \|\vec{I}_{RGB}(v) - \vec{I}_{RGB}(c)\|_2 + \|\vec{I}_{HSL}(v) - \vec{I}_{HSL}(c)\|_1 \tag{3.17}$$

Algorithm 3.2.3.1 Flood fill color-based clustering

```
foreach not-yet-considered voxel v do
  c <- new cluster containing v
  insert v in queue
  while not queue is empty do
    v' <- extract first element from the queue
    foreach v'' ∈ N(v')\{v_c|v_c ∈ c} do
      if f(v'',c) < ε then
        insert v'' in c
        insert v'' in queue
      end if
    end foreach
  end while
end foreach
```

A flood fill color-based clustering algorithm is used for this purpose (Algorithm 3.2.3.1), using as a similarity check the Equation 3.17.

The neighbourhood N of a voxel v is defined as the voxel-centered closed unit ball in the voxel set, measured with the Chebyshev distance.

The similarity between a cluster and a voxel is based on color. The color of the voxel is compared to the color of the cluster resulting from the colors associated to each already-inserted voxel in the cluster. Various types of color matching have been experimented: the chosen color match is based over the normalized means of the RGB and HSL components, as stated in Equation 3.17. Empirically, it has been found that the algorithm could produce small clusters, due to the noise over the disparity data, depending on the objects in the scene and on the lightning conditions. So a refinement has been added. First of all, for each small cluster a big neighbour cluster is searched. If any is found, the small one is aggregated to one of them (Algorithm 3.2.3.2). Every small, or with low density, remaining cluster is treated as noise and it is not considered in further computations (Algorithm 3.2.3.3).

Algorithm 3.2.3.2 Cluster aggregation

```
foreach cluster c do
  if number of voxels of c small
    and there is at least one voxel near c
  then
    c' <- cluster with biggest number of voxels near c
    aggregate c in c'
  end if
end foreach
```

Algorithm 3.2.3.3 Cluster noise removal

```
foreach cluster c do
  if number of voxels of c small
    or low density of c
  then
    remove c from list of clusters
  end if
end foreach
```

3.2.3.4 Moving obstacles

The reliability of the output is enhanced using a tracking algorithm. The clusters found in the current frame are associated with tracked clusters, according to a greedy policy based on a distance function (Algorithm 3.2.3.4).

Algorithm 3.2.3.4 Tracking associations generation

```
C <- set of current frame clusters
T <- set of tracked clusters
while C or T contain not-yet-considered elements do
  (t̂*, ĉ*) <- (t,c)∈T×C d(t,c)
  if d(t̂*, ĉ*) less than threshold then
    consider t̂* and ĉ* as associated
    remove t̂* from T and ĉ* from C
  end if
end while
foreach t ∈ T do
  consider t as non associated (ghost)
end foreach
foreach c ∈ C do
  consider c as non associated (new cluster)
end foreach
```

The information of cluster color, dimensions, density and predicted position have been used to create distance functions between clusters. Several linear combinations of these distances have been tested as a distance function for the algorithm.

Each tracked cluster is put in one of these three queues:

- a newly seen cluster is put in the *acceptance queue*, where it remains until it is seen for enough consecutive frames. If this does not happen, it is discarded.
- a cluster seen for enough consecutive frames is put in the *tracked queue*, where it remains as long as it is visible.
- a cluster that disappears from the view is put in the *ghost queue*. If it becomes

visible again, it is put in the tracked queue. A cluster that remains too long in the ghost queue is discarded.

Moreover, each tracked cluster has a 6-dimensional state associated to it, identified by the position and the speed of the centroid of the cluster. This state is used to predict and correct the spatial information of the cluster by a linear Kalman filter.

Tests and results will be detailed in Chapter 4.

3.2.4 Stixel-based occupancy map approach

Through a brief comparison between the previous implemented approaches, it is possible to deduce that the occupancy grid maps [29, 34], implemented in Section 3.2.2, are notably efficient but they are not able to represent concave or floating obstacles if another one is located at the same azimuth and position with different height. On the contrary, the geometry-based approaches, as seen in Section 3.2.3, and scene flow segmentation [6, 40] provide a full 3D perception of the obstacles detected but are time-consuming.

On the basis of these deductions, a new approach is developed in order to guarantee:

- the performance of probabilistic occupancy map;
- the capability to represent concave and floating obstacles.

This strategy has been included in the first approach based on occupancy map, described in Section 3.2.2, removing the convex hull representation with one based on *augmented stixels*. By this technique an obstacle can be defined as a cluster of stixels. The key features of an *augumented stixel* are represented by a modified version of the original structure proposed by Badino et al. [34]:

- fixed size for both longitudinal and latitudinal directions,
- textured 3D point cloud,
- estimation of the minimum and maximum height value from the ground.

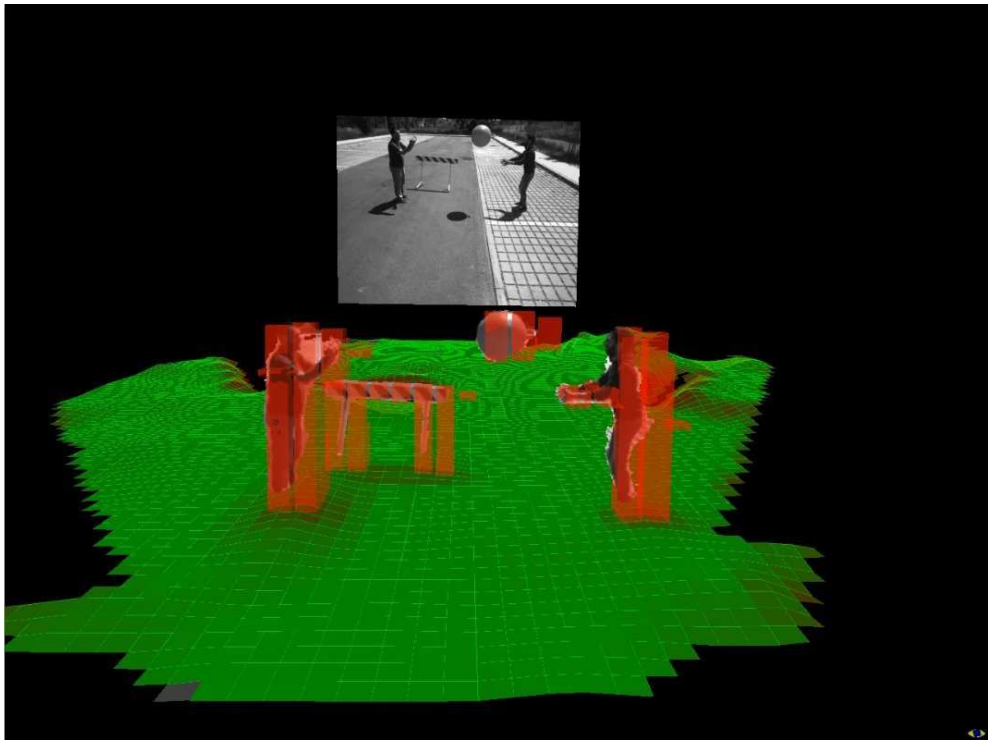


Figure 3.18: Example of output of the stixel-based occupancy map. approach

Indeed the last property allows to model concave and floating obstacles on the contrary of the original approach that it estimates only the maximum height of the stixels. Figure 3.18 shows a brief qualitative result of the implemented approach based on polar grid map. Tests and results will be detailed in Chapter 4.

Chapter 4

Tests and results

The highest goal that man can achieve is amazement.

– Johann Wolfgang von Goethe

4.1 Overview

In this chapter the tests and obtained results about the developed approaches for the terrain estimation system and obstacle detectors will be detailed as follows.

- *Perception hardware.* A brief description about the hardware setup of the camera and the calibration process.

- *Terrain estimation.* Evaluation in terms of reliability and robustness of the terrain points extraction algorithm will be describe. Also a set of qualitative results in several scenarios will be shown.

- *Obstacle detection.* In this section several tests will be analyzed in order to study and select the ideal algorithm to connect at the terrain estimation system for a real-time processing. Each obstacle detector includes a tracking system

based on a linear Kalman filter in order to estimate the motion of detected objects.

4.2 Perception hardware

This section introduces a description of the camera hardware applied for the whole tests. In order to minimize the disparity estimation error, avoiding the rectification issues detailed in Section. 2.1.3, a rigid parallel stereo camera has been mounted on the vehicle. Figure 4.1 shows three cameras at different heights and orientations; this configuration has been chosen in order to perform an exhaustive test session for the terrain estimator and obstacles detectors.

Figure 4.1: Cameras setting on a VisLab vehicle (Porter) used during the VIAC.

The cameras used to perform the tests are listed as follows:

- PointGrey BBX3-13S2C-38 color sensor with 3.8mm as focal length.
- PointGrey BBX3-13S2M-60 monochromatic sensor with 6mm as focal length.

Technical details and specifications about the used cameras are described in Appendix A.

As described in Section. 2.1.4 and 2.1.3, if the cameras are already align in order to represent a parallel stereo cameras system, the depth map values (u,v,d) in DSI coordinates can be easily translated into sensor coordinates (x,y,z) (see Figure 4.2) using the formula in Equation 4.1 where u_0 and v_0 are respectively the horizontal and vertical optical centre and k_u is the horizontal pixel focal length.

$$\begin{cases} x = \frac{baseline \cdot (u-u_0)}{d} \\ y = \frac{baseline \cdot (v-v_0)}{d} \\ z = \frac{baseline \cdot k_u}{d} \end{cases} \tag{4.1}$$

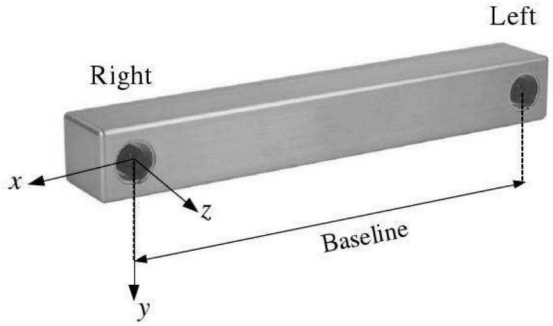


Figure 4.2: Sensor reference system.

The 3D point cloud, provided as input source for the terrain and obstacle detection algorithms, is transformed in world coordinates using the reference system visible in Figure 4.3. The rotation matrix **R** is expressed in Equation 4.2.

$$\mathbf{R} = \mathbf{R_z} \cdot \mathbf{R_y} \cdot \mathbf{R_x} \tag{4.2}$$

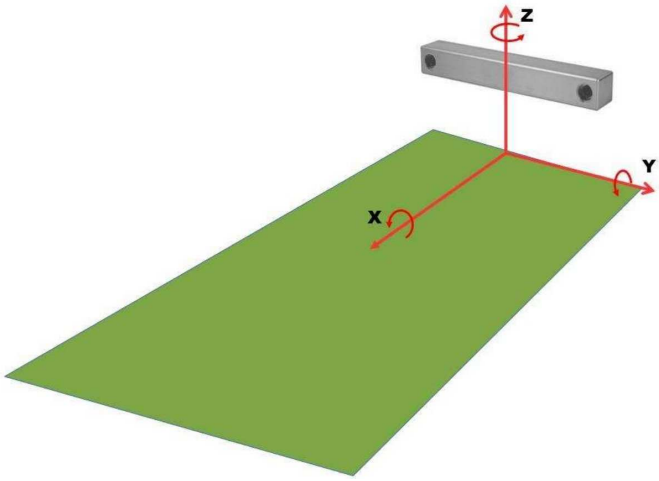


Figure 4.3: World reference system.

Figure 4.4 shows the scenario used for the calibration of the extrinsic parameters of the stereo camera.

Figure 4.4: Calibration scenario.

4.3 Terrain estimation

In order to evaluate the terrain estimation algorithm, two different kind of tests are performed. The first, detailed in Section. 4.3.1, is based on the study of the robustness and goodness of the algorithm in presence of virtual obstacles. The latter in Section 4.3.2 regards the analysis of the terrain estimation results in several scenarios and conditions.

