3D Lidar-based Static and Moving Obstacle Detection in Driving Environments: an approach based on voxels and multi-region ground planes

Alireza Asvadi*, Cristiano Premebida, Paulo Peixoto, Urbano Nunes

Institute of Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra - Polo II, 3030-290 Coimbra, Portugal

Abstract

Artificial perception, in the context of autonomous driving, is the process by which an intelligent system translates sensory data into an effective model of the environment surrounding a vehicle. In this paper, and considering data from a 3D-LIDAR mounted onboard an intelligent vehicle, a 3D perception system based on voxels and planes is proposed for ground modeling and obstacle detection in urban environments. The system, which incorporates time-dependent data, is composed of two main modules: (i) an effective ground surface estimation using a piecewise plane fitting algorithm and RANSAC-method, and (ii) a voxel-grid model for static and moving obstacles detection using discriminative analysis and ego-motion information. This perception system has direct application in safety systems for intelligent vehicles, particularly in collision avoidance and vulnerable road users detection, namely pedestrians and cyclists. Experiments, using point-cloud data from a Velodyne LIDAR and localization data from an Inertial Navigation System were conducted for both a quantitative and a qualitative assessment of the static/moving obstacle detection module and for the surface estimation approach. Reported results, from experiments using the KITTI database, demonstrate the applicability and efficiency of the proposed approach in urban scenarios.

Keywords:
LIDAR perception, Scene understanding, 3D representation, Obstacle detection

1. Introduction

In the last couple of decades, autonomous driving and advanced driver assistance systems (ADAS) have had remarkable progress. Significant scientific advancements in these research topics have benefited from continuous progress on areas such as computer vision, machine learning, control theory, real-time systems and electronics. An intelligent vehicle can be described by the relationship between three commonly accepted modules: perception, planning and control [1]. The perception module, which is of interest here, perceives the environment and builds an internal model of the environment using sensor data. In case of 3D, and for intelligent/autonomous vehicles applications, a perception system perceives and interprets the surrounding environment using, commonly, data from stereo cameras [2], [3] and/or from 3D-LIDARs [4], [5]. Although affordable and having no moving parts, a major limitation of a stereo system is the difficulty in dealing with changes in illumination and weather conditions *e.g.*, snow covered environments, intense lighting conditions and night driving scenarios. Also, stereo cameras are sensitive to calibration errors. On the other hand, LIDAR sensors, such as Velodyne devices, are less sensitive to weather conditions and can work under poor illumination conditions. The main disadvantages of these sensors are their high cost, although this tends to decrease, and they are constituted by moving elements of high precision.

In this work, we consider the 3D measurements coming in the form of a point-cloud from a Velodyne LIDAR mounted on the roof of a vehicle. Given an input point-cloud, it needs to be processed by a perception system in order to obtain a consistent and meaningful representation of the environment surrounding the vehicle. Three main types of data 'representations' are commonly used: 1) Point cloud; 2) Feature-based; and 3) Grid-based. Point cloud-based approaches directly use raw sensor data for environment representation [6]. This approach generates an accurate representation, however, it requires large memory and high computational power. Feature-based methods use locally distinguishable features (e.g. lines [7], surfaces [8], superquadrics [9]) to represent the sensor information. Grid-based methods discretize the space into small grid elements, called cells, where each cell contains information regarding the sensory space it covers. Grid-based solutions are memory- efficient, simple to implement, and have no dependency to predefined features, making them an efficient technique for sensor data representation in intelligent vehicles and robotics.

1.1. Grid-based Representation

Several approaches have been proposed to model sensory data space using grids. Moravec and Elfes [10] presented early works on 2D grid mapping. Hebert et al. [11] proposed a 2.5D grid model (called elevation maps) that stores in each cell the estimated height of objects above the ground level. Pfaff and Burgard [12] proposed an extended elevation map to deal with vertical and overhanging objects. Triebel et al. [13] proposed

*Corresponding author.
Email address: `alireza.asvadi@gmail.com` (Alireza Asvadi)

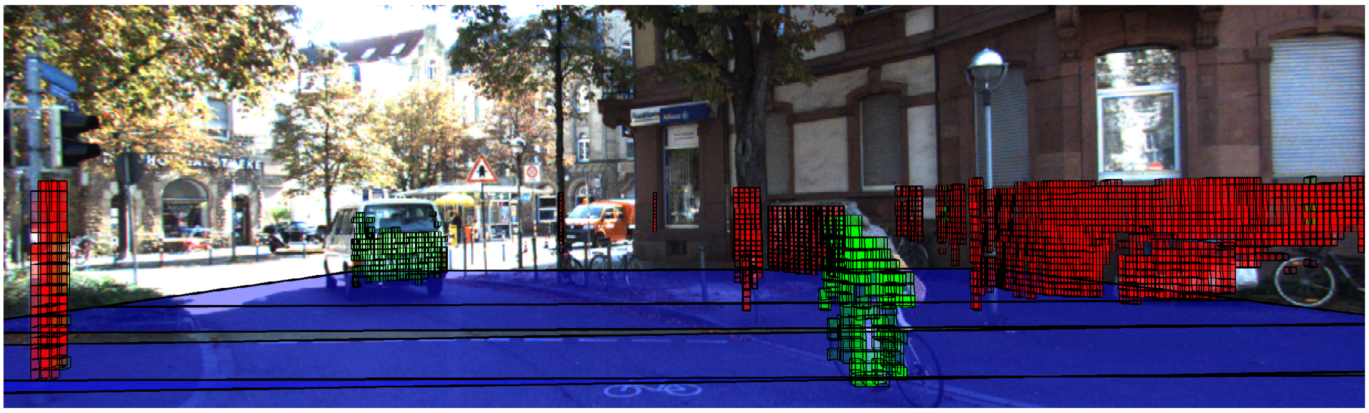

Figure 1: The image shows, for a given frame from the KITTI dataset, the projection of the resulting environment representation where piecewise plane estimation of the ground surface is shown in blue, static obstacles are shown in red, and moving (dynamic) objects are depicted in green.

a "multi-level surface map" that allows multiple levels for each 2D cell. These methods, however, do not represent the environment in a fully volumetric (3D) way. Roth-Tabak and Jain [14] and Moravec [15] proposed a 3D occupancy grid composed of equally-sized cubic volumes (called voxels). However, it requires large amounts of memory since voxels are defined for the whole space, even if there are only a few measured points in the environment. LIDAR-based 3D occupancy grids are able to represent free and unknown areas by accepting a higher computation cost of ray casting algorithms for updating the grid cells. 3D grid maps can be build faster by considering only the end-points of the beams. However, by discarding ray casting algorithms, information about free and unknown spaces is lost. However, this simplified model drastically speeds up the process [16]. A related approach is proposed by Ryde and Hu [17], in which they store a list of occupied voxels over each cell of a 2D grid map. Douillard et al. [18] used a combination of a coarse elevation map for background representation and a fine resolution voxel map for object representation. To reduce memory usage of fully 3D maps, Meagher [19] proposed octrees for 3D mapping. An octree is a hierarchical data structure for spatial subdivision in 3D. OctoMap [20] is a mature version of octree based 3D mapping. However, the tree structure of octrees causes a more complex data access in comparison with a traditional 3D grid. In another attempt, Dryanovski et al. [21] proposed the multi-volume occupancy grid, where observations are grouped together into continuous vertical volumes (height volumes) for each map cell, each volume having a starting position from the ground and a height.

1.2. Grid-based Obstacle Detection

Obstacle detection (OD) which is usually built on top of grid-based representation, is one of the main components of perception in intelligent/autonomous vehicles. It has been in the focus of active research in last years [22]. Recently, most of the OD techniques have been revisited to adapt themselves to 3D sensor technologies [23]. OD algorithms need some assumptions about the ground surface to discriminate between the ground and obstacles [24]. The perception of a 3D dynamic environment by a moving vehicle requires a 3D sensor and an ego-motion estimation mechanism. A perception system with the ability to estimate road surface and obstacle detection in dynamic 3D urban scenario has a direct application in safety systems such as: collision warning, adaptive cruise control, vulnerable road users detection and collision mitigation braking. A higher level perception system would also involve detection and tracking of moving objects (DATMO) [25], object recognition and behavior analysis [26]; neither of these perception modules are considered in this paper.

1.2.1. Ground Surface Estimation

Incoming data from a 3D sensor need firstly to be processed for ground surface estimation and subsequently for obstacle detection. Ground surface and obstacle detection have a strong degree of dependency because the obstacles (*e.g.*, walls, poles, motorcycles, vehicles, pedestrians, and cyclists) are all located on the surface that represents the roadway and the roadside. Many of the methods assume that the ground is flat and everything that stands up from the ground is considered as obstacle [29], [30], [31]. However, this simple assumption is overridden in most of practical scenarios. In [8] the ground surface is detected by fitting a plane using RANSAC on the point-cloud from the current time instance. This method only works well when the ground is planar. Non-planar grounds, such as undulated roads, curved uphill/ downhill ground surface, sloped terrains or situations with big rolling/pitch angle of the host vehicle remain unsolved. The 'V-disparity' approach [32] is widely used to detect the road surface from the disparity map of stereo cameras. However, disparity is not a natural way to represent 3D Euclidean data and it can be sensitive to roll angle changes. A comparison between 'V-disparity' and Euclidean space approaches are given in [33]. In [34] a combination of RANSAC [35], region growing and least square fitting is used for the computation of the quadratic road surface. Though it is effective, yet it is limited to the specific cases of planar or quadratic surfaces. Petrovskaya [36] proposed an approach that determines ground readings by comparing angles between consecutive readings from Velodyne LIDAR scans. Assuming A, B, and C are three consecutive readings, the slope between AB and BC should be near zero if all three points lie on the ground.

