Low-Obstacle Detection Using Stereo Vision

Robert Bichsel[1] and Paulo V K Borges[2]

***Abstract*— Real-time obstacle detection is a key component of autonomous vehicles. In this context, low obstacles are particularly challenging, as they are often discarded by traditional algorithms. Curb detection methods that can potentially be suitable for the problem usually target roads with clearly defined curbs and sidewalks. We propose a real-time algorithm for the detection of low obstacles (including, but not restricted to curbs), merging 2-D and 3-D information from stereo imaging. A set of candidate object lines is extracted based on their combined 2-D and 3-D features, tracked over time and clustered according to a novel similarity metric. Finally, a 3^{rd} order polynomial spline is fitted to each cluster to represent the obstacle. The proposed system can deal with noisy and incomplete point clouds and keeps the model assumptions to a minimum. To evaluate the algorithm, a new stereo dataset is provided and made available online. We present experiments in different scenarios and lighting conditions, illustrating the applicability of the method.**

I. Introduction

In the fields of autonomous vehicles and ground robots, the detection of obstacles is an essential element for higher-level tasks such as navigation and path planning. The problem has attracted significant attention from the robotics community over the years. A number of approaches have been proposed to tackle this key issue, from simple algorithms that detect an obstacle and stop the robot, to more complex methods that determine the position and size of the obstacle, preparing strategies to navigate around it. Obstacle detection algorithms are often limited to objects with a certain height above ground. Very low obstacles with height close to the ground level, such as curbs, pallets, or beams, may be hard to distinguish from the ground, but can still represent important obstacles. In this work, we present a stereo-vision based method to efficiently detect low obstacles, combining 2-D and 3D data.

Among low obstacles, the detection of curbs has drawn significant attention, being used as a support system in manned vehicles or as a localization cue for autonomous cars. In the literature, apart from vision, inputs from a variety of sensors have been used to detect curbs, such as LIDAR, time-of-flight and visual cameras. Time-of-flight cameras and LIDAR sensors are accurate and have low noise levels. As they generally require less computational resources in comparison to standard cameras, extensive research has focused on these devices. Research focusing on visual sensors is much rarer.

[1]R. Bichsel is with the Robotics, Systems and Control Program at ETH Zürich. rbichsel@ethz.ch

[2]P. Borges is with the Autonomous System Laboratory - CSIRO Digital Productivity Flagship - 1, Technology Court, Pullenvale, QLD, 4066, Australia. He is also an Adjunct Senior Lecturer at the ITEE School, University of Queensland, Australia. vini@ieee.org

A. Related Work

Pollard *et al.* [1], for example, use three LIDAR sensors with different orientations on a personal mobility vehicle to detect curbs and other impassable obstacles of various shapes. Zhang [2] detects the road edges in an inclined LIDAR measurement by identifying local height minima and maxima. Another approach [3] relying on an inclined LIDAR sensor uses an Extended Kalman Filter to simultaneously filter and segment the data. The lines best fitting to this segmentation as well as to the orientation of the car are chosen as curbs. This approach was improved by adding a monocular camera to the setup [4]. A different framework to combine LIDAR and cameras was proposed by Aufrère *et al.* [5]. An inclined laser is used to detect a curb in its measurement plane, at a certain distance from the vehicle. The camera image is then used to expand the identified curb point along the brightness edge defined by the point. The use of a time-of-flight camera for the detection of curbs and ramps has also been proposed [6]. Using an adaption of a Random Sample Consensus algorithm (RANSAC) [7], the authors robustly identify the artifacts present in the measured point cloud.

The sensors used in [1]- [6] are rather expensive and high-maintenance. Moreover, most LIDAR sensors only provide 2D measurements in a plane, and require mechanically activated mounts or the appropriate motion to provide 3D data. Stereo cameras, on the other hand, are cost-efficient and can provide 2D and 3D information. Approaches to detect curbs from stereo vision are less common. In earlier works, Se and Brady [8] detect clusters of straight, parallel lines in 2D space using the Hough transform for each one of the images. With the disparity information of the two images, two ground planes in 3D space can be fitted, one to the outside and one to the inside region of the curb. Regarding speed, 'v-disparity' images for obstacle detection can be very efficient [9], however they struggle to identify low objects, as there is not sufficient data for the vertical histogram.