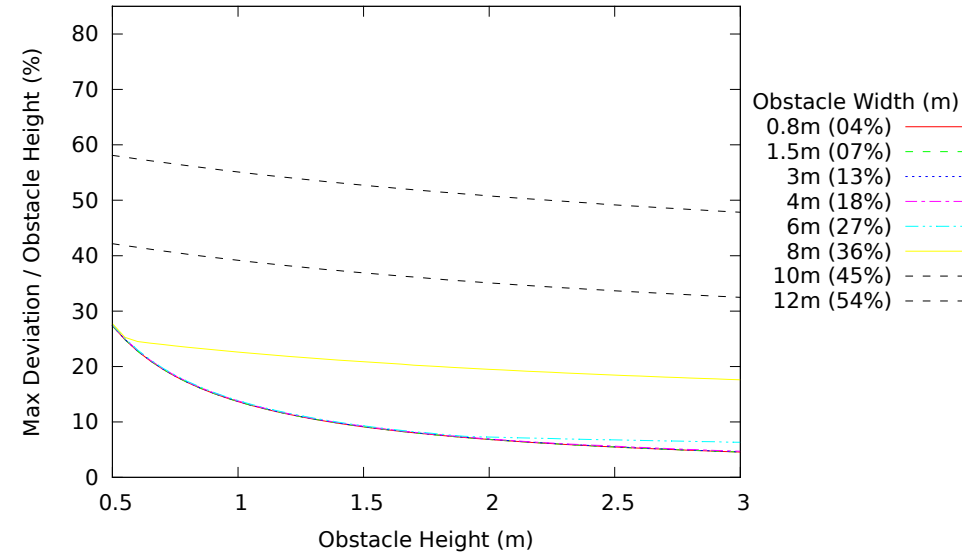
4.3.1 Ground points extraction

The input 3D point cloud includes everything is visible in the images: terrain, obstacles standing on the terrain, sky, etc. The goal of this algorithm, as detailed in Section 3.1, is to fit the terrain surface, removing all the remaining points (outliers). An easy scenario is when all the 3D points belong to the terrain (e.g. a flat road free of obstacles): in this case it is enough to fit the whole 3D point cloud into a simple spline. Now let us to assume the same scene, but with a pedestrian on the ground: a well-performing terrain estimation algorithm should detect the very same terrain surface as before, *without being affected by the presence of the object*.

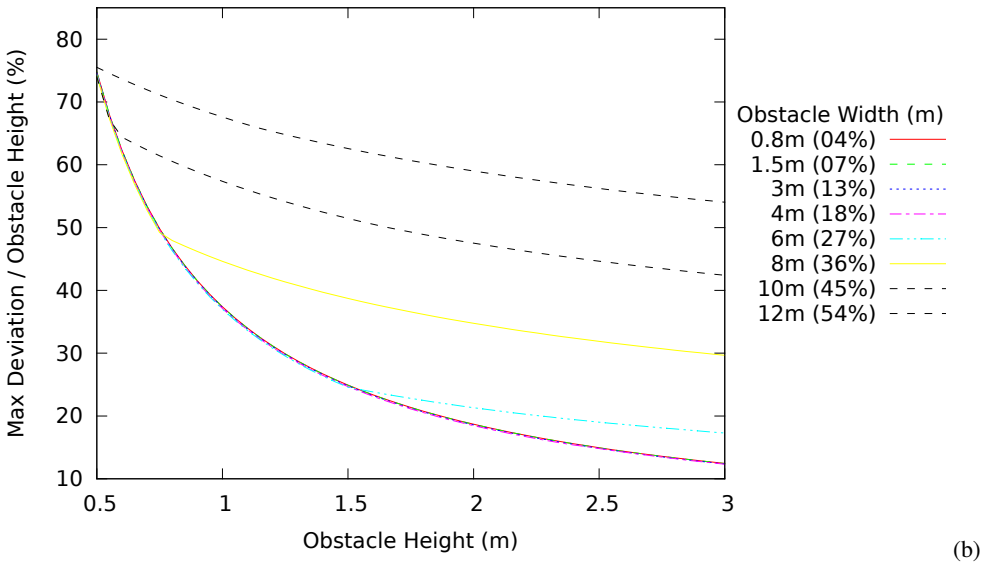
On the basis of this concept, the assessment procedure is the following:

1. select a scene where the terrain is perfectly estimated by the algorithm; e.g. a flat road;
2. add an *artificial obstacle*, of given width and height and estimate again the terrain surface;
3. compare the two surfaces, calculating the maximum deviation from the original surface.

Figure 4.5 shows, for each artificial obstacles' width and against obstacles' height, the maximum surface deviation in percentage of obstacles' height The interest area is $22m \times 22m$, control points are (a) 5×5 and (b) 25×25, splines degree is 3, sampling ratio $50cm$. The obstacle is placed in $(x,y) = (11.0, 0.0)$, with height from 0.5m to 3m and width from 0.8m to 12m.



(a)



(b)

Figure 4.5: Terrain deviation with artificial obstacle.

From this test it is possible to notice that:

- deviations are more dependent on obstacles' size rather than height; in particular they are dependent on size *compared to the area of interest's size*: a tall and thin obstacles (e.g a pole or a pedestrian), will be easily cut off, leading to small differences between the two surfaces; a wide and short obstacle will be less effectively separated from the terrain below;

- under a terrain estimation point of view, the above concept can be rephrased: those parts of the terrain characterized by low height/width ratio turn out in high deviations with respect to a flat plane; i.e. obstacle are more likely to be included into terrain;

- with few control points the surface tents to be rigid, i.e. less affected by the presence of an obstacle, regardless of its height; at the same time, a rigid surface results to be less accurate in estimating rough terrains, since deviations never get close to obstacle height;

- with many control points there is a trade off between terrain fitting and obstacles segmentation: the surface is still able to cut off obstacles, but only when characterized by a minimum steepness; conversely, it is possible to fit terrains with a wider range of slopes.

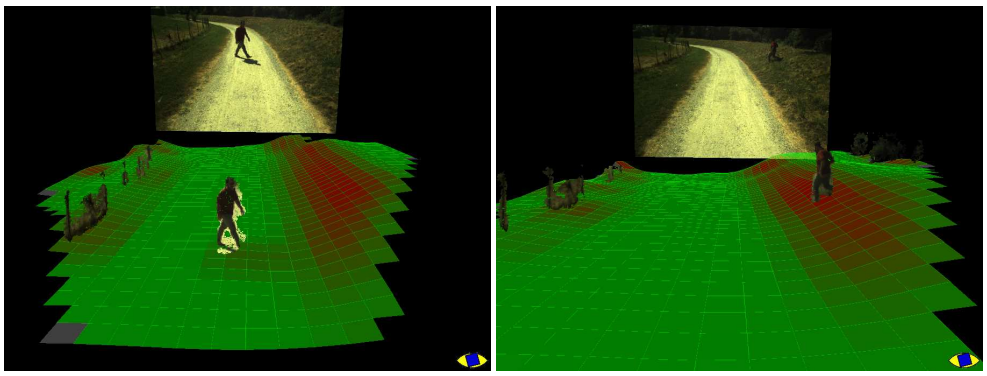
Hence, as mentioned in Section 3.1 and 3.2, it is very important to clearly define what an obstacles is in terms of size and *steepness*. Under a pure geometry based point of view, if the area of interest is limited to $22m \times 22m$, should an obstacle $15m$ wide still be considered an obstacle? Or would it be better to include it into the terrain? Probably the obstacle width is not enough to answer this question, and it is necessary also consider its height: in many extreme scenarios, a $15m$ wide, $1m$ tall area, is not considered an obstacle; but a $15m$ wide, $8m$ tall area deserves more attention, even for a big mining machines.

4.3.2 Results

On the basis of the tests performed in the previous section, the definition of the application scenario in terms of *maximum allowed terrain steepness* and *size limits* of typical obstacles compared with the area of interest is a key aspect. The right spline parameters pattern can be derived with the help of deviation plots like those shown in Figure 4.5.

The system has been tested in different configurations and scenarios (urban, off-road, mining), providing reliable terrain traversability maps. With $34m \times 22m$ area of interest along longitudinal and horizontal directions respectively, 25×25 control points, spline degree 3, and 640×480 pixel images for the DSI, the algorithm runs at 14.70Hz (Preprocessing and DSI=45ms, Terrain Estimation=15ms, Traversability Cost=8ms) on an Intel® Core™i7-3840QM 2.8 GHz with 4GB RAM. For this part, all tests are performed using a PointGrey BBX3-13S2C-38 camera (see Appendix A.1).

Figure 4.6, 4.7, 4.8 and 4.9, show various results in different scenarios, with different lighting conditions and cameras setup in terms of orientation and position.



(a) (b)

Figure 4.6: Examples of terrain estimation with pedestrian in a country road.

Figure 4.6 shows in a country scenarios. In particular, Figure 4.6a and Figure 4.6b show two paradigmatic processing results where a pedestrian walking up and down a

hill. First of all, the terrain estimation is not affected by the presence of a pedestrian, regardless of its position. Remember that this is a geometry based approach, so no analysis of the visual information (color, edge, features, histograms, etc.) is made to classify obstacles. Second, the pedestrian is always detected as an obstacle (outlier), wherever it is placed in the scene. Finally, each portion of the terrain surface is labeled with different traversability cost. The area of interest for these experiments is $22m \times 22m$ for all images.

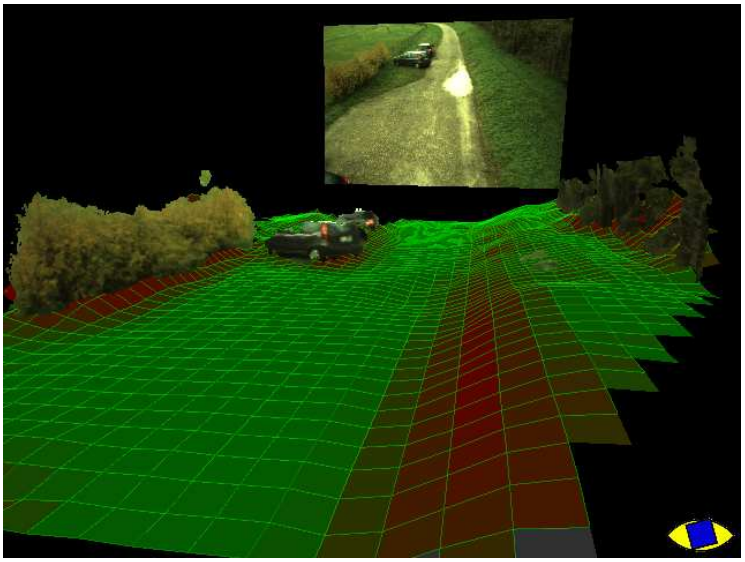
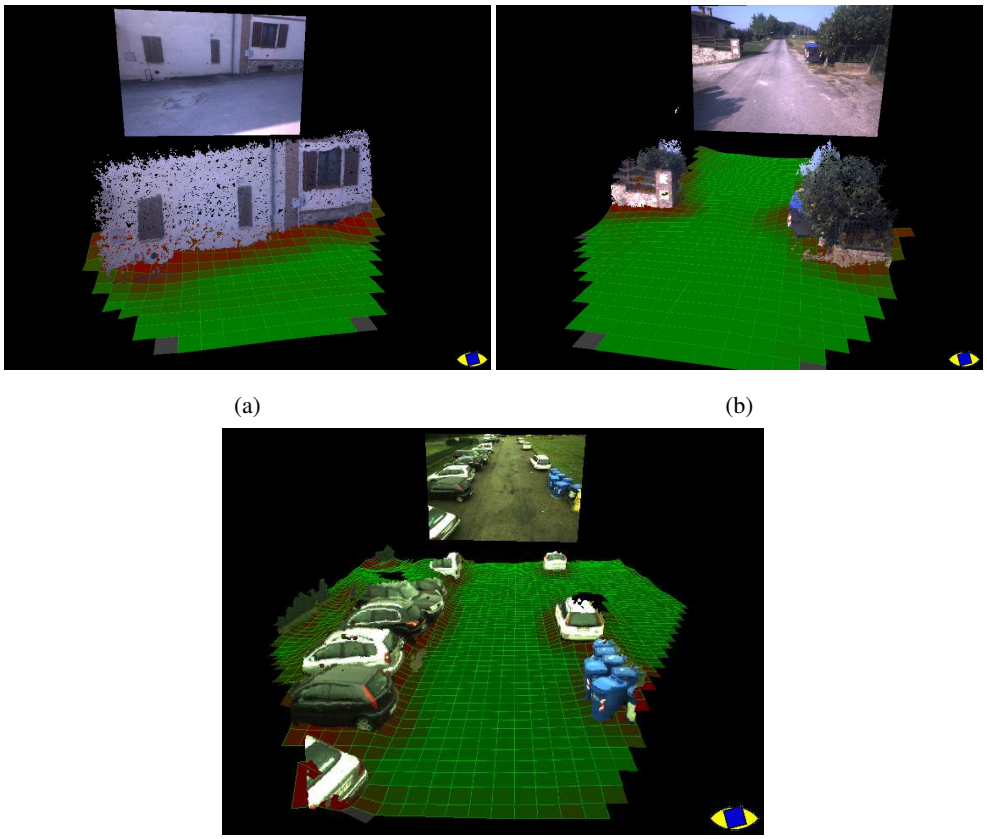


Figure 4.7: Example of terrain estimation in a country road.