Table 1: Some recent related work on 3D perception systems for intelligent vehicles applications.

Reference	3D Sensor	Ego motion estimation	Ground surface estimation	Representation for obstacle detection	Motion detection/ clustering/segmentation	Data association/ tracking
Vatavu *et al.*, 2015 [27]	Stereo camera	GNSS or GPS/IMU odometry	RANSAC, region growing and least square fitting	2.5D digital elevation map (DEM)	Object delimiters extracted from the projection of elevation grids	RaoBlackwellized particle filter
Asvadi *et al.*, 2015 [28]	Velodyne LIDAR	GPS/IMU odometry	Cells contain points with low average and variance in height	2.5D elevation grid	A static local 2.5D map is built and the last generated 2.5D grid is compared with the updated local map	Gating, nearest neighbor association and Kalman filter
Pfeiffer and Franke, 2010 [29]	Stereo camera	Visual odometry	Planar ground assumption	Stixel (2.5D vertical bars in depth image)	Segmentation of stixels based on motion, spatial and shape constraints using graph cut algorithm	6D-vision Kalman filter
Broggi *et al.*, 2013 [30]	Stereo camera	Visual odometry	Planar ground assumption	3D voxel grid	Distinguish stationary/moving objects using ego-motion estimation and color-space segmentation of voxels	A greedy approach based on a distance function and Kalman filter
Azim and Aycard, 2014 [31]	Velodyne LIDAR	GPS/IMU odometry and scan matching	Planar ground assumption	Octomap	Inconsistencies on map and density based spatial clustering	Global Nearest Neighborhood (GNN) and Kalman filter

A similar method was independently developed in [37]. In the grid-based framework presented in [28], grid-cells belonging to the ground are detected by means of the average and variance of the height of points falling into each cell. In [38], all objects of interest are assumed to reside on a common ground plane. The bounding boxes of objects, from the object detection module, are combined with stereo depth measurements for the estimation of the ground plane model.

1.2.2. On-Ground Obstacle Detection

This section briefly reviews OD in a grid-map basis. Vatavu et al. [27] built a rectangular Digital Elevation Map (DEM) with explicit connectivity between adjacent 3D locations using the 3D data inferred from dense stereo. The ground plane projection of this intermediate obstacle representation is used to extract free-form object delimiters. A particle filter-based mechanism is adopted for tracking free-form object delimiters extracted from the grid. In [28], Asvadi et al. used 2.5D elevation grids to represent the environment, having as input data the point-clouds from a Velodyne LIDAR and the measurements from a GPS/IMU localization system, where a local 2.5D static map is built by the combination of 2.5D grids and localization data. Based on a robust spatial reasoning, and comparing the current grid with the updated static map, a 2.5D motion grid is obtained; such motion grids are then grouped to provide an object-level representation of the scene. Finally, a data association strategy in conjunction with Kalman Filter (KF) is used for tracking grouped motion grids. Pfeiffer and Franke [29] used a stereo vision system for acquiring 3D data and visual odometry for ego-motion estimation. They applied the Stixel representation [39], sets of thin and vertically oriented rectangles, for the dynamic environment representation. Stixels are segmented based on the motion, spatial and shape constraints, and are "tracked" by a so-called 6D-vision KF [40] which is a framework for the simultaneous estimation of 3D-position and 3D-motion. Another solution using stereo vision, as a source of 3D data, is addressed by Broggi et al. [30]. Ego-motion is estimated by means of visual odometry and then proceed to distinguish between stationary and moving objects. Using voxel representation, a color-space segmentation is performed on the voxels that are assumed to be above the ground plane, followed by merging (clustering) voxels with similar features. Finally, the geometric center of each cluster is computed and a KF is applied to estimate their velocity and position. Azim and Aycard [31] provide an approach that integrates data from a Velodyne and ego-motion data. Their method is based on the inconsistencies between observation and local grid maps represented by an Octomap [20]. An Octomap is basically a 3D occupancy grid with an octree structure. Potential objects are segmented using a density based spatial clustering. Global Nearest Neighborhood (GNN) data association and KF are used for tracking. An Adaboost classifier is used for object classification. In summary, Table 1 provides an overview of the major environment representation systems proposed in the aforementioned works in terms of the ground surface estimation, obstacle representation and techniques used for estimating the dynamics of such representations.

1.3. Contribution

This paper proposes a complete framework for ground surface estimation and static/moving obstacle detection as illustrated in Fig. 1. While in our previous work [41] the focus was on static/moving obstacle detection, here we extend our framework with two main contributions: (1) a piecewise surface fitting algorithm, based on a 'multi-region' strategy and Velodyne LIDAR scans behavior, applied to estimate a finite set

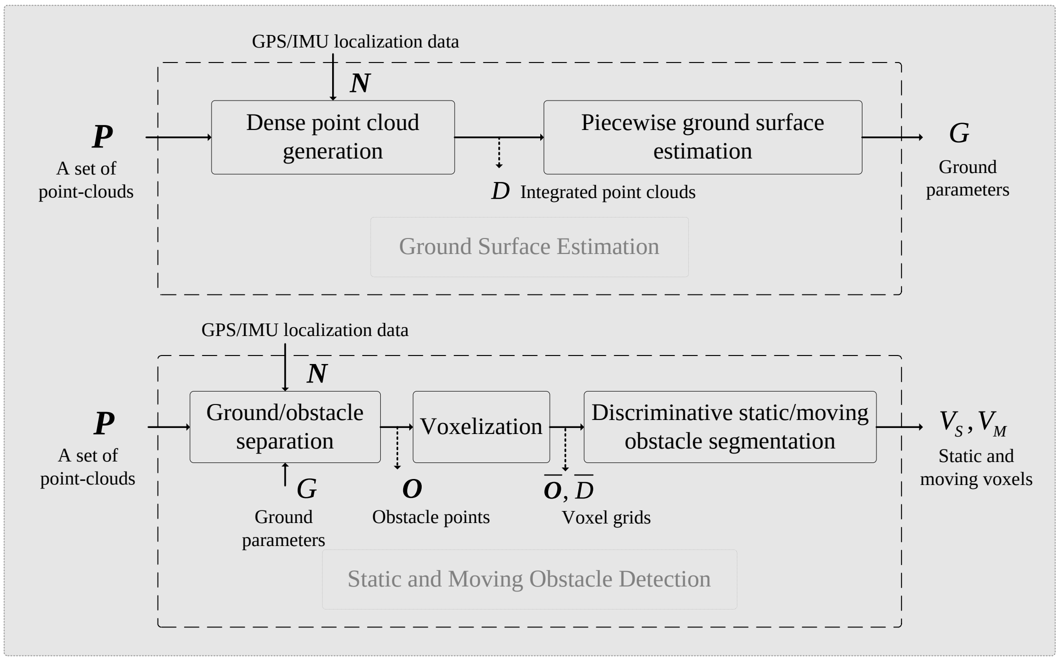
GPS/IMU localization data
N
Dense point cloud generation
Piecewise ground surface estimation
P
A set of point-clouds
G
Ground parameters
D Integrated point clouds
Ground Surface Estimation
GPS/IMU localization data
N
Ground/obstacle separation
Voxelization
Discriminative static/moving obstacle segmentation
P
A set of point-clouds
V_S, V_M
Static and moving voxels
G
Ground parameters
O
Obstacle points
O̅, D̅
Voxel grids
Static and Moving Obstacle Detection

Figure 2: Architecture of the proposed obstacle detection system.

of multiple surfaces that fit the road and its vicinity; (2) a 3D voxel-based representation, using discriminative analysis, for obstacle modeling. The proposed approach deals with non-flat roads, and also detects moving obstacles by integrating and processing information from previous measurements. A set of diversified experiments, and corresponding result analysis, aimed at evaluating the performance of the proposed approach were performed.

The remaining part of this work is organized as follows. Section 2 describes the proposed piecewise plane fitting algorithm and the voxel-based 3D environment modeling. Experimental results are presented in Section 3, and Section 4 brings some concluding remarks.

2. Proposed Obstacle Detection Approach

In this section, we present the proposed environment representation approach to continuously estimate the ground surface and detect stationary and moving obstacles above the ground-level. Fig. 2 presents the architecture of the proposed method. Each block is going to be described in the following sections.

2.1. Dense Point-Cloud Generation

This section starts by presenting the process of dense point-cloud generation, which will be used for the ground surface estimation. The dense point-cloud construction begins by transforming the point-clouds from ego-vehicle to the world coordinate system using GPS/IMU localization data. The result of this transformation is then refined employing point-cloud down-sampling using 'Box Grid Filter', followed by point-clouds alignment using Iterative Closest Point (ICP) algorithm [42]. This process, detailed in the following subsections, is summarized in Algorithm 1.