Turchetto and Manduchi [10] use a weighted Hough transform in one of the images to extract one predominant curb line per scene. The weights for the Hough transform are given by the dot product of the brightness gradient and the elevation gradient. In a follow-up, Lu and Manduchi [11] use a curvature index (from range data) as a weight for the Hough transform to determine candidate lines (peaks of the Hough transform). Finally, the 3D points of the candidate line are refined using regression, and the ends of the curb segments are found. The algorithm presented in Siegmund *et al.* [12] uses a powerful stereo-vision engine to calculate a dense, high-accuracy 3D point cloud. A Digital Elevation Model (DEM)

is computed from this data and a model with a street surface and a sidewalk surface is created. In an iterative process, the parameters of these surfaces are estimated from the DEM, using Conditional Random Fields (CRF) classification. A similar framework using CRF that is extended to posts, walls, fences, etc, has shown good results [13]. Using the most probable path of the vehicle has also been considered [14], which is suitable when structured navigation can be assumed. Oniga *et al.* [15] detect edges in a DEM based on dense stereo vision. Temporally persistent edges are classified as candidate curb points, and a Hough accumulator is used to extract straight curb segments. The largest curb segment is iteratively extended towards smaller segments to account for curved curbs as chain of segments. This approach was altered later by adding cubic polynomials as models, based on the 3D data [16]. A DEM is built and only elements with a high enough magnitude of the height gradient are considered. A 3^{rd} order polynomial is fit to these points using a RANSAC algorithm. The polynomial is re-analyzed to find the extremities of the curb and fill small gaps. The robustness of this approach for poor 3D data can be further enhanced using temporal integration [17]. In a global frame, a cubic spline can thus be fitted to the DEM points that were used for computing each single frame polynomial. Kellner *et al.* [18] detect curbs independently of their orientation in relation to the car, which makes the solution more general.

Another methodology proposed by Oniga *et al.* [19] detects obstacles, the road surface and traffic isles by analyzing solely the 3D structure of the environment without any visual priors. In a first step, a DEM is created and a quadratic road surface model is estimated using a combination of RANSAC and region growing. Then, the density of 3D points per surface unit is analyzed: regions with high densities are classified as obstacles, as they are likely to be due to vertical structures. Regions that are not classified as obstacles but have a height above the ground plane within a certain bracket are classified as traffic isles, and regions close to the road surface estimate are classified as road surface. Finally, a temporal persistence filter is applied to check for inconsistencies in the time domain and thus eliminate outliers. This algorithm, however, uses a dense 3D-information generated by an embedded stereo-vision system. The requirement for the quality of this data (subpixel resolution, dense 3D information for smooth road surfaces) is thus very high.

B. Contribution

The approach presented in this paper uses stereo vision for sensing. A 3D point cloud is calculated from the disparity of the stereo images, downsampled into a DEM, followed by the estimation of a linear ground plane. Aside from the 3D information, a set of 2D-lines is extracted from the left camera image using a combination of edge detection and a probabilistic Hough transform. These lines are projected onto the ground plane and the DEM points in their vicinity are examined. If these points suggest that the line is close to the ground and that there is a height discrepancy (in 3D) between the left and the right side, the line is accepted as obstacle line. Using a novel similarity metric, the obstacle lines are efficiently clustered. Each cluster is reduced to a cubic polynomial spline using a least-squares approach, embedding several features of the clusters in the optimization process.

The indirect approach - using lines as primitives first and only subsequently clustering them and estimating the polynomial parameters - is advantageous under limited range data, as it only needs a small number of range points to support the lines found in the 2D data. Furthermore, in contrast to curb detection algorithms [8] - [19], the model assumptions for this approach are not targeted solely on curbs, but on low obstacles in general. It was initially developed for the detection of low obstacles in an industrial environment, where in addition to curbs, obstacles such as pallets or beams are frequently present. The motivation behind this targeted scenario lies in the fact that, although the use of driverless cars on public roads is still a number of years away of becoming a reality, in industrial areas autonomous ground vehicles are incrementally gaining space. Common tasks in this domain are material handling, mining operations, carrying load between different areas of a production facility, and the transportation of objects. As illustrated by our experiments, the broader spectrum of detected obstacles make the algorithm applicable in industrial environments, and can be applied in combination with traditional "high obstacle" detection methods.

This paper is organized as follows. An overview of the algorithm is given in Section II. Then, the steps of the algorithm are described: the construction of a DEM and the ground plane estimation (Section III), the detection of obstacle line candidates (Section IV), the clustering of the obstacle lines and the fitting of curves to the cluster (Section V). Section VI presents the experimental results, followed by relevant conclusions in Section VII.