Figure 4.7 shows a gravel road with ditch, cars, trees and bushes with an area of interest of $34m \times 22m$.

Figure 4.8 shows the results of the terrain estimation algorithm in little urban roads. The area of interest for these tests is $34m \times 22m$ for all images. Observing Figure 4.8a and Figure 4.8b, it is possible to deduce how little walls with trees and buildings, recognized as outliers, don't interfere with the ground estimation. In Figure 4.8a part of buildings is represented as terrain but is marked as not traversable by the color of the cells below (red). Figure 4.8c shows an experiment in a little park-

ing where all vehicles and obstacles are perfectly marked as outliers using the same stereo camera located at a height of 5m from the ground.

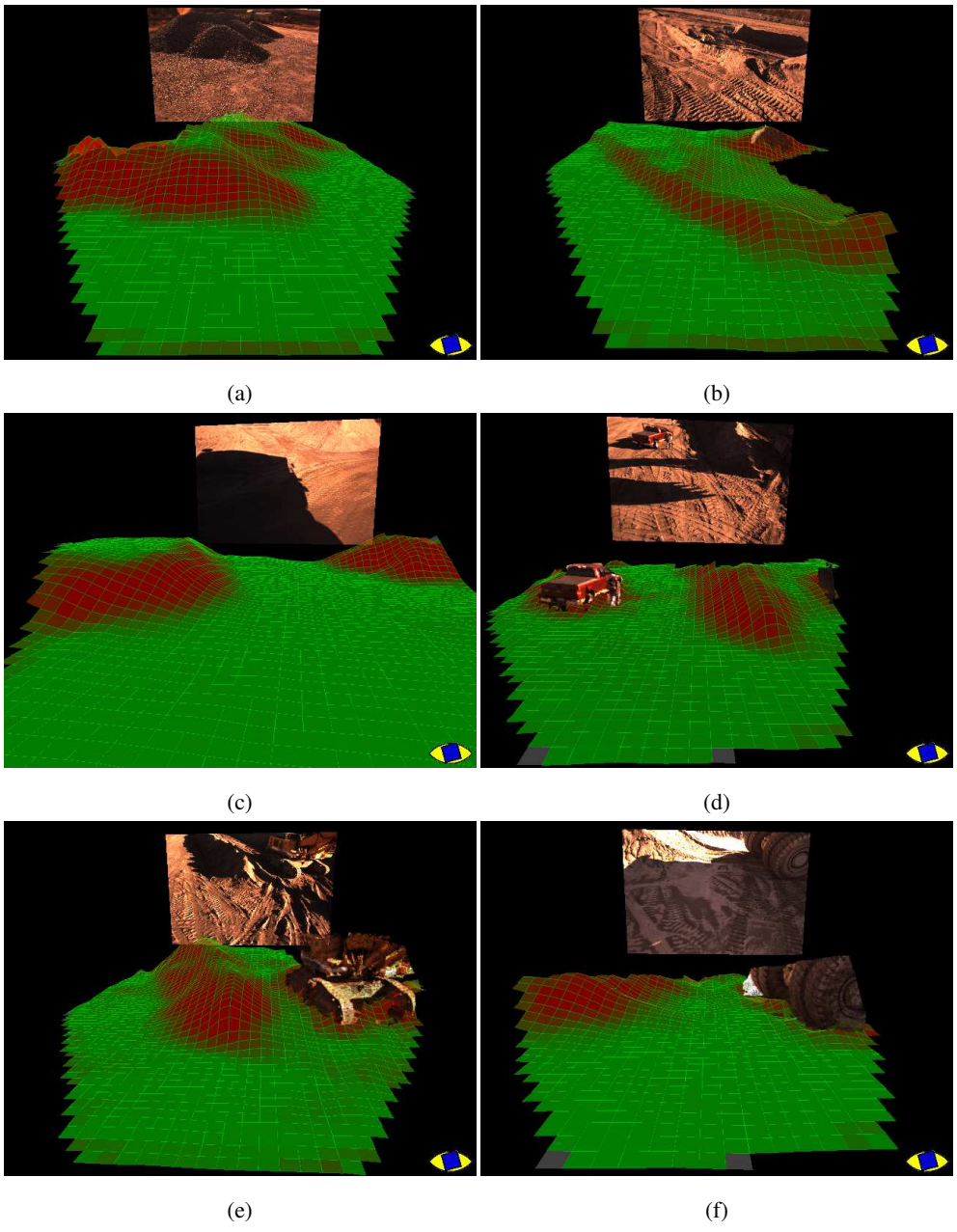


(a) (b)

(c)

Figure 4.8: Output of terrain estimator in a simple urban scenario.

Tests in mining environment are also performed. Mounds of gravel, ditches, berms and gravel "canyons" are marked as part of terrain as respectively shown in Figure 4.9a-c while a car and a pedestrian close to a berm, an excavator close to a berm, a truck close to a high berm are recognized as outliers of terrain as represented in Figure 4.9d-f. The area of interest is $34m \times 22m$ for all images.



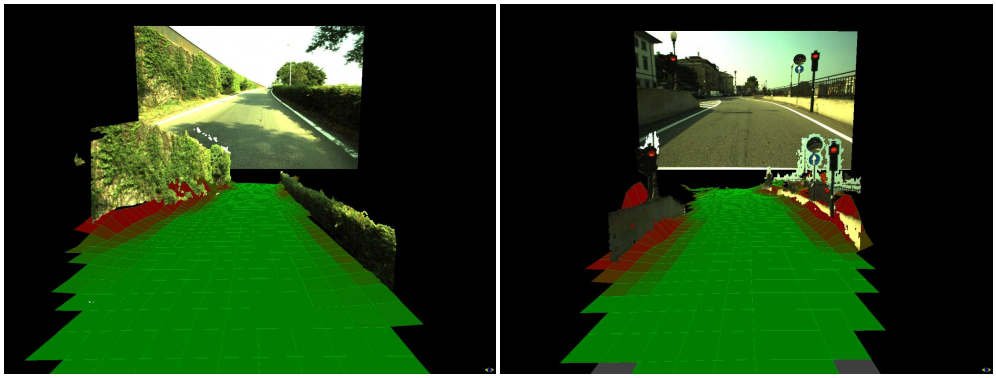
(a) (b)

(c) (d)

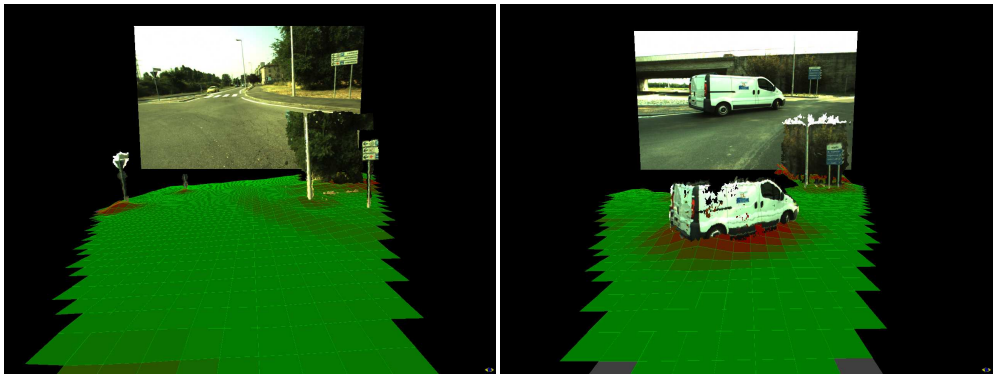
(e) (f)

Figure 4.9: Examples of terrain estimation in mining environment.

Although this system was applied on off-road and country environments, it was also successfully tested in several urban scenarios during the Public ROad Urban Driverless-Car Test (PROUD-Car Test) on 12th July 2013 in Parma - Italy [4] (see Appendix C).



(a) (b)

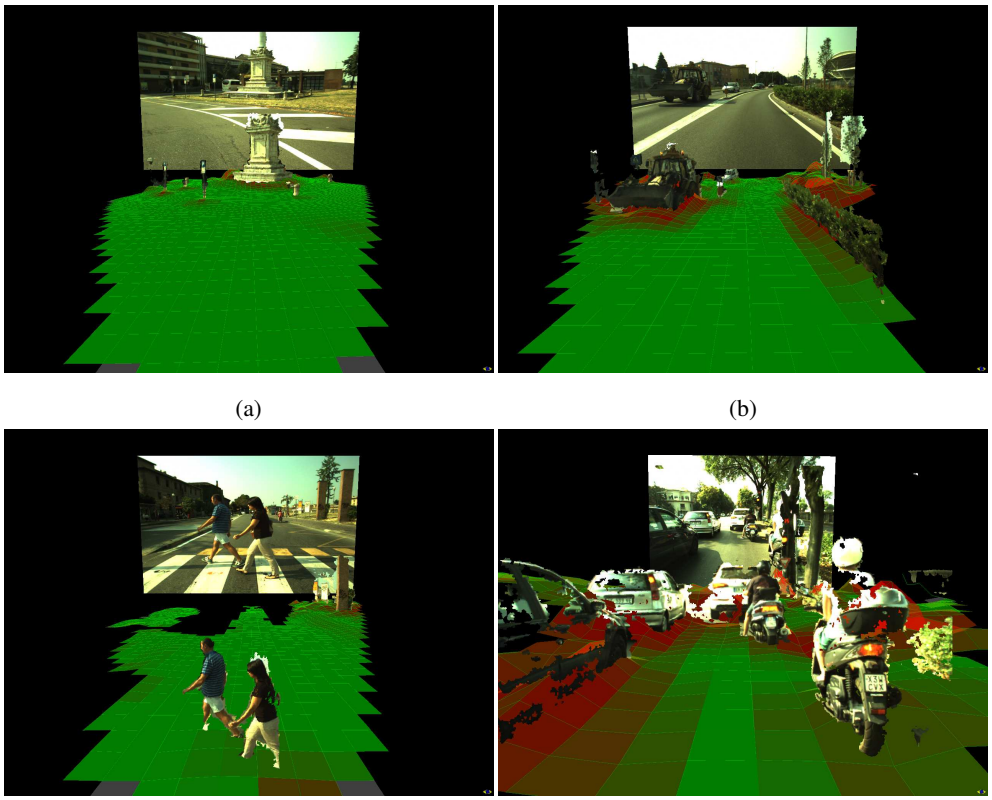


(c) (d)

Figure 4.10: Examples of terrain estimation performed on the PROUD-Car Test data.