Algorithm 1 Dense Point Cloud Generation.

```
1: Inputs: Point Clouds: $\mathbf{P}$ and Ego-vehicle Poses: $\mathbf{N}$
2: Output: Dense Point Cloud: $D$
3: for scan k = $i-m$ to $i$ do
4:     $\dot{N}_k \leftarrow$ ICP (BGF (GI $(P_i, N_i)$), BGF (GI $(P_k, N_k)$))
5:     $D \leftarrow$ Merge $(P_k, N_i, \dot{N}_k)$
6: end for
```

2.1.1. GPS/IMU Localization

Let P_i denote a 3D point-cloud in the current time i, and $\mathbf{P} = \{P_{i-m}, \cdots, P_{i-1}, P_i\}$ is a set composed of the current and m previous point-clouds. Using a similar notation, let $\mathbf{N} = \{N_{i-m}, \cdots, N_{i-1}, N_i\}$ be the set of vehicle pose parameters, a 6-DOF pose in Euclidean space, given by a high precision GPS/IMU localization system. $N_k = [R_k \mid T_k]$ consists of a 3×3 rotation matrix R_k and a 3×1 translation vector T_k, when k ranges from $i-m$ to i. GI (P_k, N_k) denotes the transformation of a point-cloud from ego-vehicle to the world coordinate system using: $R_k \times P_k + T_k$.

2.1.2. Point-Cloud Registration

– Pre-processing (down-sampling): A 'Box Grid Filter' is used for down-sampling the point-clouds. It partitions the space into voxels and averages (x, y, z) value of points within each voxel (the voxel size is set as 0.1 m). This step makes the point-cloud registration faster, while keeping accurate results. In Algorithm 1 this step is represented by function BGF.

– ICP Alignment: ICP is applied for minimizing the difference between every point-cloud and the considered reference point-cloud. The down-sampled version of the cur-

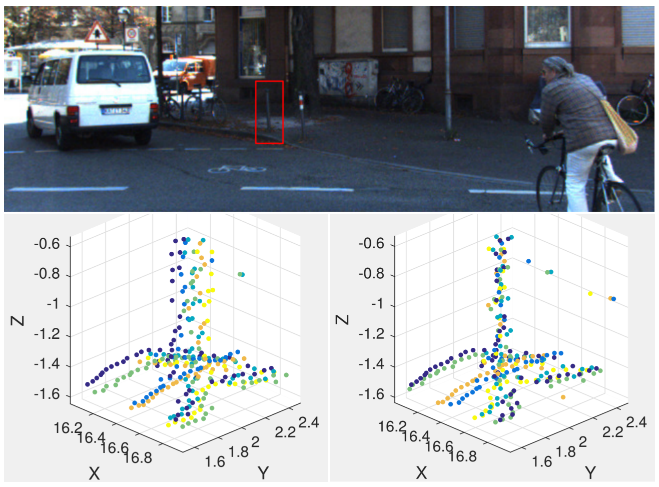

Figure 3: The generated dense point-cloud of a traffic pole before and after applying ‘Box Grid Filter’ and ‘ICP algorithm’. The red rectangle in the upper image shows the above mentioned traffic pole. Bottom left shows the corresponding dense point-cloud, generated only using GPS/IMU localization data. Bottom right shows the result obtained after ‘Box Grid Filter’ and ‘ICP algorithm’ steps to align consecutive point-clouds and reduce the localization error. The sparse points located on the right of the pole correspond to chain that exists between poles. Different colors encode distinct Lidar scans. The dense point-clouds were rotated regarding their original position in the image above to better evidence the difference.

rent point-cloud P_i is used as the reference *‘the fixed point-cloud’* and the 3D rigid transformation for aligning other down-sampled point-clouds P_k *‘moving point-clouds’* with the fixed point-cloud is estimated. Assuming $\dot{N}_k$ as the corrected transformation of GPS/IMU after employing ICP, the so called dense point-cloud (D_i) is obtained using the ‘Merge’ function, by transforming the point-clouds $\mathbf{P}$ into the current coordinates’ system of the ego-vehicle using the parameters of $\dot{N}_k = [\dot{R}_k \mid \dot{T}_k]$ and $N_i = [R_i \mid T_i]$,

$$D_i = \bigcup_{k=i-m}^{i} R_i^{-1} \times ((\dot{R}_k \times P_k + \dot{T}_k) - T_i) \qquad (1)$$

where $\bigcup$ defines the union operation. The integrated point-cloud D is cropped to the local grid: $D \leftarrow Crop(D)$. Note that the subscript i has been omitted to simplify notation. An example of a dense point-cloud generated using GPS/IMU and ICP registration is shown in Fig. 3.

2.2. *Piecewise Ground Surface Estimation*

A piecewise plane fitting algorithm is then applied to D in order to estimate the ground geometry. Existent methods in the literature are mainly developed to estimate specific types of ground surface *e.g.*, planar or quadratic surfaces. In comparison to the previous methods, we contribute with a piecewise plane fitting that is able to estimate an arbitrary ground surface *e.g.*, a ground with a curve profile. The proposed algorithm is composed by four steps: 1) Slicing, 2) Gating, 3) Plane Fitting, and 4) Validation. First, a finite set of regions on the ground are generated in accordance to the car orientation. These regions (hereafter called “slices”) are variable in size and follow the geometrical model that governs the Velodyne LIDAR scans. Second, a gating strategy is applied to the points in each slice using an interquartile range method to reject outliers. Then, a RANSAC algorithm is used to robustly fit a plane to the inlier set of 3D data points in each slice. At last, every plane parameter is checked for acceptance based on a validation process that starts from the closest plane to the farthest plane.

2.2.1. *Slicing*

This process starts from an initial region, defined by the slice S_0, centered in the vehicle and with a radius of $\lambda_0 = 5$ m, as illustrated in Fig. 4. This is the closest region to the host vehicle, with the densest number of points and with less localization errors. It is reasonable to assume that the plane fitted to the points belonging to this region is estimated with more confidence and provides the best fit among all the remaining slices hence, can be considered as a ‘reference plane’ for the validation task. The remaining regions, having increasing size, are obtained by a strategy that takes into account the LIDAR-scans behavior: assumed to follow a tangent function law.

According to the model illustrated in Fig. 4, the area between λ_0 and λ_N is defined by a tangent function (2), where $\alpha_0 = \arctan(\lambda_0/h)$ and h is the height of the Velodyne LIDAR to the ground ($h \approx 1.73$ m, available in the dataset used). Each slice/region begins from the endmost edge of the previous slice in the vehicle movement direction. The edge of the slice S_k is given by,

$$\lambda_k = h \cdot tan(\alpha_0 + k \cdot \eta \cdot \Delta\alpha), \{k : 1, ..., N\} \qquad (2)$$

where N is the total number of slices given by $N = \left\lfloor \frac{\alpha_N - \alpha_0}{\eta \cdot \Delta\alpha} \right\rfloor$. $\lfloor . \rfloor$ denotes truncation operation (the floor function). $\Delta\alpha$ is the angle between scans in elevation direction ($\Delta\alpha \approx 0.4°$). Here, η is a constant that determines the number of $\Delta\alpha$ intervals used to compute each slice. For $\eta = 2$, as represented in Fig. 4, at least two ground readings of a single Velodyne scan fall into each slice which is enough for fitting a plane. Considering $X = (x, y, z)$, $X \in D$, the points with $\lambda_{k-1} < x < \lambda_k$ fall into the k-th slice: $S_k \leftarrow Slice(D)$.

2.2.2. *Gating*

A gating strategy using the interquartile range (IQR) method is applied to S_k to detect and reject outliers that may occur in the LIDAR measurement points. First we compute the median of the height data which divides the samples into two halves. The lower quartile value ($Q_{25\%}$) is the median of the lower half of the data. The upper quartile value ($Q_{75\%}$) is the median of the upper half of the data. The range between the median values is called interquartile range: $IQR = Q_{75\%} - Q_{25\%}$. The lower and upper gate limits are learned empirically, and were chosen as $Q_{min} = Q_{25\%} - 0.5 \cdot (IQR)$ and $Q_{max} = Q_{75\%}$ respectively which is a stricter range when compared to the usual $Q_{25\%} - 1.5 \cdot (IQR)$ and $Q_{75\%} + 1.5 \cdot (IQR)$. The points $X = (x, y, z)$, $X \in D_k$, with $Q_{min} < z < Q_{max}$ are considered as inliers (denoted by $\dot{S}_k$) and are the output of the function: $\dot{S}_k \leftarrow Gate(S_k)$. Please refer to Fig. 5 and Fig. 6 for a better clarification of the proposed method.