II. Overview of the Algorithm

The proposed algorithm relies on three inputs: (i) A dense 3D point cloud from a stereo-vision system calculated with Efficient Large-Scale Stereo (ELAS) matching algorithm [20], (ii) a rectified image from one of the cameras, and (iii) any type of odometry information (e.g., visual odometry, wheel odometry, etc.). We employ visual odometry in this work.

A. Criteria

For detection purposes, we assume that a low obstacle meets the following criteria:

- A low obstacle (such as a curb, for example) contains a piecewise straight line in any direction parallel to the ground. This assumption can be interpreted as low obstacles being "cut off" at a certain height instead of being high enough to be detected by an obstacle detection algorithm. This "cut-off" line, henceforth called *obstacle line*, can be detected in the 2D image.
- The obstacle line(s) of a low obstacle is (are) close to the ground.
- The heights above ground on the left and on the right hand side of the obstacle line must differ by a minimum

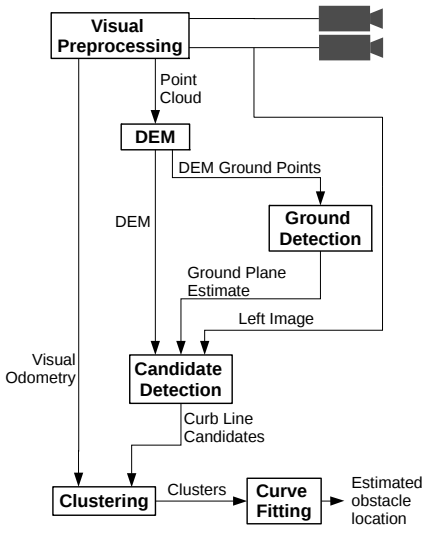
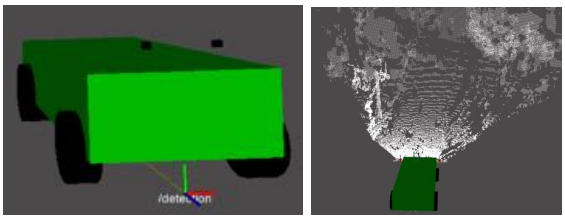
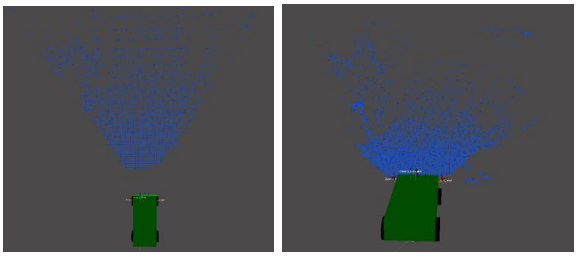


Fig. 1. Outline of the main steps in the obstacle detection algorithm. Each block apart from the visual preprocessing is discussed in this paper. The DEM and the ground detection in Section III, the candidate detection in Section IV, the clustering in Section V and the curve fitting in Section V-C.



(a) (b)

Fig. 2. (a): Model of the vehicle with its coordinate system: x-axis (red), y-axis (green) and z-axis (blue). (b): The 3D point cloud used for the DEM.



(a) (b)

Fig. 3. (a): Top view of the DEM point cloud (note the grid structure). (b): Front view of the DEM.

threshold δ. This criterion identifies lines that delimit a step function, such as obstacles that stick out from the ground, and discards other lines such as markers or shadows.

B. Evaluation

These criteria can be evaluated by the following rules:

1) Firstly, a DEM is computed and a linear estimation of the ground plane is made using a RANSAC algorithm.
2) Secondly, on a frame-to-frame basis, a set of potential obstacle lines is elicited based on the criteria mentioned in Section II-A, using the DEM as well as the 2D image. This step is henceforth denoted 'candidate detection'.
3) Thirdly, the potential obstacle lines of multiple frames are compared to each other. An accumulation of similarly oriented obstacle lines is defined as a cluster. Isolated potential obstacle lines are considered to be false positives and are rejected.
4) Finally, three cubic splines are fitted to each obstacle cluster, a skeleton line and two error boundaries (left and right). It is assumed that the actual curb lies within the error boundaries.

A summary of these steps is shown in Figure 1, and each part of the algorithm is detailed in the following sections.