This test was performed with the same configuration as the previous ones using an area of interest is $32m \times 14m$ for all images. Figure 4.10 and Figure 4.11 show several examples of terrain estimation processing in various urban scenarios, where having a good terrain estimation significantly helps the obstacle detection phase, es-

pecially in case of non-flat terrain. Figure 4.10a-b show how the proposed algorithm allows for lane reconstruction in presence of strong slopes; traffic lights, walls on side and bushes are still marked as outliers and consequently as obstacle candidates. Figure 4.10c-d show another similar scenario in a roundabout; indeed Figure 4.10c highlights the advantage of this tridimensional algorithm for the terrain estimation against techniques based on ground assumptions or 2-dimensional reconstruction [23, 26] where the roll angle of the stereo camera is not parallel to the ground surface.



(a) (b)

(c) (d)

Figure 4.11: Examples of terrain estimation performed on the PROUD-Car Test data in a dense urban scenario.

The algorithm is also fully parameterizable in terms of obstacles height, terrain slope, terrain resolution, etc., allowing to reliably detect even short obstacles, like bushes and short poles, as show Figure 4.11a-b. In Figure 4.10c-d show respectively example of terrain estimation and outlier extraction in a dense urban scenario: a pedestrian couple while walking on an elevated crosswalk and a noisy estimation of outliers (standing objects) at an intersection. At last in Figure 4.12 a correct detection of a car and guard-rail in a highway is shown.

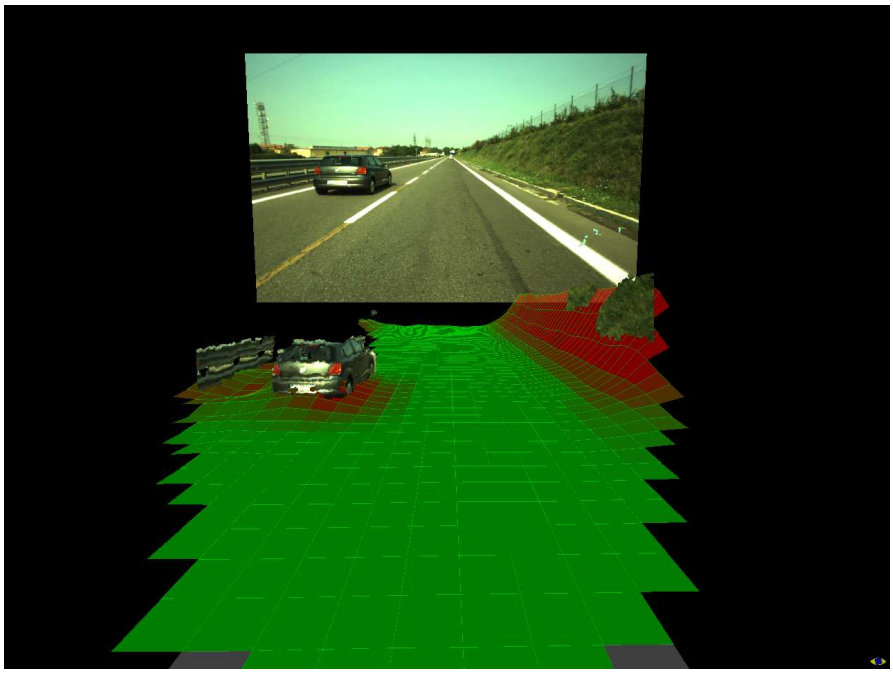


Figure 4.12: Output of the terrain estimation with a car and guard-rail in a highway.

These kinds of scenario are critical for sensors that cannot provide a dense 3D point cloud in a single frame (e.g. 2D LIDAR) that, consequently, typically rely on flat terrain assumption for obstacle detection processing.

4.4 Obstacle detection

Provide a complete system that allows to reconstruct and model the ground shape and the obstacles around the vehicle represents the main goal of this thesis. On the basis of this idea, several obstacle detectors are implemented, as described in Section 3.2, in order to analyze and find the best one. Each proposed strategy depends on the terrain estimation result. Indeed the 3D input data are the discarded 3D points computed by the terrain estimator. The sections below describe the performed tests and the obtained results for each developed algorithm.

4.4.1 Probabilistic occupancy map approaches

As cited in Section 3.2.1.1 and Section 3.2.2, these techniques are strongly depended on the clustering function [29] and the cell size of each cluster. The tests are performed with $34m \times 22m$ area of interest along longitudinal and horizontal directions in corresponding of the vehicle pose as described in Section 4.2.

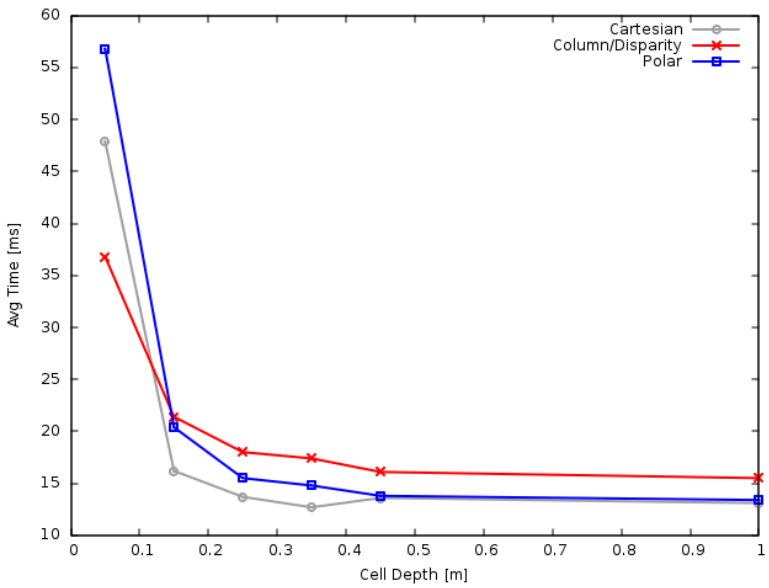
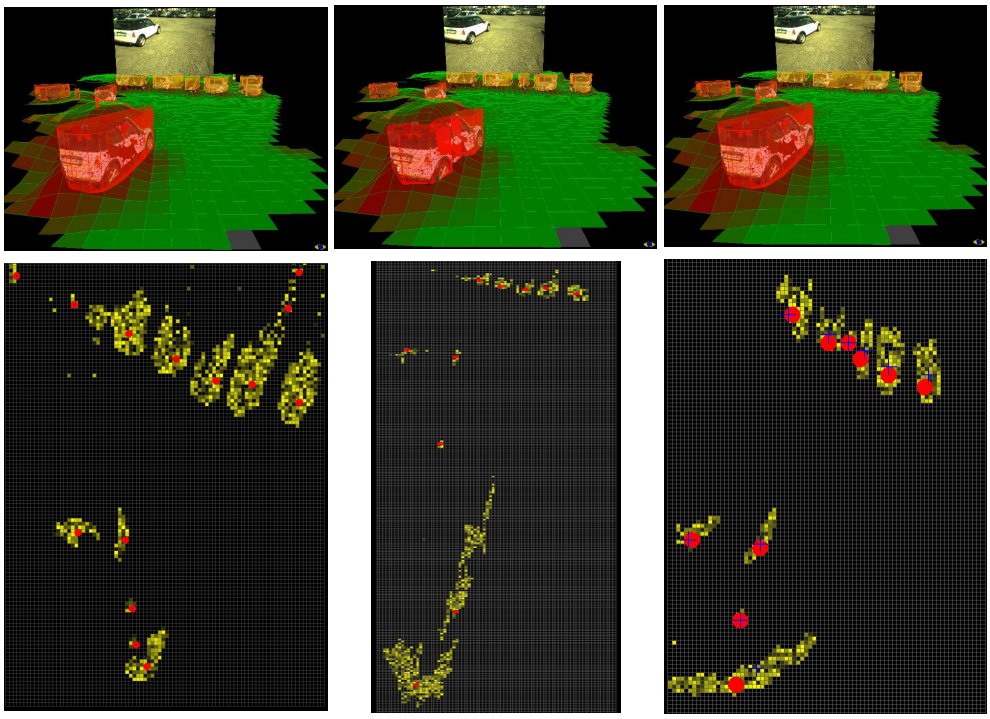


Figure 4.13: Processing times of the occupancy map-based approaches.

Figure 4.13 shows the processing time required by each obstacle detectors for different cell sizes. All tests are also performed with square cells for each applied technique on an Intel® Core™i7-3840QM 2.8 GHz with 4GB RAM.



(a) Cartesian. (b) Column/Disparity. (c) Polar.

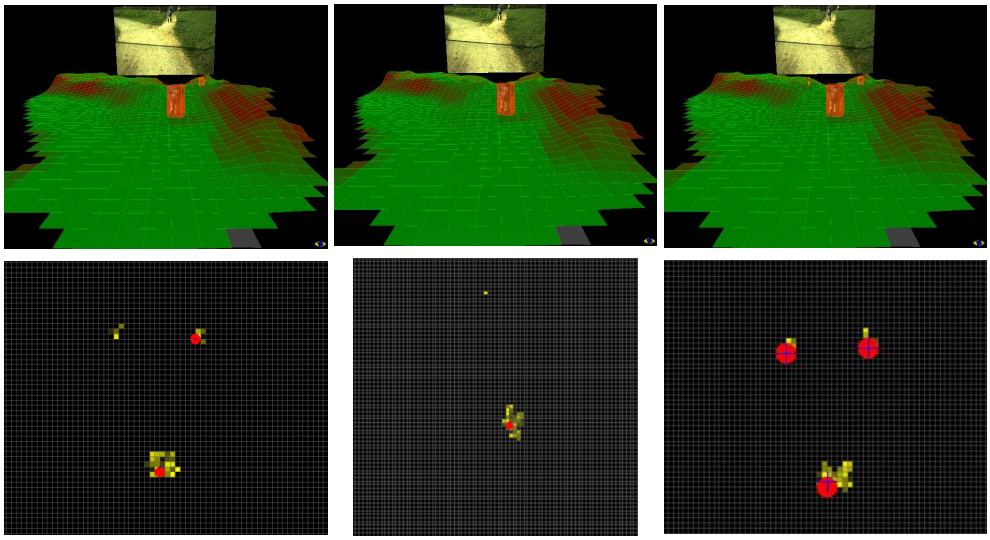
Figure 4.14: Examples of obstacle detection based on probabilistic occupancy maps in a parking.

Figure 4.14 and Figure 4.15 show the output of these obstacle detectors based on Badino's clustering approach [29] where 3 probabilistic occupancy maps using a fixed cell size of 0.25m are computed: cartesian, column/disparity and polar. For each figures, the images, represented in the row at the bottom side, show how the accumulation function for each occupancy map works, as seen in Figure 3.7. As previously described, the input data for each strategy is represented by the 3D points in world

coordinates marked as obstacle candidates (outliers) by the terrain estimation system.

A first qualitative test is represented in Figure 4.14. In this urban scenario the 3D input point cloud, outlier for the terrain estimator, is composed by stationary vehicles in a parking.

Another test, performed in a country scenario, is shown in Figure 4.15 where three far obstacles must be marked as as they are in order to respect the ground truth: a pedestrian and two little poles.



(a) Cartesian. (b) Column/Disparity. (c) Polar.

Figure 4.15: Examples of obstacle detection based on probabilistic occupancy maps in a country scenario.

From a brief analysis of both figures on the top row, it is possible to note that the column/disparity approach is not able to detect the whole obstacles due by the reprojection model based on the accumulation function, as detailed in Equation 3.15. Indeed studying the images on the bottom side, the deductions made using the images on the top side are proved. As advantage this approach provides a compact clustering for each obstacle in order to avoid the splitting problem where parts of the same object

are separately marked as new one. The cartesian occupancy map criteria contains this kind of the problem. Figure 4.14c and Figure 4.15c show how the approach based on polar occupancy solve the problem of the split obstacles, as represented in Figure 4.14a and Figure 4.15a and also the missing detection due by the reprojection strategy of the column/disparity approach (see Figure 4.14b and Figure 4.15b).