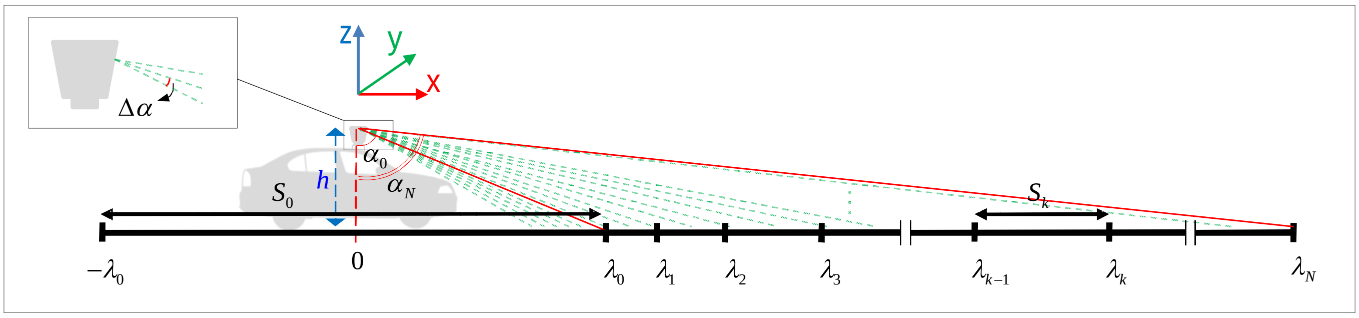

Figure 4: Illustration of the variable-size ground slicing for $\eta = 2$. Velodyne LIDAR scans are shown as dashed green lines.

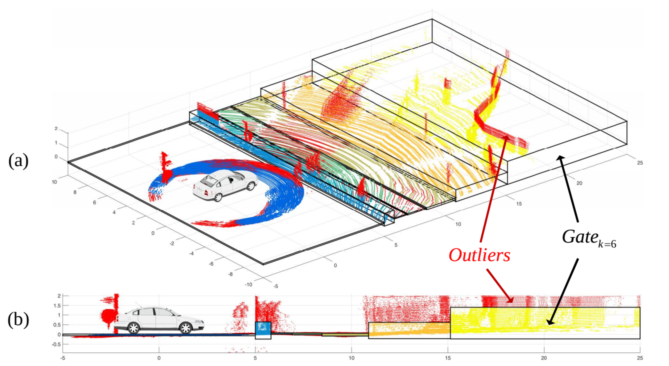

Figure 5: (a) An example of the application of the gating strategy on the dense point-cloud. (b) shows the same scene in a lateral view. 3D black bounding boxes indicate the gates. Inlier points in different gates are shown in different colors, and red points outside the 3D bounding boxes indicate outliers.

2.2.3. *RANSAC Plane Fitting*

The RANSAC method [35] robustly fits a mathematical model to a dataset containing outliers. Differently from the Least Square (LS) method that directly fits a model to the whole dataset (when outliers occur the least square method will not be accurate), RANSAC estimates parameters of a model using different observations from data subsets.

A sub-sample of the filtered point-cloud in each slice $R \subset \dot{S}_k$ is selected and the 3-point RANSAC algorithm is used to fit a plane to it. In each iteration, the RANSAC approach randomly selects three points from the dataset. A plane model is fitted to the three points and a score is computed. The score is computed as the number of inliers points whose distance to the plane model is below than a given threshold. The plane having the largest number of inliers is chosen as the best fit to the considered data.

A given plane, fitted to the road and its vicinity pavement, which are sometimes two planes with few centimeters of difference, is defined as $a_k x + b_k y + c_k z + d_k = 0$, denoted by $G_k \leftarrow [a_k, b_k, c_k, d_k]$. The process for the piecewise RANSAC plane fitting is illustrated in Fig. 6.

2.2.4. *Validation of Piecewise Planes*

The plane computed from the immediate region of the host vehicle is considered as a 'reference plane' G_0 and is assumed to have the best fit among the other slices. Due to tangent-based slicing, the number of Lidar's ground readings which is considered to compute the other planes G_k is almost equal (see Fig. 4 for the case of $\eta = 2$).

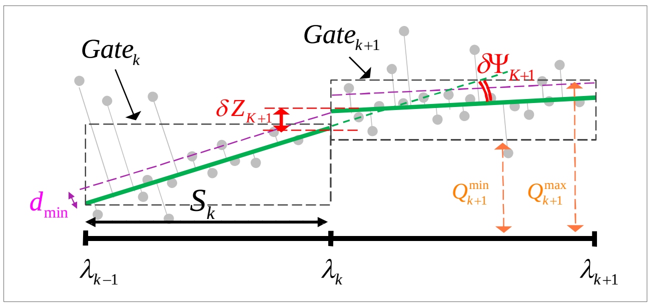

Figure 6: The piecewise RANSAC plane fitting process. Dashed orange shows the lower and upper gate limits. Dashed black rectangles show the gate computed for the outlier rejection task. Solid green lines show the estimated plane using RANSAC in a lateral view. Dashed green line shows the continuation of the S_k plane in slice S_{k+1}. The distance (δZ_{k+1}) and angle ($\delta\psi_{k+1}$) between two consecutive planes are shown in red. Dashed magenta lines show the threshold that is used for the ground/on-ground obstacle separation task. Points under dashed magenta lines are considered as ground points. The original point-cloud is represented using filled gray circles.

The validation process starts from the closest plane G_1 to the farthest plane G_N. For the validation of piecewise planes, two features are considered:

1. The angle between two consecutive planes G_k and G_{k-1} is computed as follows: $\delta\psi_k = \arctan\left|\frac{\hat{n}_{k-1} \times \hat{n}_k}{\hat{n}_{k-1} \cdot \hat{n}_k}\right|$ where $\hat{n}_k$ and $\hat{n}_{k-1}$ are the unit normal vectors of the planes G_k and G_{k-1} respectively.
2. The distance between G_{k-1} and G_k planes computed by $\delta Z_k = |Z_k - Z_{k-1}|$, where Z_k and Z_{k-1} are the z value for G_{k-1} and G_k on the edge of slices: $\binom{x}{y} = \binom{\lambda_k}{0}$. The z value for G_k can be computed by reformulating the plane equation as: $z = -(a_k/c_k)x - (b_k/c_k)y - (d_k/c_k)$.

If the angle between the two normals $\delta\psi_k$ is less than τ° and the distance between planes δZ_k is less than ℓ (τ° and ℓ are given thresholds), the current plane is assumed valid. Otherwise, the parameters from the previous plane G_{k-1} are propagated to the current plane G_k and the two slices are considered to be part of the same ground plane: $G_k \leftarrow G_{k-1}$. This procedure is summarized in Algorithm 2. The output of this algorithm is the ground model defined by the set $G = \{G_1, \cdots, G_N\}$.

Algorithm 2 Piecewise Ground Surface Estimation.

```
1: Input: Dense Point Cloud: $D$
2: Output: Ground Model: $G = \{G_1, \cdots, G_N\}$
3: for slice k = 1 to $N$ do
4:     $S_k \leftarrow$ Slice ($D$)
5:     $\dot{S}_k \leftarrow$ Gate ($S_k$)
6:     $G_k \leftarrow$ RANSAC ($\dot{S}_k$)
7: end for
8: for slice k = 1 to $N$ do
9:     if $\neg((\delta\psi_k < \tau^\circ) \wedge (\delta Z_k < \ell))$ then
10:        $G_k \leftarrow G_{k-1}$
11:    end if
12: end for
```

2.3. Ground/On-Ground Obstacle Separation

The multi-region ground model G is used for the ground/obstacle separation task. It is performed based on the distance between the points inside each slice region S_k to the corresponding surface plane G_k. An arbitrary point p_0 inside the surface plane G_k is selected (*e.g.*, $p_0 = [0, 0, -\frac{d_k}{c_k}]$). The distance from a point $p \in S_k$ to the plane G_k is given by the dot product: $d = (\overrightarrow{p - p_0}) \cdot \hat{n}_k$, where $\hat{n}_k$ is the unit normal vector of G_k plane. The points under a certain reference height d_{min} are considered as a part of the ground plane and are removed (see Fig. 6). The remaining points represent obstacles' points. This process is applied on the last m previous scans: $\mathbf{O} \leftarrow Obstacle(\mathbf{P})$.

2.4. Voxelization

Urban scenarios, especially those in downtown areas, are complex 3D environments, with a great diversity of objects and obstacles. Voxel grids are dense 3D structures with no dependency to predefined features which allow them to provide detailed representation of such complex environments. The voxelization process is performed using essentially two main steps:

1. Quantizing end-points of the beams: Considering the obstacle points set $\mathbf{O} = \{O_{i-m}, \cdots, O_{i-1}, O_i\}$ obtained from the previous module, the quantization of $X = (x, y, z)$, $X \in O_k$ is attained by $\check{X} = \lfloor X \,/\, \upsilon \rfloor \times \upsilon$, where $\lfloor . \rfloor$ denotes the floor function, and υ is the voxel size, here chosen to be equal to 0.1 m. This process converts the original values in $\mathbf{O}$ to the quantized set $\check{\mathbf{O}}$.
2. Computing the occupancy values: The repeated elements in $\check{O}_k$ denote points within the same voxel. The occupancy value of a voxel is determined by counting the number of points in $\check{O}_k$ that have the same value. The output of this task is a list of voxels with the occupancy values of $N(\check{O}_k = U)$, $\forall \check{X} \in U$. $\check{X} = (\check{x}, \check{y}, \check{z})$, $\check{X} \in U$, and $U = unique(\check{O}_k)$.