III. Digital Elevation Model and Ground Detection

This section discusses preliminary steps required by the algorithm. This includes the computation of a DEM from the 3D point cloud input (Figure 2(b)) and a linear ground plane estimation based on those points of the DEM that are assumed to lie on the ground. A DEM is a discretized representation of elevation data, where the 3D points are sampled into a fixed grid structure. The estimation of the ground plane serves as a reference for the algorithm when the lines in 2D space are compared to the points in 3D space (cf. Section IV).

A. DEM

The coordinate system used has its origin on the ground in front of the vehicle, laterally centered. The x-axis points to the left, the y-axis upward and the z-axis to the front in the forward direction, as illustrated in Figure 2(a).

To reduce the computational cost for real-time operation, the points of the 3D point cloud are transformed into a DEM (cf. Figure 3). As often done in literature [19], a rectangular grid (in our implementation we use: length=40 m, width=13 m, resolution=0.1m) of equally distributed cells is set up in the x-z-plane in front of the vehicle. For each grid cell, the maximum height of the 3D points falling into the cell defines the height of the cell. The new point cloud that is now used (instead of the dense 3D point cloud) contains a point for every non-empty grid cell, with the center of the cell as the x- and z-coordinates, and its height as the y-coordinate. Although other metrics (such as the mean height of the 3D points within a cell) could be used to define the cell height, the height of the highest 3D point provides robustness against outliers in the negative y-direction.

B. Identification of Ground Points in the DEM

Estimating the ground plane based on the whole DEM presents a drawback: instead of the actual ground plane, the result of the RANSAC estimation (described in Section III-C) can be a building wall or a slanted surface next to the ground plane. This issue can be overcome by only feeding to the RANSAC algorithm a subset of DEM points that are very likely to be on the ground. Starting from points in front of the vehicle, the ground points are generated according to:

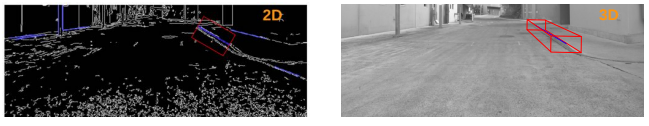
- The points in the first two non-empty rows in front of the vehicle are declared ground points.
- For every successive row, the previous non-empty rows are assessed. Every point that has a similar height to its predecessor is declared as a ground point.

C. Estimation of Ground Plane Parameters

Although roads usually present some degree of curvature, we assume a linear model for our ground estimation. This is a simplification that is commonly used [8], [10], [11] and has the advantage of a substantially lowered complexity of the algorithm. A RANSAC algorithm with a linear plane model is run on the subset of ground points to yield the ground plane estimate. Finally, a median filter in the time domain is applied to a number N_{med} of frames. This removes outliers and thereby avoids jerks.

IV. Candidate Detection

The left rectified input image $\mathbf{I}$ is used to extract the visual information. $\mathbf{I}$ is cropped to its lower half and enhanced with standard gamma correction [21] defined by the exponent γ for a better performance in shadowy areas, yielding the image $\mathbf{I}_\gamma$. A Canny edge detector is applied to $\mathbf{I}_\gamma$ to create the Canny image $\mathbf{I}_C$, a binary image of edges. Then, applying the probabilistic Hough transform to $\mathbf{I}_C$, a set of Hough lines $\mathcal{H}$ is formed. Subsequently, the depth information of the 3D point cloud must be transferred onto the lines of $\mathcal{H}$ in order to assess their aptitude of being part of an obstacle. For this goal, one possibility is to project the 3D points to the 2D image and consider the points close to the line. However, even though these points are close in the 2D space, they are not necessarily close to the line in 3D. Another way of associating the 3D points with the 2D lines is to project the lines onto the ground plane in 3D space and assess the points in the vicinity of the line in 3D space. The downside in this case is that lines far off the ground will yield faulty projections. A two-step assessment using both of these methods overcomes their respective drawbacks. This assessment includes a selection step, where the points are projected onto 2D, and a validation step, where the lines are projected onto the ground plane in 3D. In both of these steps, each line is evaluated based on the subset of points assigned to them. The evaluation process is described in more detail in Sections IV-A and IV-B.



(a) Canny image with the lines detected by the probabilistic Hough transform (blue). The test window in 2D space is indicated in red.

(b) Gamma-corrected lower half of the left image. For illustration, a line detected by the Hough transform (blue) is projected into the image and the test box around it is indicated in red.