Table 4.1 shows the detection rates of each developed occupancy maps on several environments. It is possible to note that the polar approach guarantees a really reliable detection of the obstacles. Therefore some parts of the obstacles are not completely reconstructed in mining scenario; this problem can be solved using a different cell size dimensions in order to boost up the detection performance.

As cited in Section 3.2.2, these approaches guarantee acceptable detection rates but are not suitable for the detection of the hanging and concave objects.

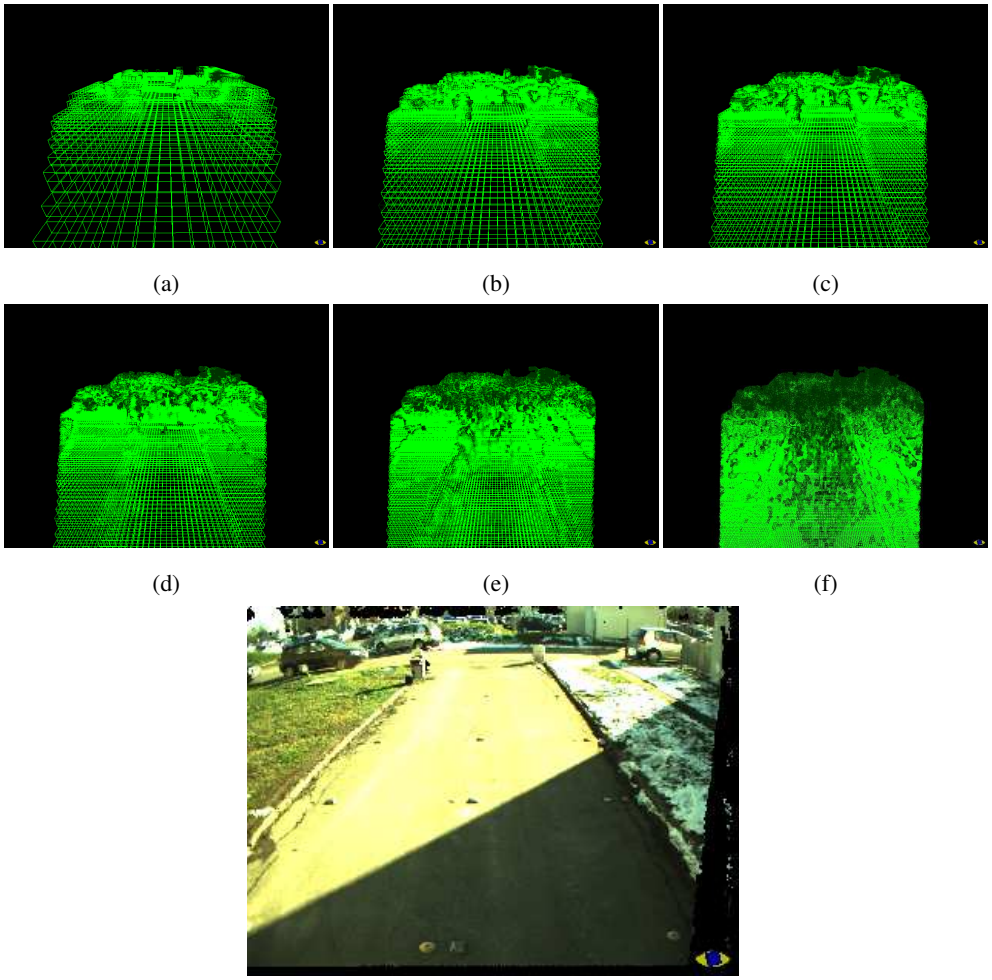
Scenario	*Cartesian %*	*Column/Disparity %*	*Polar %*	#Frames
Urban	90.0	93.2	99.9	4956
Country	89.2	92.5	99.8	3200
Mining	80.0	91.6	99.5	3500

Table 4.1: POM detection rates in several scenarios.

4.4.2 Voxel approach

In Section 3.2.3 the candidate proposes an obstacles detector able to generate a full 3D reconstruction of both stationary and moving objects with minimum assumptions about the road, modelling the 3D point cloud, derived from a disparity image, to an accurate voxel reconstruction hence building complex clusters. These data structures contain the geometric and texture information to perform a segmentation following a flood fill approach. A vehicle pose estimation is carried out to determine the obstacles speed and position by means of an *egomotion* estimation, based on the *visual odometry* approach introduced in [8]. Through a temporal interpolation of previous 3D voxel reconstructions, the objects above the ground can be easily detected and their velocity and position can be estimated using a Kalman Filter.

The input 3D point cloud includes everything that is visible in the images: terrain, obstacles standing on the terrain, sky, etc. The goal of our algorithm is to detect moving obstacles, removing all the remaining points (considered as outliers). The core of algorithm is based on the reconstruction of a voxel map. To figure out the reliability of the developed approach, an evaluation that provides the optimal rendering resolution of voxel partitioning is required. An easy scenario is when all 3D points are processed with no ego-movements (e.g. a flat road in a known position): in this case it is enough to rebuild the whole 3D point cloud into a voxel map at different resolutions. Figure 4.16 shows these experiments. A low resolution indicates a high level detail of the voxel map and vice versa. A high level detail of 3D reconstruction requires a high computational cost in terms of performance; moreover, when the voxel size reduces to the limit where each voxel contains exactly one 3D point, the resulting resolution tends to correspond to the Disparity Map, jeopardizing the advantages of a voxel representation. Hence, it is important to define the right resolution to apply to the clustering algorithm, to achieve the best trade-off between obstacle detection robustness and accuracy in position and speed estimation.



(a) (b) (c)

(d) (e) (f)

(g)

Figure 4.16: Examples of voxel partitioning at resolutions 0.50m(a), 0.25m(b), 0.20m(c), 0.15m(d), 0.10m(e) and 0.05m(f) applied to a static scenario(g).

According to this concept, and the experiments shown in Figure 4.16, when using a volume of interest of $28.7m \times 16.4m \times 3.5m$, it is possible finally to found a satisfactory voxel resolution at $0.25m$, able to provide good processing performance and real-time computation. Alternative configurations provided less accurate results

in terms of detail or several clusters that required a higher computational power to estimate the object's observed shape; in this case many objects are represented in two or more clusters resulting as truncated due to the little variations in clustering criteria values (e.g. unique color-texture, single point in voxel). Figure 4.17 shows the computational costs required to build a complete voxel map.

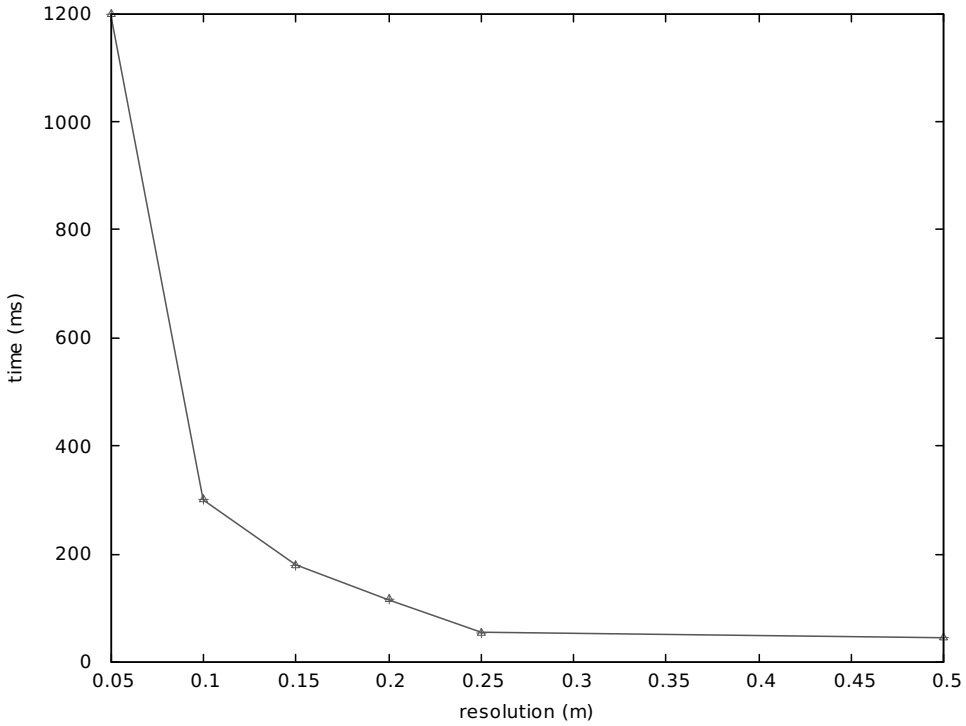
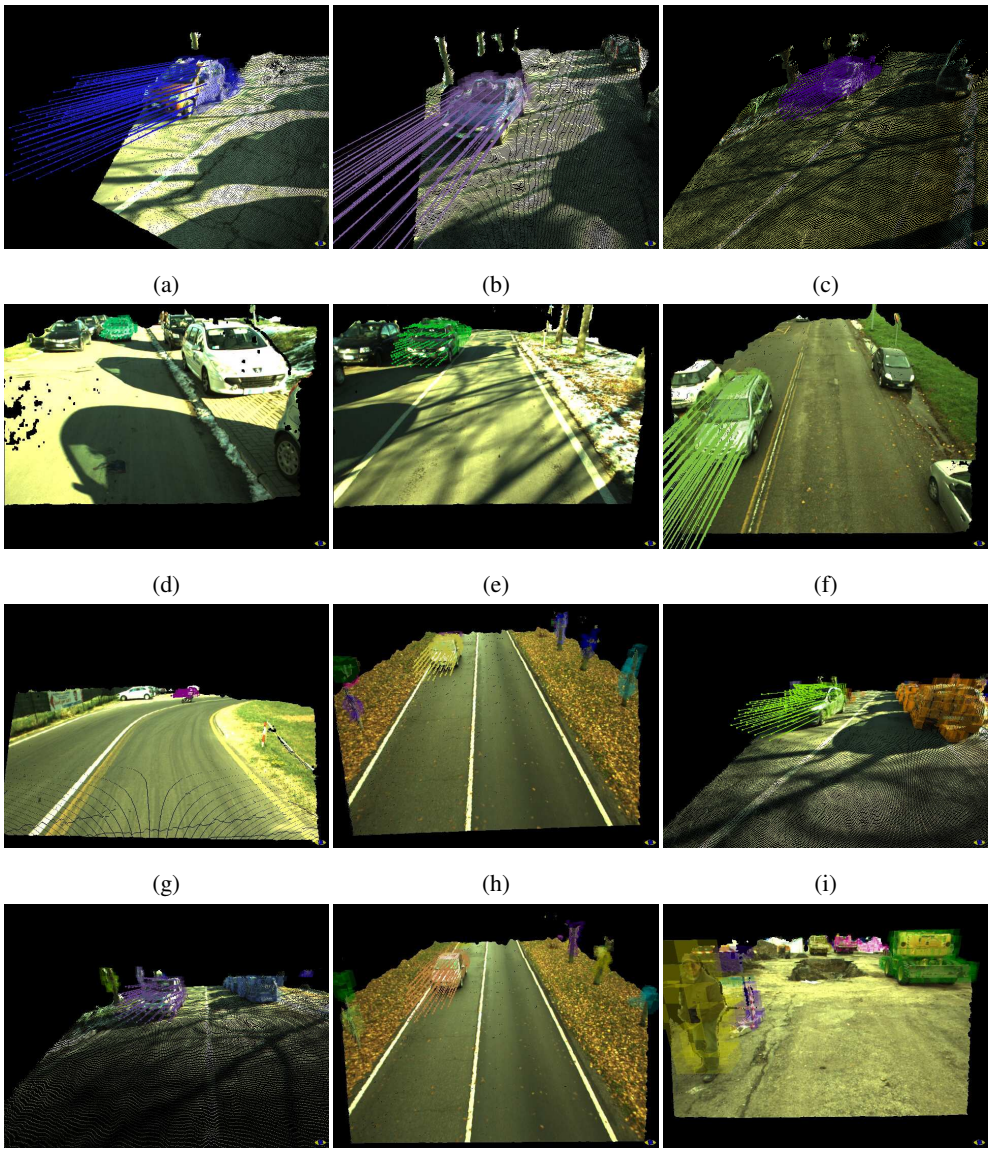


Figure 4.17: Processing times of a voxel map at different resolutions.