Voxelization is applied to the obstacle points set, $\overline{\mathbf{O}} \leftarrow Voxelization(\mathbf{O})$, and to the dense point-cloud (after ground removal), $\overline{D} \leftarrow Voxelization(D)$.

2.5. Discriminative Stationary/Moving Obstacle Segmentation

The obstacle voxel grids $\overline{\mathbf{O}} = \{\overline{O}_{i-m}, \cdots, \overline{O}_{i-1}, \overline{O}_i\}$ and the integrated voxel grid $\overline{D}$ are used for the stationary/moving obstacle segmentation. The main idea is that a moving object occupies different voxels along time while a stationary object will be mapped into the same voxels in consecutive scans. Therefore, the occupancy value in voxels corresponding to static parts of the environment is greater in $\overline{D}$. To materialize this concept, first a rough approximation of stationary and moving voxels is obtained by using a simple subtraction mechanism. Next, the results are further refined using a discriminative analysis based on *2D Counters* built in the $X - Y$ plane. The Log-Likelihood Ratio (LLR) of the *2D Counters* is computed to determine the binary masks for the stationary and moving voxels.

2.5.1. Pre-processing

A subtraction mechanism is used as a pre-processing step. The cells belonging to the static obstacles in $\overline{D}$ capture more amount of data and therefore have a greater occupancy value in comparison with each of the obstacle voxel grids in $\overline{\mathbf{O}}$ (see Fig. 7-a). On the other hand, since moving obstacles occupy different voxels in the grid, it may be possible that for those voxels some elements of $\overline{D}$ and $\overline{O}_k$ will have the same occupancy values. Having this in mind, $\overline{D}$ is initialized as the stationary model. The voxels in $\overline{D}$ are then compared with the corresponding voxels in each of the obstacle voxel grids $\overline{O}_k \in \overline{\mathbf{O}}$. Those voxels in $\overline{D}$ that have the same value as corresponding voxels of $\overline{O}_k$ are considered as moving voxels and filtered out. Next, the filtered $\overline{D}$ is used to remove stationary voxels from the current obstacle voxel grid $\overline{O}_i$. Filtered $\overline{D}$ and $\overline{O}_i$ are outputted. To keep the notation simple, we keep the variable names the same as before pre-processing and dismiss the subscript of $\overline{O}_i$.

2.5.2. 2D Counters

We assume that all voxels in the same $X - Y$ cell of the ground-surface plane have the same state (stationary or moving). Based on this assumption, 2D counters are built in the $X - Y$ plane for both $\overline{D}$ and $\overline{O}$ (see Fig. 7-c and -d). A given cell $(\check{x}, \check{y})$, which can assume the stationary or the moving state, is subjected to a summation operation (our 2D counter) as expressed by:

$$H_s(\check{x}, \check{y}) = \sum_{k=1}^{n(\check{x}, \check{y})} \overline{D}(\check{x}, \check{y}, \check{z}) \tag{3}$$

$$H_d(\check{x}, \check{y}) = \sum_{k=1}^{m(\check{x}, \check{y})} \overline{O}(\check{x}, \check{y}, \check{z}) \tag{4}$$

where $(\check{x}, \check{y}, \check{z})$ is the location of a cell in a voxel grid. H_s and H_d are the computed static and dynamic counters, $n(\check{x}, \check{y})$ and $m(\check{x}, \check{y})$ indicate the number of voxels in the column/bar of $(\check{x}, \check{y})$ in $\overline{D}$ and $\overline{O}$, respectively.

2.5.3. Log-Likelihood Ratio

The Log-Likelihood Ratio (LLR) expresses how many times more likely data is under one model than another. LLR of the

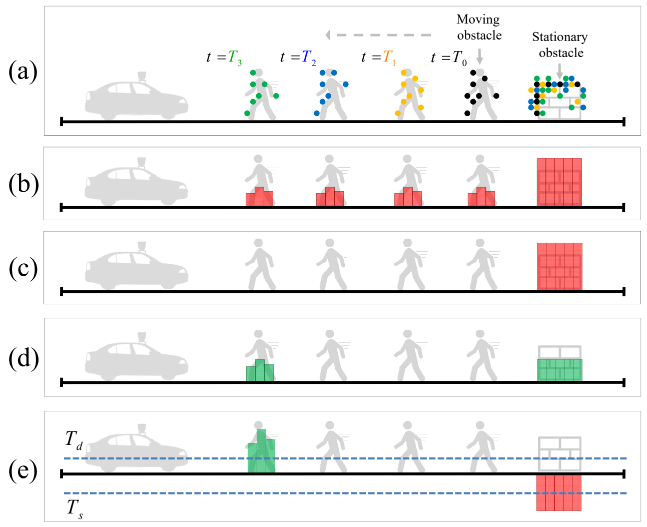

Figure 7: The process used for the creation of binary masks of the stationary and moving voxels. (a) shows a moving pedestrian and a stationary obstacle. The car in the left represents the ego-vehicle. The black, orange, blue and green points are hypothetical LIDAR hitting points that occur in different time steps. As it can be seen, since the stationary obstacle captures multiple scans, it will evidence a higher occupancy value in comparison with the moving obstacle that occupies different locations. (b) and (c) show 2D counters computed from the candidate stationary voxel grid $\overline{D}$ before and after pre-processing (H_s), respectively. (d) shows the 2D counter computed from the candidate moving voxel grids H_d. (e) shows the output of the log-likelihood ratio of (c) and (d). T_s and T_d are the thresholds used for the computation of the binary masks.

2D counters H_s and H_d is used to determine the binary masks for the stationary and dynamic voxels, and is given by:

$$R(\check{x},\check{y}) = log\frac{max\{H_d(\check{x},\check{y}),\varepsilon\}}{max\{H_s(\check{x},\check{y}),\varepsilon\}} \tag{5}$$

where ε is a small value (we set it to 1) that prevents dividing by zero or taking the log of zero. The counter cells belonging to moving parts have higher values in the computed LLR. Static parts have negative values and cells that are shared by both static and moving obstacles tend toward zero. By applying a threshold on $R(\check{x};\check{y})$, 2D binary masks of the stationary and moving voxels (see Fig. 7-e) can be obtained using the following expressions:

$$B_d(\check{x},\check{y}) = \begin{cases} 1 & \text{if } R(\check{x},\check{y}) > T_d \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$B_s(\check{x},\check{y}) = \begin{cases} 1 & \text{if } R(\check{x},\check{y}) < T_s \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

T_s and T_d are the thresholds used to compute the 2D decision masks for detecting the most reliable stationary and moving voxels. The static (B_s) and dynamic (B_d) binary 2D masks are applied to all levels of $\overline{D}$ and $\overline{O}$ voxel grids to generate voxels labeled as stationary (V_S) or moving (V_M).

3. Experimental Results

The presented approach was implemented in MATLAB and tested on the KITTI database [43]. Quantitative and qualitative

Table 2: Detailed information about each sequence.

Seq. name	No. of frames	Host vehicle situation	Scene condition	Object type	No. of objects Stationary	Moving
(1)	154	Moving	Highway	C.Y.P	11	5
(2)	447	Moving	Highway	C.P	67	6
(3)	373	Hybrid	Highway	C	25	14
(4)	340	Moving	Downtown	Y.P	27	25
(5)	376	Hybrid	Hybrid	C.Y.P	1	14
(6)	209	Stationary	Downtown	Y.P	0	17
(7)	145	Stationary	Downtown	P	0	10
(8)	339	Moving	Highway	C	0	18

Table 3: Main parameter values used in the proposed algorithm.

m	η	τ°	ℓ	d_{min}	υ	T_d	T_s
6	6	10	10	20	10	5	50

evaluations were performed to evaluate the robustness and performance of the proposed method. A brief description of the dataset and the methodology used in the experiments, as well as the experimental results on ground surface estimation and object detection (OD), are provided in the next sections.

3.1. *Dataset - 'Object Tracking Evaluation'*

The KITTI dataset was captured using a Velodyne 3D laser scanner and a high-precision GPS/IMU inertial navigation system. The Velodyne HDL-64E spins at 10 frames per second with 26.8 degree vertical field of view ($+2^\circ/-24.8^\circ$ up and down), provides 64 equally spaced angular subdivisions (approximately 0.4°) and angular resolution of 0.09 degree. The maximum recording range is 120 m. The inertial navigation system is a OXTS RT3003 inertial and GPS system with a 100 Hz sampling rate and a resolution of 0.02 m / 0.1°.

3.2. *Experimental Setup*

For the evaluation task, we have used 8 sequences from the 'Object Tracking Evaluation' set of the KITTI Vision Benchmark Suite. Two of the sequences (6 and 7) were taken by a stationary vehicle and four of them (1, 2, 4 and 8) were taken by a moving vehicle. In the remaining sequences the vehicle went through both stationary and moving situations. The dataset was captured in highways and roads in urban and rural areas (highway) or alleys and avenues in downtown areas (downtown). Different types of objects such as cars (C), pedestrians (P) and cyclists (Y) are available in the scenes. The total number of objects (stationary and moving) that are visible in the perception field of the vehicle is also reported per sequence. The characteristics of each sequence are summarized in Table 2.