Fig. 4. Illustration of the candidate detection. Each line detected by the probabilistic Hough transform (blue) in the Canny image (b) is assessed twice: A first time based on the subset of points falling into a test window in 2D space (b), and a second time based on a test box in 3D space (a).

A. Selection of Ground Lines

The set of 3D points $\mathcal{P}$ in the point cloud is back-projected onto the image plane to generate the set $\mathcal{P}'$ of points in pixel coordinates. Then, for each Hough line $\bar{h}_i$ in $\mathcal{H}$:

- The subset $\mathcal{P}'_{\bar{h}_i}$ of points in $\mathcal{P}'$ that are contained in a test window of a given width w_{window} around $\bar{h}_i$ is determined (cf. Figure. 4(a)). The corresponding subset of points in 3D-space is denoted $\mathcal{P}_{\bar{h}_i}$.
- For subset $\mathcal{P}_{\bar{h}_i}$, the mean $\mu_{\bar{h}_i}$ and the variance $\sigma^2_{\bar{h}_i}$ of the height above ground (given by the ground plane estimate) are determined.
- If $\sigma^2_{\bar{h}_i}$ is below a certain threshold α, the height variance is small and it is assumed that $\bar{h}_i$ is horizontal or close to horizontal relative to the ground plane estimate. Additionally, if $\mu_{\bar{h}_i}$ is below a certain threshold κ, the line can be considered to be close to the ground plane estimate. If both of these conditions are fulfilled, it is assumed that a projection of the line onto the ground plane will yield a satisfactory result, based on extensive evaluation. The line is thus added to the set of ground lines $\mathcal{G}$. Otherwise $\bar{h}_i$ is rejected.

B. Validation of Obstacle Lines

The set of ground lines $\mathcal{G}$ (in pixel coordinates) is projected onto the estimated ground plane (in 3D-space) forming the set of 3D-lines $\mathcal{G}'$. For each line $\bar{g}'_i$ in $\mathcal{G}'$:

- The subset of points $\mathcal{P}_{\bar{g}'_i}$ that are contained in a test box of width w_{box} and height h_{box} around $\bar{g}'_i$ is determined (Figure 4(b)).
- Firstly, with the new subset of points $\mathcal{P}_{\bar{g}'_i}$ the findings in the selection step are verified (mean $\mu_{\bar{g}'_i}$ and variance $\sigma^2_{\bar{g}'_i}$ of height below their respective thresholds α and κ, as discussed in Section IV-A). If this is not the case, $\bar{g}'_i$ is deleted from $\mathcal{G}'$.
- Again, for subset $\mathcal{P}_{\bar{g}'_i}$, the mean height above ground on the left hand side and on the right hand side of the line ($\mu^{left}_{\bar{g}'_i}$ and $\mu^{right}_{\bar{g}'_i}$) are determined.
- If the absolute difference $|\mu^{left}_{\bar{g}'_i} - \mu^{right}_{\bar{g}'_i}|$ is above a threshold δ, it is concluded that the line originates from a step (obstacle, curb or other). Hence,$\bar{g}'_i$ is accepted as a close-to-ground obstacle and added to the set of obstacle lines in 3D space $\mathcal{L}$. These obstacle lines are the candidates that are now clustered. If $|\mu^{left}_{\bar{g}'_i} - \mu^{right}_{\bar{g}'_i}|$ is below the threshold, it is assumed that the line represents something flat on the ground (e.g. shadow, lane marker), and thus it is eliminated as a candidate.

V. Clustering and Curve Fitting

The obstacle lines detected so far are simply an accumulation of snippets of the actual obstacle and may contain false positives. A more accurate representation of the obstacle with fewer false positives can be obtained by appropriately clustering the lines. To this end, a metric is defined to measure the dissimilarity between two obstacle lines. Based on this metric, it is assessed whether the lines are similar enough to be considered as belonging to the same obstacle, and if

so, clustered together. Such clusters are defined as actual low obstacles. Single lines and clusters with a small number of elements are considered false positives and are rejected. The details of the clustering algorithm are discussed in this section.