This algorithm has been tested for 640×480 pixel images on a Intel® Core™i7-3840QM 2.8 GHz with 4GB RAM at 10Hz (DSI=45ms, Voxel clustering and tracking=55ms). Some qualitative results are illustrated in Figure 4.18.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 4.18: Examples of obstacle detection based on voxel map.

Figure 4.18 shows various results in urban scenarios with different cameras, lighting conditions and system configurations: (a)(b)(c)(d)(e)(f) moving cars without detection of stationary obstacles, (g) moving motorcycle without detection of stationary obstacles, (h)(i)(j)(k) moving cars and stationary obstacles, (l) stationary pedestrian and others obstacles in a construction site. Figure 4.18a to Figure 4.18g show only moving obstacles detection results in different urban scenarios; the pose prediction of each obstacle is shown using colored vectors representing the estimated obstacles' velocity in terms of speed and direction. The next figures show some examples where also stationary obstacles are highlighted. Figure 4.18l shows a complex scenario with various obstacles in a construction site, such as trucks and pedestrians, as stationary objects. In this work everything that stands up from a flat ground is detected as an obstacle and no classification or knowledge assumption is applied to estimate the detected object typology.

Preliminary performances about the moving obstacle detection are shown in Table 4.2. Moving pedestrians and cars have been classified as dynamic. In these tests we have used two annotated image sequences as ground truth with different lighting, cameras and system configurations evaluating the correct (*TP*) and false (*FP*, *FN*) detection rates.

#Sequence	TP	FP	FN	#Frames
Sequence 1	82%	16%	2%	4500
Sequence 2	75%	22%	3%	6200

Table 4.2: Voxel approach results on annoted image sequences

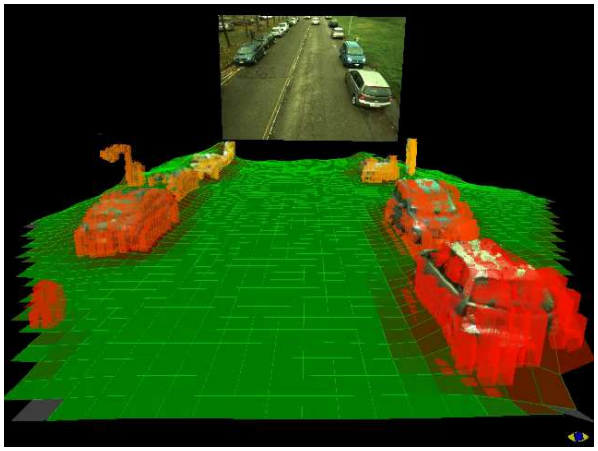
4.4.3 Stixel-based occupancy map approach

As discussed in Section 3.2.4, the occupancy grid maps, implemented in Section 3.2.2, are notably efficient but they are not able to represent concave or floating obstacles if another one is located at the same azimuth and position with different height, as proved in Section 4.4.1. On the contrary, the last approach based on voxel map (see Section 4.4.2) provides a full 3D reconstruction of the obstacles detected but it is really time-consuming. This last strategy, based on a modified version of traditional stixel [34], let to fuse the advantages of polar occupancy map (processing time and reliability) with the full 3D clustering property provided by geometry-based approaches. Indeed, as cited in Section 3.2.4, the key features of this *augmented stixel* are represented by a modified version of traditional structure proposed by Badino et al. [34], as follows:

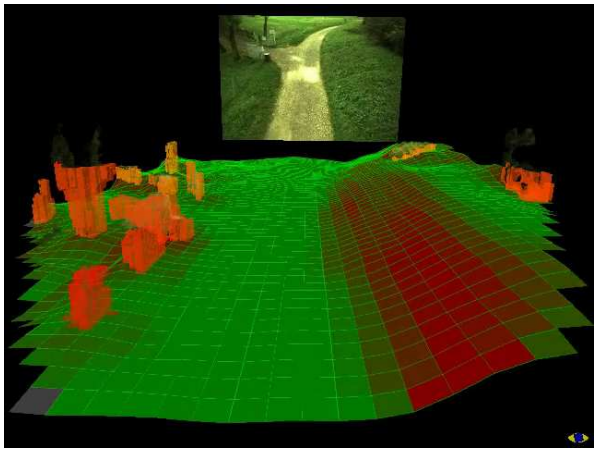
- fixed size for both longitudinal and latitudinal directions;
- textured 3D point cloud;
- estimation of the minimum and maximum height value from the ground.

By this approach it is possible to divide and reconstruct an obstacle, previously clustered with a convex-hull method (see Section 3.2.2), into a set of *augmented stixels* as in the voxel map strategy proposed in Section 3.2.3.

Figure 4.19 shows some examples of this approach in two different scenario: urban and country. In Figure 4.19a and Figure 4.19b it is possible to deduce how each obstacle shape has been modelled by this new *augmented stixel* structure. In both of the images some branches of the trees are represented as well; indeed this proves the capability to reconstruct floating and concave obstacles.



(a)



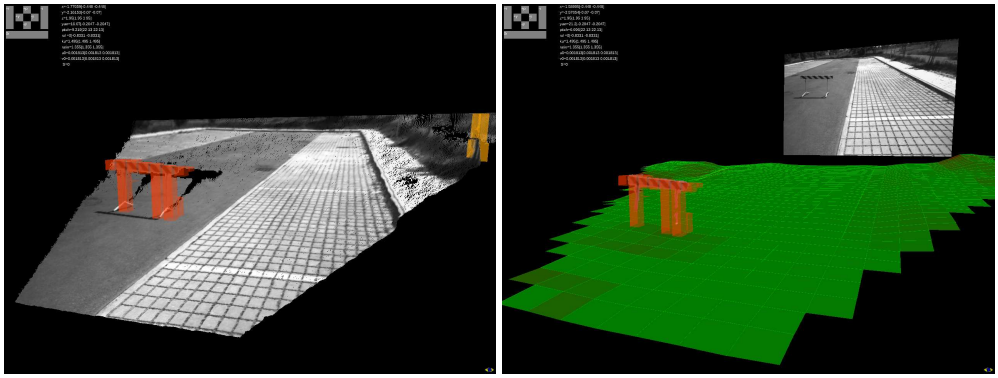
(b)

Figure 4.19: Examples of the final approach based on augmented stixels.

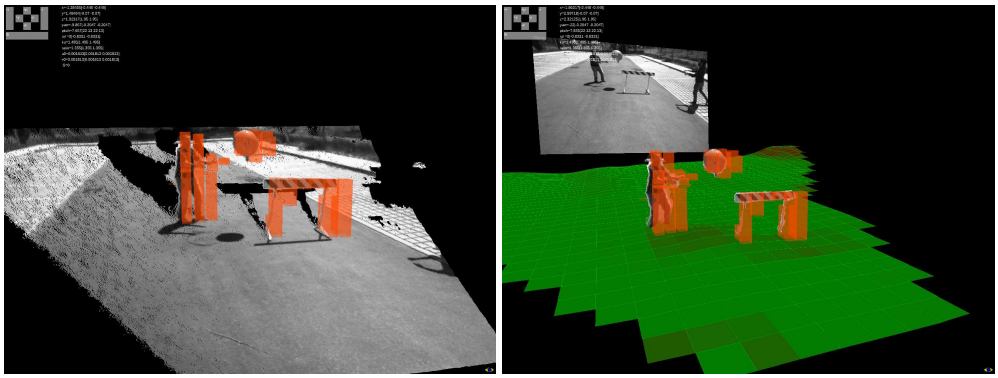
In order to test the reconstruction capability floating and concave obstacles, some supervised tests are performed. Figure 4.20 shows the obtained results. The image on the left side represents the whole 3D point cloud and the obstacles clustered as stixels. On the right side, the complete output of the terrain estimation system combined with this last technique is shown as final solution to the obstacle detection problem for this

work.

In Figure 4.20a an hanging obstacle is perfectly represented by the *augmented stixel* structure. In Figure 4.20b an original output is shown where a floating ball, a board for construction works and a pedestrian have been reconstructed.



(a)



(b)

Figure 4.20: Examples of concave and floating obstacle reconstruction.

This last obstacle detector, combined with terrain estimator, has been tested on an Intel® Core™i7-3840QM 2.8 GHz with 4GB RAM at 12.05Hz (Preprocessing and DSI=45ms, Terrain Estimation=15ms, Obstacle Detection=15ms, Traversability Cost=8ms). All tests are performed using both cameras detailed in Appendix A.

Chapter 5

Conclusions and Future Works

The little that is completed, vanishes from the sight of one
who looks forward to what is still to do.

– Johann Wolfgang von Goethe

5.1 Conclusions

For this thesis a real-time approach for 3D terrain estimation and obstacle detection using on stereo vision has been successfully developed and tested. The terrain has been computed using rational B-Splines surfaces performed by re-weighted iterative least square fitting and equalization. A cloud of 3D points is sampled into a 2.5D grid map; then grid points are iteratively fitted into rational B-Splines surfaces with different patterns of control points and degrees, depending on *traversability* consideration. The obtained surface also represents a segmentation of the initial 3D points into terrain *inliers* and *outliers*.

This strategy presents some advantages, compared with previous techniques:

1. obstacle/terrain segmentation is geometry based only, performed online during the surface fitting process. This allows further analysis of visual data such as

color and texture to reinforce estimates.

2. Terrain estimation (and segmentation) takes into account vehicle capabilities to traverse rough terrain.

3. B-Splines minimize the terrain fitting error over the whole area of interest, providing robustness and computational efficiency; they are also locally controllable through control points, ensuring accuracy in identification of localized terrain independent by the application context.

As contribution to the obstacle detection (OD) problem, a brief survey about the real-time approaches for obstacle detection, mainly based on stereo vision has been presented. A taxonomy of these methods has been postulated in order to highlight the approaches successfully presented in literature in the last years. This research has been focused on the discrimination and selection of the OD techniques that gave a real contribution in terms of reliability, real-time processing capability and robustness. The analyzed approaches have proved effective but also showed some issues. DEM approach [23] and occupancy grid maps [34] are notably efficient but they are not able to represent concave or floating obstacles if another one is located at the same azimuth and position with different height. On the contrary, approaches based on geometry-based clusters, as described in Section 3.2.3, and scene flow segmentation [6], [40] provide a full 3D perception of the obstacles detected but are time-consuming. In order to find a reliable solution to the obstacle detection problem, 3 approaches has been developed, as detailed in Section 3.2:

1. *probabilistic occupancy map approaches*

2. *voxel approach*

3. *stixel-based occupancy map approach*

By these results obtained in Section 4.4, it has been possible to deduce that the last proposed strategy, based on polar occupancy map with *augmented stixels*, allows to reconstruct obstacles in most of the challenging scenarios. In terms of the obstacle

motion estimation each analyzed and developed algorithm provides a reliable assessment through a generic linear Kalman Filter.

The whole algorithm is fully parameterizable in terms of obstacles height, terrain slope, terrain resolution, etc., allowing to reliably detect even short obstacles, like bushes and short poles. Another important advantage of this strategy that it allows to detect floating and concave obstacles as well.

As final contribution, in this work the candidate proposes an innovative real-time stereo vision system for intelligent/autonomous ground vehicles able to provide a full and reliable 3D reconstruction of the terrain and the obstacles.