The parameter values used in the implementation of the proposed algorithm are reported in Table 3. The first parameter m is a general parameter indicating the number of merged scans. The next four parameters, (η, τ°, ℓ and d_{min}) are related to the ground surface estimation. η shows the number of multiplication of $\Delta\alpha$ used to compute each slice limits. τ° and ℓ are the maximum acceptable angle and distance between two planes respectively that are applied in the validation phase of the piecewise plane fitting. d_{min} is a threshold in centimeters. Points with heights lower than d_{min} from the piecewise planes are considered as part of the ground plane. The last three parameters (υ, T_d and T_s) are used to set the obstacle detection algorithm. υ is the voxel size in centimeters. T_d and T_s are thresholds for computing the binary mask of stationary and moving voxels. The proposed approach detects obstacles in an area covering 25 meters ahead of the vehicle, 5 meters behind it and 10 meters on the left and right sides of the vehicle, with 2 meters in height. Please notice that m and η were selected experimentally as described at the end of section 3.3.1 (see also Fig. 9).

3.3. Quantitative Evaluation

Ground-surface estimation and OD evaluation using LIDAR is a challenging task. To the best of authors knowledge, there is no available dataset with ground truth for ground estimation or general OD[1]. The closest work to ours is presented in [34], where their evaluation methodology for OD is carried out as follows: detections are projected into the image plane and a human observer performs a visual analysis in terms of: missed and false obstacles (and also for traffic isles). They presented one example of ground evaluation for the case of a quadratic road surface. We followed a similar approach for evaluating the proposed obstacle detection system.

For the evaluation of the ground estimation process, inspired by [38], we assume that all objects are placed on the same surface as the vehicle and that the base points of the 3D bounding boxes available for ground truth, are located on the real ground surface. An example of ground truth data is shown in Fig. 8.

3.3.1. Evaluation of the Estimated Ground

The average distance from the *base* of labeled objects (ground-truth) to the estimated ground surface (detailed in sect. 2.3) is used as a measure of error. The average displacement error (ADE) for every frame is defined by,

$$ADE = \frac{1}{N}\sum_{i=1}^{N} |(\overrightarrow{p_i^g - p}) \cdot \hat{n}| \tag{8}$$

where p_i^g denotes the *base* of the ground-truth 3D bounding box of the i^{th} object, with $i = (1, \cdots, N)$; N = total number of objects, and $|(\overrightarrow{p_i^g - p}) \cdot \hat{n}|$ represents the absolute distance from the *base* of object i to the estimated plane on that location. The variables p and $\hat{n}$ are the point and unit normal vector that define the corresponding surface plane, respectively. The total average displacement error (TADE) for all sequences is computed by,

$$TADE = \frac{\sum_{k=1}^{N} f_k \times (\frac{1}{M}\sum_{j=1}^{M} ADE_{kj})}{\sum_{k=1}^{N} f_k} \tag{9}$$

where j ranges from 1 to the total number of frames M, f_k is the number of frames for a given sequence k, and $k = (1, \cdots, N)$ with $N = 8$ denotes the total number of sequences.

To evaluate the proposed ground estimation method, *TADE* was computed for different number of integrated frames m and

[1]The benchmarks usually provide specific object classes *e.g.*, pedestrians, vehicles.

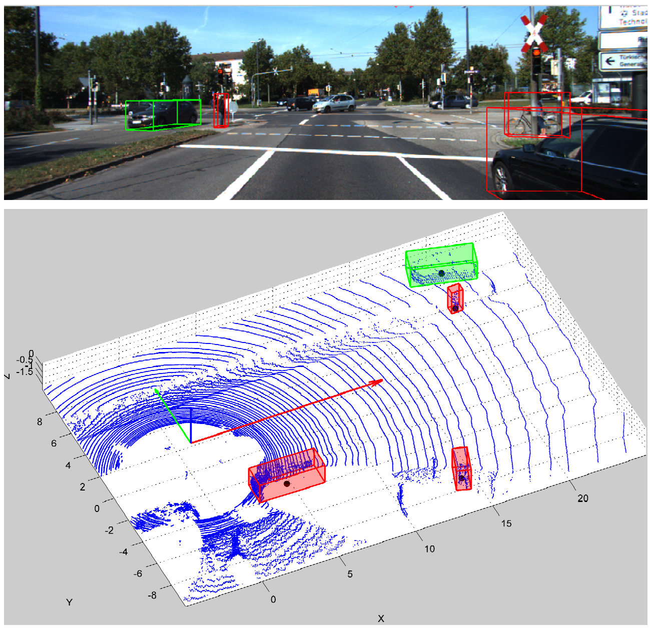

Figure 8: An example of the ground truth data. The top image shows a screenshot from the KITTI dataset with 3D bounding boxes being used to represent objects in the scene. The 3D bounding boxes are available in the KITTI dataset. This boxes are used for the evaluation of moving obstacle detection task. The 3D bounding boxes of stationary and moving objects are discriminated and labeled manually by an annotator. The bottom figure shows the corresponding 3D bounding boxes in the Euclidean space. Green and red indicate moving and stationary objects, respectively. The black dots represent the bases of the 3D bounding boxes, and are used to evaluate the ground estimation.

for different number of η (η is the multiplication of $\Delta\alpha$ that is used to compute each slice). The results over all sequences are reported in Fig. 9. The minimum $TADE = 0.086$ is achieved by the combination of $m = 6$ and $\eta = 6$.

3.3.2. Evaluation of the Stationary/Moving Obstacle Detection

The performance analysis follows a similar procedure as described in [34] using a visual analysis performed by a human observer. A number of 200 scans (25 scans for each sequence) of different scenes were selected randomly out of the more than 2300 scans available on the dataset. An evaluation was performed for the general obstacle detection and another one for the moving obstacle detection (see Fig. 10).

– For evaluating the general obstacle detection, voxel grids of stationary/moving obstacles are projected into the corresponding image, and a human observer performs an approximate visual analysis in terms of: *'missed'* and *'false'* obstacles. It should be noticed that every distinct element identified above the terrain level is considered as an obstacle (*e.g.*, pole, tree, wall, car and pedestrian). The total number of missed obstacles is 186 out of 3,011. The total number of false detected obstacles is 90. Table 4 reports the details of the obstacle detection results for each sequence, in terms of the numbers of missed and false obstacles. The highest number of missed obstacles occurs in sequences (1) and (2) that contain many thin and small

Table 4: Results of the evaluation of the proposed obstacle detection algorithm.

Seq. name	No. of obstacles		No. of missed obstacles		No. of false obstacles	
	All Obst.	Moving	All Obst.	Moving	All Obst.	Moving
(1)	501	59	83	0	0	4
(2)	288	28	56	0	0	7
(3)	281	61	24	1	0	9
(4)	381	94	10	2	0	0
(5)	254	83	1	0	7	0
(6)	791	551	9	0	37	8
(7)	336	215	1	0	46	2
(8)	179	110	2	0	0	1
Total	3011	1201	186	3	90	31

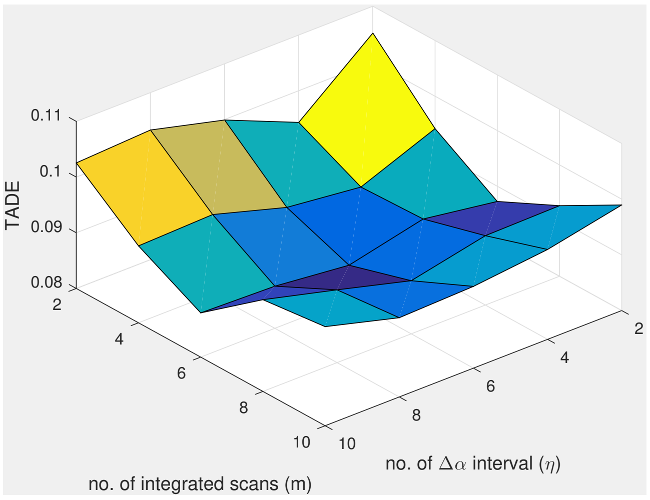

Figure 9: Evaluation of the proposed ground estimation algorithm by varying the number of integrated scans m and parameter η related to the slice sizes.

poles. Most of the false detections happen in sequences (6) and (7) that contain slowly moving objects. Some parts of very slowly moving objects may have been seen several times in the same voxels and therefore, may wrongly be integrated into the static model of the environment. The shadow of the wrongly modeled stationary obstacle stays for a few scans and causes this false detection.

– The proposed obstacle detection method is able to discriminate moving parts from the static map of the environment. Therefore, we performed an additional evaluation for measuring the moving obstacle detection performance. Among the 1201 moving obstacles present in the considered scans, only 3 moving obstacles were missed. A number of 31 obstacles were wrongly modeled as moving parts of the environment, mainly due to localization errors. Localization errors cause thin poles, observed in different locations by the ego-vehicle's perception system, to be wrongly considered as moving obstacles. The result for each sequence is also shown in Table 4.