A. Metric

The orientation of the obstacle lines is an important information. For measuring the dissimilarity between two obstacle lines $\bar{l}_i$ and $\bar{l}_j$, common metrics like the Euclidean distance are not well suited as they lack orientation information. A custom metric d using both distance and orientation as inputs is defined as the weighted sum of three quantities: The lateral distance d_{lat}, the longitudinal distance d_{long} and angle α:

$$d[\bar{l}_i, \bar{l}_j] = w_{lat} d_{lat}[\bar{l}_i, \bar{l}_j] + w_{long} d_{long}[\bar{l}_i, \bar{l}_j] + w_\theta |\theta[\bar{l}_i, \bar{l}_j]| \tag{1}$$

where w_{lat}, w_{long} and w_θ are the weight parameters and $|.|$ represents the absolute value. The parameters are tuned based on experimental evaluation on training sets, and are further discussed in Section VI. The distances are defined as:

1) Lateral Distance: The lateral distance is defined as the maximal perpendicular distance from an end-point of one line to another line. It can be found as follows:

$$\begin{aligned} d_{lat}[\bar{l}_i, \bar{l}_j] = max(&dist(\mathbf{p}^1_{\bar{l}_i}, \bar{l}_j), dist(\mathbf{p}^2_{\bar{l}_i}, \bar{l}_j), \\ &dist(\mathbf{p}^1_{\bar{l}_j}, \bar{l}_i), dist(\mathbf{p}^2_{\bar{l}_j}, \bar{l}_i)) \end{aligned} \tag{2}$$

where

$$dist(\mathbf{p}^e_{\bar{l}_i}, \bar{l}_j) = |[(\mathbf{p}^e_{\bar{l}_i} - \mathbf{m}_{\bar{l}_j}) - (\mathbf{p}^e_{\bar{l}_i} - \mathbf{m}_{\bar{l}_j}) \cdot \mathbf{v}_{\bar{l}_j}] \cdot \mathbf{v}_{\bar{l}_j}| \tag{3}$$

and $\mathbf{p}^e_{\bar{l}_i}$ is the endpoint e $(e = 1, 2)$ of line $\bar{l}_i$, $\mathbf{m}_{\bar{l}_j}$ denotes the midpoint of line $\bar{l}_j$, and $\mathbf{v}_{\bar{l}_j}$ stands for the unit vector in direction of line $\bar{l}_j$. The operator $\cdot$ corresponds the dot product between two vectors.

2) Longitudinal Distance: The longitudinal distance is found by projecting the end points of one line onto the other line and calculating the distance between the midpoint of the line and the projected points. If one of the projected end points lies between the end points of the second line, the longitudinal distance is set to zero. Otherwise the longitudinal distance is set to the minimum distance between one of the projected points and the end points of the second line. The longitudinal distance is determined by:

$$\begin{aligned} d_{long}[\bar{l}_i, \bar{l}_j] = \max(\min(&proj(\mathbf{p}^1_{\bar{l}_i}, \bar{l}_j), proj(\mathbf{p}^2_{\bar{l}_i}, \bar{l}_j), \\ &proj(\mathbf{p}^1_{\bar{l}_j}, \bar{l}_i), proj(\mathbf{p}^2_{\bar{l}_j}, \bar{l}_i)), 0) \end{aligned} \tag{4}$$

where

$$proj(\mathbf{p}^e_{\bar{l}_i}, \bar{l}_j) = |(\mathbf{p}^e_{\bar{l}_i} - \mathbf{m}_{\bar{l}_j}) \cdot \mathbf{v}_{\bar{l}_j}| - l_{\bar{l}_j}/2 \tag{5}$$

and $l_{\bar{l}_j}$ is the length of the line $\bar{l}_j$.

3) Angle θ: The angle between the two lines is given by:

$$\theta[\bar{l}_i, \bar{l}_j] = \min(\arccos(\mathbf{v}_{\bar{l}_i} \cdot \mathbf{v}_{\bar{l}_j}), \pi - \arccos(\mathbf{v}_{\bar{l}_i} \cdot \mathbf{v}_{\bar{l}_j})) \tag{6}$$

If the dissimilarity between two obstacle lines defined in (1) is below a threshold ϵ, they fulfill the proximity criterion.