The system is tested on different platforms providing the following performances:

- Intel® Core™i7-3840QM 2.8 GHz with 4GB RAM at 12.05Hz.
- Intel® Core™i7-820QM 1.73 GHz with 8GB RAM at 10Hz.

Table 5.1 show the processing times of the whole system on the platforms previously described.

Platform	DSI	Terrain Estimation (ms)	Obstacle Detection (ms)	Traversability Cost (ms)
i7-3840QM	45	15	15	8
i7-820QM	55	15	20	10

Table 5.1: Processing times of the terrain and obstacle detection system.

This approach has been also integrated in a commercial stereoscopic system (see Figure 5.1) developed by the VisLab srl: *3DV*[1]. It is composed by a 24cm baseline stereo camera and a ruggedized PC, able to process in real time the whole terrain and obstacle estimation system described in this work.

[1] 3DV website: `http://www.vislab.it/3dv`

Figure 5.1: VisLab stereoscopic system for 3D reconstruction (3DV)

5.2 Future Works

The proposed algorithm is fully frame based, so it does not perform any temporal interpolation. Knowing the vehicle odometry and pose (see Section 3.2.3), it would be possible to integrate in time the surfaces obtained, increasing the robustness. In addition, more constrains on points' heights and derivatives could be added in the spline fitting phase.

The vision based 3D engine implementation can be negatively affected by poor visibility conditions, leading to low disparity map densities. More testing sessions in challenging environments are needed to characterize the system performance with respect to 3D points resolution and density.

A classification of the obstacles, in terms of pedestrians and vehicles, would be considered to improve the obstacle detection and refine the result of the clustering process.

Appendix A

PointGrey BumbleBee XB3

A.1 BBX3-13S2C-38

Resolution	1280 x 960
Frame Rate	16 FPS
Megapixels	1.3 MP
Chroma	Color
Sensor Name	Sony ICX445
Sensor Type	CCD
Readout Method	Global shutter
Sensor Format	1/3"
Pixel Size	3.75 μm
Lens Mount	3 x M12 microlens
Focal Length	3.8 mm, 66-deg HFOV
Aperture	f/2.0
ADC	12-bit
Exposure Range	0.03ms to 66.63ms
Trigger Modes	Standard, bulb, skip frames, overlapped
Flash Memory	512 KB non-volatile memory
Non-isolated I/O Ports	4 bi-directional

Serial Port	2 RS-232 (dedicated pins)
Auxiliary Output	3.3 V, 150 mA maximum
Interface	FireWire 1394b
Power Requirements	12 V
Power Consumption (Maximum)	4W at 12V
Dimensions	277mm x 37mm x 41.8mm
Mass	505 grams
Temperature (Operating)	0 °C to 45 °C
Temperature (Storage)	-30 °C to 60 °C
Humidity (Operating)	20 to 80% (no condensation)
Humidity (Storage)	20 to 95& (no condensation)

Table A.1: Technical specifications of BBX3-13S2C-38.

A.2 BBX3-13S2M-60

Resolution	1280 x 960
Frame Rate	16 FPS
Megapixels	1.3 MP
Chroma	Mono
Sensor Name	Sony ICX445
Sensor Type	CCD
Readout Method	Global shutter
Sensor Format	1/3"
Pixel Size	3.75 μm
Lens Mount	3 x M12 microlens
Focal Length	6mm, 43-deg HFOV
Aperture	f/2.5
ADC	12-bit
Exposure Range	0.03ms to 66.63ms
Trigger Modes	Standard, bulb, skip frames, overlapped

Flash Memory	512 KB non-volatile memory
Non-isolated I/O Ports	4 bi-directional
Serial Port	2 RS-232 (dedicated pins)
Auxiliary Output	3.3 V, 150 mA maximum
Interface	FireWire 1394b
Power Requirements	12 V
Power Consumption (Maximum)	4W at 12V
Dimensions	277mm x 37mm x 41.8mm
Mass	505 grams
Temperature (Operating)	0 °C to 45 °C
Temperature (Storage)	-30 °C to 60 °C
Humidity (Operating)	20 to 80% (no condensation)
Humidity (Storage)	20 to 95& (no condensation)

Table A.2: Technical specifications of BBX3-13S2M-60.

Appendix B

VisLab Intercontinental Autonomous Challenge

B.1 Description

The VisLab Intercontinental Autonomous Challenge (VIAC) [1] is a test of autonomous driving along an unknown route from Italy to China through, Slovenia, Croatia, Serbia, Hungary, Ukraine, Russia, and Kazakhstan the trip began on July 20^{th}, took over three months and the total distance covered was more than 13,000 km. Vehicles travelled from Italy to China through as shown in Table B.1.

The base vehicle is an Electric Porter Piaggio which has been transformed into an Intelligent Vehicle for the challenge, most of the sensing technologies installed on the base vehicle are directly derived from the perception suite of BRAiVE [2]; however, BRAiVE was not designed to drive autonomously in off-road environments, hence it misses all the cross country driving skills needed during an intercontinental trip like VIAC. Vehicles were equipped keeping sensors, actuators, and processing units accessible, in order to optimize development time, usability, and ease maintenance in remote locations.

Throughout the journey the expedition travelled across a plurality of scenarios completely different from each other. Crossing a large part of the Eurasian continent

all sorts of situations, environments, roads, and weather conditions were met: mountains, planes, unpaved, and dusty roads. In Europe and in the first part of the Russian Federation the convoy travelled for many kilometers on highways and drove into the heavy urban traffic of many great eastern Europe cities like Belgrade, Budapest, Kiev, Moscow, then went across Siberia, the flat steppes of Kazakhstan, up the Tien Shan Mountains and finally all the way across China towards its destination, Shanghai. VIAC had his official conclusive event on October the 28^{th} 2010 at the Belgium-EU pavilion inside Shanghai's 2010 World Expo.

The data collected refer to the effective 61 days of autonomous driving on an overall 90 days journey: 191 runs for a total of 214 hours in autonomous mode were completed. Usually the runs ended when no battery power was left, but sometimes logistic needs mandated a stop, such as when crossing a state border. The maximum distance traveled in autonomous mode per run was 96.7 km, with an average of 77.0 km. No autonomous run were performed in some tracks due to technical or logistic problems. The sum of the tracks gives 8244 km in autonomous mode covered at an average speed of 38.4 km/h and a maximum speed of 70.9 km/h. Maximum distance covered in a single day was of 273 km and the maximum amount of time spent in a day driving in autonomous mode was of 6 h, 26 min.

This project was carried out in the frame of the ERC Advanced Investigator Grant (OFAV) received by Prof. Alberto Broggi. Several technical had been taken part to the VIAC expedition : Piaggio, Topcon, Thales, IBM, Enfinity, and Overland Network, including all other partners that worked for the success of this expedition.

Day	Date	Place	Distance [km]	
1	19/07	Milano		
2	20/07	Milano-Parma	153	
3	21/07	Parma-Roma	470	
4	22/07	Roma-Venezia	530	
5	23/07	Venezia-Trieste	57	
6	24/07	Trieste-Lubiana	98	SLOVENIA
7	25/07	Lubiana-Zagabria	153	CROAZIA
8	26/07	Zagabria-Nova Gradiska	200	SERBIA
9	27/07	Nova Gradiska-Belgrado	186	
10	28/07	Belgrado		
11	29/07	Belgrado-Novi Sad	96	
12	30/07	Novi Sad-Subotica	110	
13	31/07	Subotica-Budapest	192	UNGHERIA
14	01/08	Budapest-Zàhony	313	
15	02/08	Zàhoni-L'viv	287	UCRAINA
16	03/08	L'viv-Novohrad Violyns'kyi	312	
17	04/08	Novohrad Violyns'kyi-Kiev	224	
18	05/08	Kiev		
19	06/08	Kiev-Chorol	237	
20	07/08	Chorol-Kharkov	246	
21	08/08	Kharkov		
22	09/08	Kharkov-Slovjansk	186	
23	10/08	Slovjansk-Border	207	
24	11/08	Border-Rostov	100	RUSSIA
25	12/08	Rostov		
26	13/08	Rostov-Millerovo	214	
27	14/08	Millerovo-Voronezh	360	
28	15/08	Voronezh		
29	16/08	Voronezh-Novomoskovk	327	
30	17/08	Novomoskovk-Mosca	242	
31	18/08	Mosca		
32	19/08	Mosca		
33	20/08	Mosca		
34	21/08	Mosca-Vladimir	185	
35	22/08	Vladimir-Niznij Novgorod	236	
36	23/08	Niznij Novgorod		

37	24/08	Niznij Novgorod-Saransk	289	
38	25/08	Saransk-Saratov	361	
39	26/08	Saratov		
40	27/08	Saratov-Sirzan’	311	
41	28/08	Sirzan’-Samara	166	
42	29/08	Samara		
43	30/08	Samara-Dimitrovgrad	145	
44	31/08	Dimitrovgrad-Kazan	234	
45	01/09	Kazan		
46	02/09	Kazan-Naberesnje Celni	241	
47	03/09	Naberesnje Celni-Ufa	287	
48	04/09	Ufa-Yuryuzan’	182	
49	05/09	YuryuzanÃćÂĂÂŹ-Celiabinzk	250	
50	06/09	Celiabinzk		
51	07/09	Celiabinzk		
52	08/09	Celiabinzk-Snezhinsk	110	
53	09/09	Snezhinsk-Jekaterinburg	92	
54	10/09	Jekaterinburg		
55	11/09	Jekaterinburg-Kamysiov	149	
56	12/09	Kamysiov-Tjumen	189	
57	13/09	Tjumen		
58	14/09	Tjumen-Vagay	146	
59	15/09	Vagay-Ishim	163	
60	16/09	Ishim-Omsk	319	
61	17/09	Omsk		
62	18/09	Omsk-Tatarsk	182	
63	19/09	Tatarsk-Kujbysev	255	
64	20/09	Kujbysev-Novosibirsk	330	
65	21/09	Novosibirsk		
66	22/09	Novosibirsk-Bolotnoe	136	
67	23/09	Bolotnoe-Kemerovo	130	
68	24/09	Kemerovo		
69	25/09	Kemerovo-Novokuznetsk	207	
70	26/09	Novokuznetsk-Barnaul	238	
71	27/09	Barnaul-Rubcovsk	315	
72	28/09	Rubcovsk-Semipalatinsk	150	KAZAKISTAN
73	29/09	Semipalatinsk-Georgiyevka	162	
74	30/09	Georgiyevka-Ayagoz	192	

75	01/10	Ayagoz-Usharal	250	
76	02/10	Usharal-Taldykorgan	270	
77	03/10	Taldykorgan-Almaty	261	
78	04/10	Almaty		
79	05/10	Almaty-Sary Ozek	190	
80	06/10	Sary Ozek-Zharkent	180	
81	07/10	Zharkent-Khorgas	50	
82	08/10	Khorgas: border		CINA
83	09/10	Khorgas-Yining	90	
84	10/10	Yining		
85	11/10	Yining-Qingshuihe-Jinghe	290	
86	12/10	Tuotuoxiang-Hutubi	340	
87	13/10	Hutubi-Shanshan	350	
88	14/10	Shanshan-Hami	340	
89	15/10	Hami-Hongliuyuan	300	
90	16/10	Hongliuyuan-Jiayuguan	320	
91	17/10	Jayuguan-Shandan	300	
92	18/10	Shandan-Yongdeng	335	
93	19/10	Yongdeng-Huining	290	
94	20/10	Huining-Binxian	335	
95	21/10	Binxian-Xian	150	
96	22/10	Xian-Sanmenxia	320	
97	23/10	Sanmenxia-Luohe	320	
98	24/10	Luohe-Xiangcheng	360	
99	25/10	Xiangcheng-Mingguang	300	
100	26/10	Mingguang-Changzhou	300	
101	27/10	Changzhou-Kunshan	260	
102	28/10	Kunshan-Shanghai	90	
103	29/10	Shanghai		
104	30/10	Shanghai		
105	31/10	Shanghai		

Table B.1: Trip schedule involved in VIAC project.