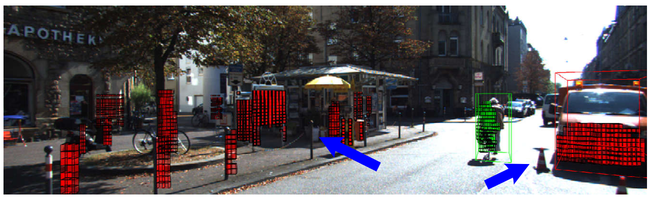

Figure 10: An example of the obstacle detection evaluation. Red and green voxels show results of the proposed method. The 3D bounding boxes of stationary and moving obstacles are shown in red and green respectively. Only green boxes are considered for the evaluation of the moving obstacle detection algorithm performance. Blue arrows show two missed obstacles (thin and small poles).

3.4. *Qualitative Evaluation*

In order to qualitatively evaluate the performance of the proposed algorithm, 8 challenging sequences were used (see Table 2). The most representative results are summarized in Fig. 11. The proposed method detects and classifies stationary and moving obstacles' voxels around the ego-vehicle when they get into the local perception field.

In the first sequence, our method detects a cyclist and a car as moving obstacles, while they are in the perception field, and models the walls and stopped cars as part of the static model of the environment. The second and third sequences show moving ego-vehicle in urban areas roads. The proposed method models trees, poles and stopped cars as part of the stationary environment and moving cars and pedestrians as dynamic obstacles. The sequence number (4) shows a downtown area, where the proposed method successfully modeled moving pedestrians and cyclists as part of the dynamic portion of the environment. Pedestrians without a movement correctly become part of the stationary model of the environment. Sequence number (5) shows a crosswalk scenario. Our method models passing pedestrians as moving objects, represented in the image by the green voxels. In sequences number (6) and (7), the vehicle is not moving. Most of the moving objects are pedestrians whom our method successfully detects. In particular, notice the last image of sequence number (6) and the first image of sequence number (7) which represent very slowly moving pedestrians that may temporarily be modeled as stationary obstacles, which will not be critical in practical applications. Sequence number (8) shows a road with moving vehicles. The proposed method performs well on most of the moving vehicles.

3.5. *Computational Analysis*

There is a compromise between the computational cost versus the detection performance of the method presented here.

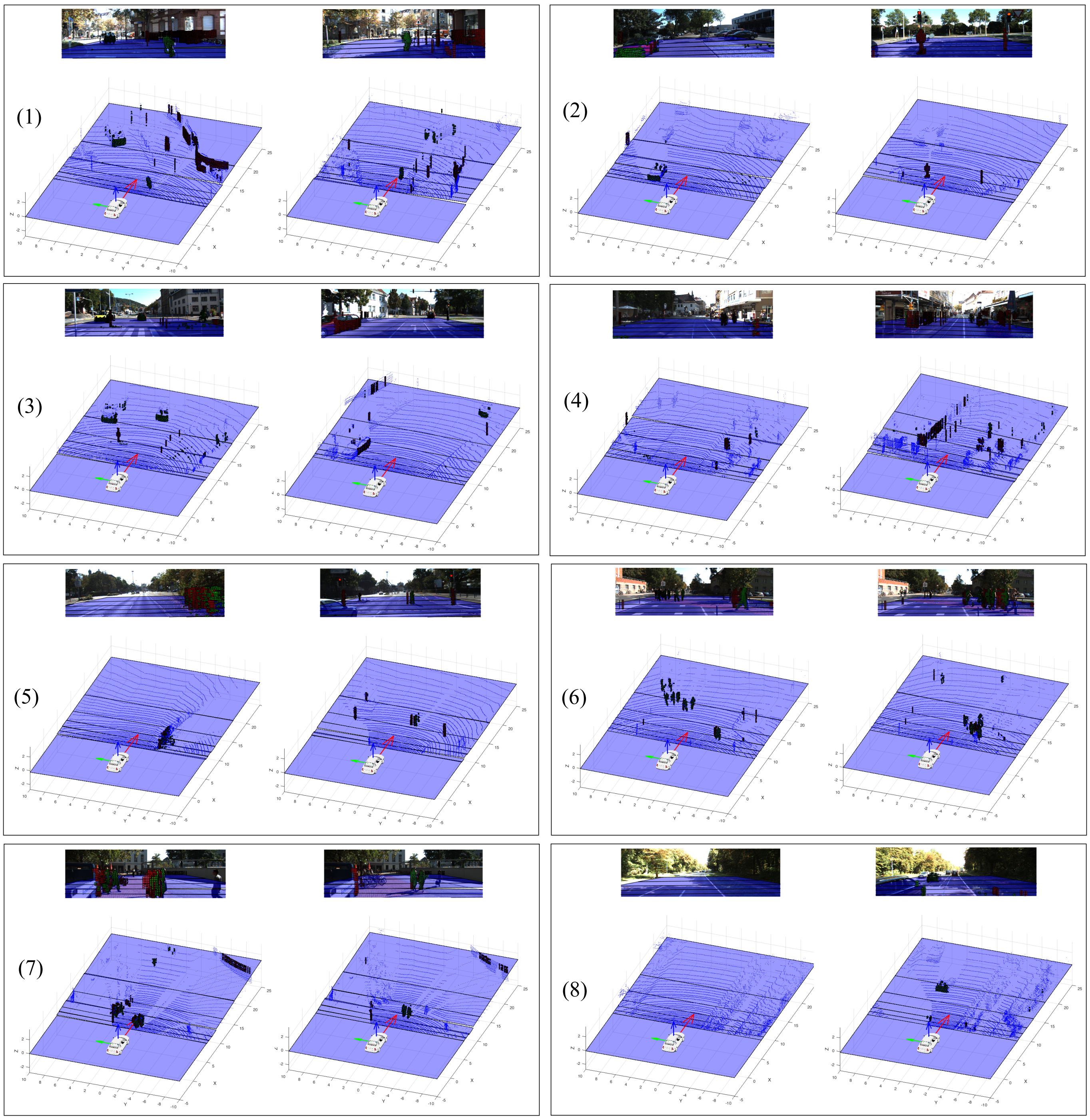

Figure 11: Sample screenshots of obstacle detection results obtained for sequences 1 to 8 as listed in Table 2 and its corresponding representation in 3D. Piecewise ground planes are shown in blue. Stationary and moving voxels are shown in red and green respectively.

Clearly, as the number of integrated scans increases, the performance in terms of stationary and moving object detection is improved. However, it adds an additional computational cost and makes the method becomes slower. On the other hand, less integrated scans make the environment model weaker. Overall, the proposed method presents satisfactory results when the number of integrated scans is greater than 4.

The computational cost of the proposed method depends on the size of the local grid, the size of a voxel, the number of integrating scans, and the number of non-empty voxels (this is because only non-empty voxels are indexed and processed). The experiments reported in this section were conducted on the first sequence and with a fixed sized local grid. The scenario has in average nearly 1% non-empty voxels. The size of a voxel and the number of integrating point-clouds are two key parameters that have a correspondence with the spatial and temporal

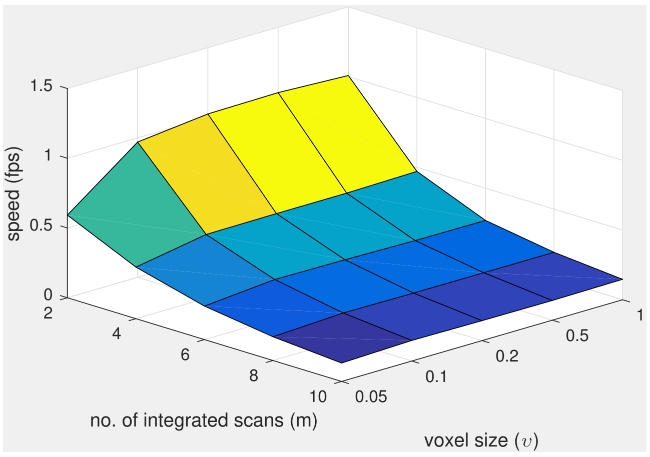

Figure 12: Computational analysis of the proposed method as a function of the number of integrated scans m and the voxel size υ, where the voxel volume is given by $\upsilon \times \upsilon \times \upsilon$.

Table 5: The percentages of the computational loads of the different steps of the proposed system: (a) dense point cloud generation, (b) piecewise ground surface estimation, (c) ground/on-ground obstacle separation and voxelization and (d) stationary/moving obstacle segmentation.

(a)	(b)	(c)	(d)
83.2%	7.1%	7.7%	2%

properties of the proposed algorithm respectively, and directly impact on the computational cost of the method. The experiment was carried out using a quad core 3.4 GHz processor with 8 GB RAM under MATLAB R2015a. The average speed of the proposed algorithm (in frames/scans per second) together with the value of each parameter (voxel size and number of integrating scans) are reported in Fig. 12. As it can be seen the number of integrated scans has the greatest impact on the computational cost of the proposed method. The proposed method configured with the parameters listed in Table 3 works at about 0.3 fps.

In order to evaluate what steps of the algorithm are more time consuming, the percentages of the processing loads of the different phases are reported in Table 5. The first stage is the most computationally demanding part of the algorithm, mostly because of the ICP algorithm (consuming 83.2% of the computational time). Piecewise ground surface estimation and ground/obstacle separation modules are accounted for 14.8% of total computational time.

4. Concluding Remarks and Future Work

The 3D perception of a dynamic environment is one of the key components for intelligent vehicles to operate in real-world environments. In this paper, we proposed a highly descriptive 3D representation for obstacle detection (OD) in dynamic urban environments using an intelligent vehicle equipped with a Velodyne LIDAR and an Inertial Navigation System (GPS/IMU). It has an application in safety systems of the vehicle to avoid collisions or damages to the other scene participants. A novel ground surface estimation is proposed using a piecewise plane fitting algorithm, based on a 'multi-region' strategy and on a RANSAC-approach to model the ground and separate ground/on-ground obstacles. A voxel-based representation of obstacles above the estimated ground is also presented, by aggregating and reasoning temporal data using a discriminative analysis. A simple yet efficient method is proposed to discriminate moving parts from the static map of the environment.

Experiments on the KITTI dataset, using point-cloud data obtained by a Velodyne LIDAR and localization data from an Inertial Navigation System (GPS/IMU), demonstrate the applicability of the proposed method for the representation of dynamic scenes. The system was proven robust and accurate, as the result of both a quantitative and a qualitative evaluation.

We propose two new directions as future work. First, the color information from the image can be incorporated to provide a more robust stationary/moving obstacle detection. Second, the identified moving obstacles can be further analyzed for object recognition and tracking purposes.

Acknowledgments

This work has been supported by FCT projects "AMS-HMI2012: RECI/EEIAUT/0181/2012" and "UID/EEA/0048/2013", and COMPETE 2020 program through project "ProjB-Diagnosis and Assisted Mobility: Centro-07-ST24-FEDER- 002028" with FEDER funding. The authors thank the reviewers for their comments that helped improve the journal.

References

[1] R. Murphy, Introduction to AI robotics, MIT press, 2000.

[2] C. Laugier, I. Paromtchik, M. Perrollaz, M. Yong, J. Yoder, C. Tay, K. Mekhnacha, A. Negre, Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety, Intelligent Transportation Systems Magazine, IEEE 3 (4) (2011) 4–19.

[3] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. Keller, E. Kaus, R. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, E. Zeeb, Making bertha drive - an autonomous journey on a historic route, Intelligent Transportation Systems Magazine, IEEE 6 (2) (2014) 8–20.

[4] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, D. Ferguson, Autonomous driving in urban environments: Boss and the urban challenge, Journal of Field Robotics 25 (8) (2008) 425–466.

[5] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, S. Thrun, Junior: The stanford entry in the urban challenge, Journal of Field Robotics 25 (9) (2008) 569–597.

[6] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, 6d slam with approximate data association, in: Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on, IEEE, 2005, pp. 242–249.

[7] D. Sack, W. Burgard, A comparison of methods for line extraction from range data, in: Proc. of the 5th IFAC symposium on intelligent autonomous vehicles (IAV), 2004.

[8] M. Oliveira, V. Santos, A. Sappa, P.Dias, Scene representations for autonomous driving: an approach based on polygonal primitives, in: 2nd Iberian Robotics Conference, 2015.

[9] R. Pascoal, V. Santos, C. Premebida, U. Nunes, Simultaneous segmentation and superquadrics fitting in laser-range data, Vehicular Technology, IEEE Transactions on 64 (2) (2015) 441–452.

[10] H. P. Moravec, A. Elfes, High resolution maps from wide angle sonar, in: Robotics and Automation. Proceedings. 1985 IEEE International Conference on, Vol. 2, IEEE, 1985, pp. 116–121.

[11] M. Herbert, C. Caillas, E. Krotkov, I. S. Kweon, T. Kanade, Terrain mapping for a roving planetary explorer, in: Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on, IEEE, 1989, pp. 997–1002.

[12] P. Pfaff, R. Triebel, W. Burgard, An efficient extension to elevation maps for outdoor terrain mapping and loop closing, The International Journal of Robotics Research 26 (2) (2007) 217–230.

[13] R. Triebel, P. Pfaff, W. Burgard, Multi-level surface maps for outdoor terrain mapping and loop closing, in: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, IEEE, 2006, pp. 2276–2282.

[14] Y. Roth-Tabak, R. Jain, Building an environment model using depth information, Computer 22 (6) (1989) 85–90.

[15] H. Moravec, Robot spatial perceptionby stereoscopic vision and 3d evidence grids, Perception,(September).

[16] D. Haehnel, Mapping with mobile robots, Ph.D. thesis, University of Freiburg, Department of Computer Science (December 2004).

[17] J. Ryde, H. Hu, 3d mapping with multi-resolution occupied voxel lists, Autonomous Robots 28 (2) (2010) 169–185.

[18] B. Douillard, J. Underwood, N. Melkumyan, S. Singh, S. Vasudevan, C. Brunner, A. Quadros, Hybrid elevation maps: 3d surface models for segmentation, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE, 2010, pp. 1532–1538.

[19] D. Meagher, Geometric modeling using octree encoding, Computer graphics and image processing 19 (2) (1982) 129–147.

[20] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, Octomap: an efficient probabilistic 3d mapping framework based on octrees, Autonomous Robots 34 (3) (2013) 189–206.

[21] I. Dryanovski, W. Morris, J. Xiao, Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE, 2010, pp. 1553–1559.

[22] A. Discant, A. Rogozan, C. Rusu, A. Bensrhair, Sensors for obstacle detection-a survey, in: Electronics Technology, 30th International Spring Seminar on, IEEE, 2007, pp. 100–105.

[23] N. Bernini, M. Bertozzi, L. Castangia, M. Patander, M. Sabbatelli, Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey, in: Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on, IEEE, 2014, pp. 873–878.

[24] Z. Zhang, R. Weiss, A. R. Hanson, Obstacle detection based on qualitative and quantitative 3d reconstruction, Pattern Analysis and Machine Intelligence, IEEE Transactions on 19 (1) (1997) 15–26.

[25] A. Petrovskaya, M. Perrollaz, L. Oliveira, L. Spinello, R. Triebel, A. Makris, J.-D. Yoder, C. Laugier, U. Nunes, P. Bessiere, Awareness of road scene participants for autonomous driving, in: Handbook of Intelligent Vehicles, Springer, 2012, pp. 1383–1432.

[26] S. Sivaraman, M. M. Trivedi, Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis, Intelligent Transportation Systems, IEEE Transactions on 14 (4) (2013) 1773–1795.

[27] A. Vatavu, R. Danescu, S. Nedevschi, Stereovision-based multiple object tracking in traffic scenarios using free-form obstacle delimiters and particle filters, Intelligent Transportation Systems, IEEE Transactions on 16 (1) (2015) 498–511.

[28] A. Asvadi, P. Peixoto, U. Nunes, Detection and tracking of moving objects using 2.5d motion grids, in: Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on, 2015, pp. 788–793.

[29] D. Pfeiffer, U. Franke, Efficient representation of traffic scenes by means of dynamic stixels, in: Intelligent Vehicles Symposium (IV), 2010 IEEE, 2010, pp. 217–224.

[30] A. Broggi, S. Cattani, M. Patander, M. Sabbatelli, P. Zani, A full-3d voxel-based dynamic obstacle detection for urban scenario using stereo vision, in: Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on, 2013, pp. 71–76.

[31] A. Azim, O. Aycard, Layer-based supervised classification of moving objects in outdoor dynamic environment using 3d laser scanner, in: Intelligent Vehicles Symposium Proceedings, 2014 IEEE, 2014, pp. 1408–1414.

[32] R. Labayrade, D. Aubert, J.-P. Tarel, Real time obstacle detection in stereovision on non flat road geometry through" v-disparity" representation, in: Intelligent Vehicle Symposium, 2002. IEEE, Vol. 2, IEEE, 2002, pp. 646–651.

[33] A. D. Sappa, R. Herrero, F. Dornaika, D. Gerónimo, A. López, Road approximation in euclidean and v-disparity space: a comparative study, in: Computer Aided Systems Theory–EUROCAST 2007, Springer, 2007, pp. 1105–1112.

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

[35] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[36] A. V. Petrovskaya, Towards dependable robotic perception, Stanford University, 2011.

[37] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al., A perception-driven autonomous urban vehicle, Journal of Field Robotics 25 (10) (2008) 727–774.

[38] A. Ess, K. Schindler, B. Leibe, L. Van Gool, Object detection and tracking for autonomous navigation in dynamic environments, The International Journal of Robotics Research 29 (14) (2010) 1707–1725.

[39] H. Badino, U. Franke, D. Pfeiffer, The stixel world-a compact medium level representation of the 3d-world, in: Pattern Recognition, Springer, 2009, pp. 51–60.

[40] U. Franke, C. Rabe, H. Badino, S. Gehrig, 6d-vision: Fusion of stereo and motion for robust environment perception, in: Pattern Recognition, Springer, 2005, pp. 216–223.

[41] A. Asvadi, P. Peixoto, U. Nunes, Two-stage static/dynamic environment modeling using voxel representation, in: Robot 2015: Second Iberian Robotics Conference, Springer, 2016, pp. 465–476.

[42] P. J. Besl, N. D. McKay, Method for registration of 3-d shapes, in: Robotics-DL tentative, International Society for Optics and Photonics, 1992, pp. 586–606.

[43] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The KITTI dataset, The International Journal of Robotics Research 32 (11) (2013) 1231–1237.