Appendix C

Public ROad Urban Driverless-Car Test

C.1 Description

In this challenge, performed in Parma on 12th July 2013, a vehicle moved autonomously on a mixed traffic route (rural, freeway, and urban) open to public traffic; in the final part, nobody sat on the driver seat. Recently other similar systems have been realized by other researchers and auto makers all over the world, but this has been the first time that nobody was in the driver seat, to underline the programmers' confidence. In *Public ROad Urban Driverless-Car Test* (PROUD-Car Test) the most complex part was the handling of real traffic, both in a highway setting (ring around Parma) and in a urban setting (downtown Parma). Figure C.1 shows the vehicle driving autonomously in downtown Parma. An element that greatly increased complexity is the need to negotiate roundabouts (of different size and shape), underpasses, pedestrian crossings, artificial bumps, traffic lights, since these articulated situations require a deep environmental interpretation by the on-board system.

The possibility to conduct tests in an environment open to public traffic (as opposed to closed test tracks) is of paramount importance for the validation of the final system. The safety of the test vehicle and other road participants was guaranteed by

the possibility of a direct intervention by the passenger thanks to a brake pedal in case of unexpected situations; furthermore people on the following vehicle were able to shut down the autonomous vehicle and block it at any time thanks to a remote control over a radio link.

Figure C.1: BRAiVE: Autonomous driving mode in downtown

The vehicle owns an area map also featuring a subset of ADAS information. Once determined the route, shown in Figure C.2, the vehicle defines its future trajectory considering information such as GPS map, lane markings, possible obstacles on the path where, the planned route from the University Campus (A) to Piazza della Pace (B), included two-way rural roads, two freeways with junctions, and plenty of urban areas such as pedestrian crossings, tunnels, artificial bumps, tight roundabouts, and traffic lights. At the same time it defines its speed profile based on road geometry, obstacle presence, and obviously also respecting speed limits.

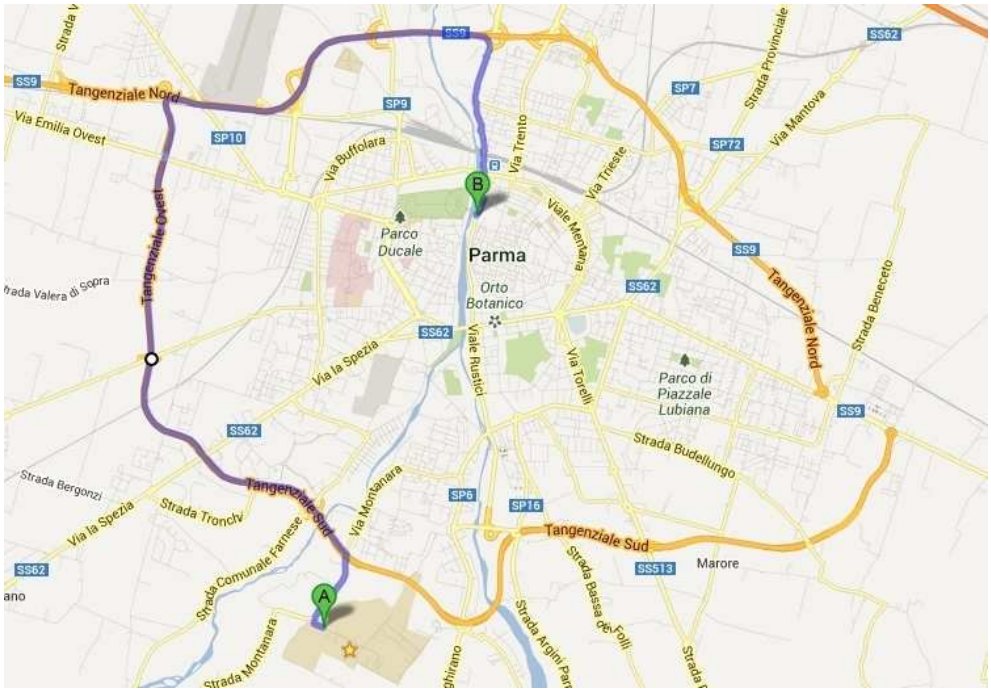


Figure C.2: PROUD route map.

Many different sensors are installed on the vehicle, but not all have been used in this test. The sensors used are: two frontal cameras locate obstacles (pedestrians, bicycles, other vehicles) on the path, locate and interpret traffic lights, determine the position of lane markings, and reconstruct the terrain profile lateral cameras together with lateral laserscanners handle merging and roundabouts a frontal laserscanner together with two lateral laserscanners locate lateral objects (like nearby vehicles, barriers, tunnel sides) two backward looking cameras locate vehicles in adjacent lanes. The sensors installed on the prototype include two different technologies: cameras and lasers, which complement each other in a very straightforward way.

Bibliography

[1] Massimo Bertozzi, Luca Bombini, Alberto Broggi, Michele Buzzoni, Elena Cardarelli, Stefano Cattani, Pietro Cerri, Alessandro Coati, Stefano Debattisti, Andrea Falzoni, Rean Isabella Fedriga, Mirko Felisa, Luca Gatti, Alessandro Giacomazzo, Paolo Grisleri, Maria Chiara Laghi, Luca Mazzei, Paolo Medici, Matteo Panciroli, Pier Paolo Porta, Paolo Zani, and Pietro Versari. VIAC: an Out of Ordinary Experiment. In *Procs. IEEE Intelligent Vehicles Symposium 2011*, Baden Baden, Germany, June 2011.

[2] Luca Bombini, Stefano Cattani, Pietro Cerri, Rean Isabella Fedriga, Mirko Felisa, and Pier Paolo Porta. Test bed for Unified Perception & Decision Architecture. In *Procs. 13th Int. Forum on Advanced Microsystems for Automotive Applications*, pages 287–298, Berlin, Germany, May 2009.

[3] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R.G. Herrtwich. Making bertha see. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 214–221, Dec 2013.

[4] Alberto Broggi, Pietro Cerri, Stefano Debattisti, Maria Chiara Laghi, Paolo Medici, Matteo Panciroli, and Antonio Prioletti. PROUD-public road urban driverless test: Architecture and result. In *Procs. IEEE Intelligent Vehicles Symposium 2014*, pages 648–654, Dearborn (MI), United States, June 2014.

[5] Heiko Hirschmüller. Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In *Intl. Conf. on Computer Vision and Pat-*

tern Recognition, volume 2, pages 807–814, San Diego, CA, USA, June 2005. IEEE Computer Society.

[6] Uwe Franke, Clemens Rabe, Hernán Badino, and Stefan K. Gehrig. 6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception. In *Proceedings of the 27th DAGM Symposium*, pages 216–223, Vienna, Austria, August 2005. Springer.

[7] Mirko Felisa and Paolo Zani. Incremental Disparity Space Image computation for automotive applications. In *Procs. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, St.Louis, Missouri, USA, October 2009.

[8] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, Nov 2000.

[11] Juho Kannala, Janne Heikkila, and Sami S. Brandt. *Geometric Camera Calibration*. John Wiley Sons, Inc., 2007.

[12] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106–1112, Jun 1997.

[13] M.Z. Brown, D. Burschka, and G.D. Hager. Advances in computational stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(8):993–1008, Aug 2003.

[14] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pages 131–140, 2001.

[15] W. van der Mark and D.M. Gavrila. Real-time dense stereo for intelligent vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, 7(1):38–50, March 2006.

[16] Alberto Broggi, Michele Buzzoni, Mirko Felisa, and Paolo Zani. Stereo obstacle detection in challenging environments: the VIAC experience. In *Procs. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 1599–1604, San Francisco, California, USA, September 2011.

[17] F. Malartre, T. Feraud, C. Debain, and R. Chapuis. Digital elevation map estimation by vision-lidar fusion. In *Robotics and Biomimetics (ROBIO), 2009 IEEE Int. Conference on*, pages 523 –528, dec. 2009.

[18] Jens-Steffen Gutmann, Masaki Fukuchi, and Masahiro Fujita. 3d perception and environment map generation for humanoid robot navigation. *Int. J. Rob. Res.*, 27(10):1117–1134, 2008.

[19] H. Moracev. Robot spatial perception by stereoscopic vision and 3D evidence grids. Technical report, Technical Report CMU-RI-TR-96-34, CMU Robotics Institute, 1996.

[20] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010.

[21] Arturo L. Rankin, Andres Huertas, and Larry H. Matthies. Stereo vision based terrain mapping for off-road autonomous navigation. In *Proc. of SPIE, the International Society for Optical Engineering*, volume 7332, 2009.

[22] David Pfeiffer and Uwe Franke. Towards a global optimal multi-layer stixel representation of dense 3d data. In *Proceedings of the British Machine Vision Conference*, pages 51.1–51.12. BMVA Press, 2011.

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

[24] A. Cappalunga, S. Cattani, A. Broggi, M.S. McDaniel, and S. Dutta. Real time 3d terrain elevation mapping using ants optimization algorithm and stereo vision. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 902 –909, june 2010.

[25] Les Piegl and Wayne Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[26] A. Wedel, U. Franke, H. Badino, and D. Cremers. B-spline modeling of road surfaces for freespace estimation. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 828 –833, june 2008.

[27] A. Talukder, R. Manduchi, A. Rankin, and L. Matthies. Fast and reliable obstacle detection and segmentation for cross-country navigation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 610–618 vol.2, June 2002.

[28] Zhongfei Zhang, Richard Weiss, and Allen R. Hanson. Obstacle detection based on qualitative and quantitative 3d reconstruction. *IEEE Trans. on PAMI*, 19:15–26, 1997.

[29] Hernán Badino, Uwe Franke, and Rudolf Mester. Free space computation using stochastic occupancy grids and dynamic. In *Programming, Proc. Int'l Conf. Computer Vision, Workshop Dynamical Vision*, 2007.

[30] A. Discant, A. Rogozan, C. Rusu, and A. Bensrhair. Sensors for obstacle detection - a survey. In *Electronics Technology, 30th International Spring Seminar on*, pages 100–105, May 2007.

[31] Hans P. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical report, 1996.

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

[33] Sebastian Thrun. Learning occupancy grids with forward sensor models. *Autonomous Robots*, 15:111–127, 2002.

[34] Hernán Badino, Uwe Franke, and David Pfeiffer. The stixel world - a compact medium level representation of the 3d-world. In *Proceedings of the 31st DAGM Symposium on Pattern Recognition*, pages 51–60, Berlin, Heidelberg, 2009. Springer-Verlag.

[35] In So Kweon and T. Kanade. High resolution terrain map from multiple sensor data. In *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, pages 127–134 vol.1, Jul 1990.

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

[37] W. Kruger, W. Enkelmann, and S. Rossle. Real-time estimation and tracking of optical flow vectors for obstacle detection. In *Intelligent Vehicles '95 Symposium., Proceedings of the*, pages 304–309, Sep 1995.

[38] C. Braillon, C. Pradalier, J.L. Crowley, and C. Laugier. Real-time moving obstacle detection using optical flow models. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 466–471, 2006.

[39] Andreas Wedel, Annemarie Meißner, Clemens Rabe, Uwe Franke, and Daniel Cremers. Detection and segmentation of independently moving objects from dense scene flow. In Daniel Cremers, Yuri Boykov, Andrew Blake, and FrankR. Schmidt, editors, *Energy Minimization Methods in Computer Vision and Pattern*

Recognition, volume 5681 of *Lecture Notes in Computer Science*, pages 14–27. Springer Berlin Heidelberg, 2009.

[40] Philip Lenz, Julius Ziegler, Andreas Geiger, and Martin Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *Intelligent Vehicles Symposium (IV)*, 2011.

[41] Clemens Rabe, Uwe Franke, and Stefan Gehrig. Fast Detection of Moving Objects in Complex Scenarios. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 398–403, Istanbul, June 2007.

[42] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robot. Automat. Mag.*, 18(4):80–92, 2011.