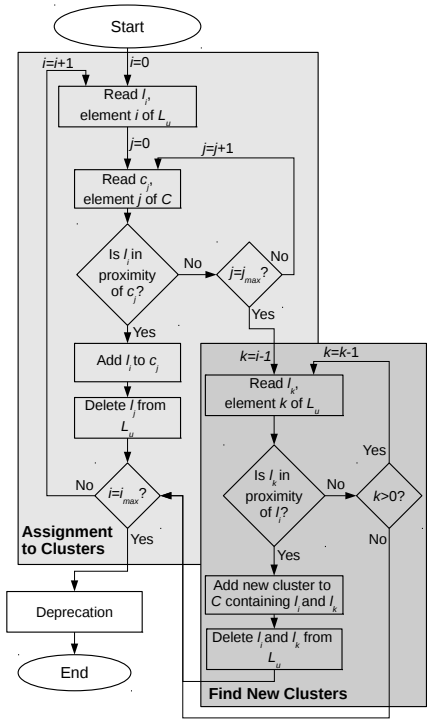


Fig. 5. Flowchart of the clustering algorithm. First, the algorithm checks whether an obstacle line can be assigned to an existing cluster (this module is represented by the light grey box). If not, the system evaluates whether it can initialize a new cluster with one other obstacle line (dark grey box).

B. Clustering Algorithm

For clustering, a (initially empty) set of clusters $\mathcal{C}$ is defined. Each cluster c_j in $\mathcal{C}$ contains a set of obstacle lines. A line $\bar{l}_i$ is considered to belong to the cluster c_j ($\bar{l}_i \in c_j$) if it fulfills the proximity criterion with at least one of the lines in the cluster. In addition, a (initially empty) set of unassigned obstacle lines $\mathcal{L}_u$ is defined. With every incoming set of obstacle lines $\mathcal{L}$ in every new frame, $\mathcal{L}$ is projected into the global (static) reference frame using the odometry information to create a new set $\mathcal{L}'$. The elements of $\mathcal{L}'$ are added to $\mathcal{L}_u$. Then, for each $\bar{l}_i$ in $\mathcal{L}_u$, the scheme depicted in Figure 5 is applied:

1) In each c_j in $\mathcal{C}$ it is checked whether $\bar{l}_i \in c_j$. If l_i belongs to a single cluster, it is added to this cluster. If l_i belongs to two or more clusters, all these clusters are merged into a new cluster containing all the obstacle lines of the previous clusters plus $\bar{l}_i$. Then, $\bar{l}_i$ is deleted from $\mathcal{L}_u$.
2) If $l_i \notin \mathcal{C}$, it is checked whether l_i and any other obstacle line of $\mathcal{L}_u$ fulfill the proximity criterion. If so, a new cluster c_{new} containing the two obstacle lines is added to the set of clusters $\mathcal{C}$. Then, the two obstacle lines are deleted from $\mathcal{L}_u$.
3) Obstacle line deprecation: If an obstacle line has been in $\mathcal{L}_u$ for longer than N_{dep} loops, it is deleted.
4) Cluster deprecation: If the accumulated length of lines inside a cluster exceeds $K \cdot L$, where L is the longest distance in the cluster and K is a parameter to be tuned,

the oldest lines are deleted. Furthermore, if a cluster has existed for longer than N_{dep} loops and does not contain a minimal amount N_{min} of obstacle lines, it is deleted.

The result of this clustering algorithm is clusters of similarly oriented lines. When a cluster contains a high number of lines, the new ones replace the old ones, thus updating the cluster to the latest measurements.

C. Curve Fitting

Since a cluster of lines is a poor representation of an obstacle, we fit a curve to each cluster. For this purpose, a cubic splined polynomial (or cubic spline) is used, as it represents a fairly generic model for different obstacle shapes. One of the advantages of using splines as opposed to simple polynomial approximations, is that splines tend to avoid the high frequency (oscillating) fitting often observed in polynomial interpolations. If the points that form the obstacle border correspond to a straight line, the spline interpolation still creates a generally acceptable representation for the obstacle detection task. A weighted least-squares approach is used for this fitting problem, and three curves are considered: a skeleton line indicating the center of the obstacle, and two border lines on each side of the skeleton line. The weights are given by the lenghts of the individual lines. A single long line hence will have the same impact as a number of shorter lines (e.g. in a curved curb segment). Figure 8 illustrates the components of the curve fitting, with the borders in pink, the skeleton in red and the obstacle lines cluster in green.

VI. Experiments

A. Practical Considerations

The algorithm is implemented in C++, using the OpenCV Library [23], and the Robotics Operating System (ROS) [24]. The stereo imaging system was formed by two monochromatic Point Grey Grasshopper cameras with resolution of 800×600 pixels fitted with Kowa 2/3" lenses with 5mm focal length and aperture F1.8. The system was setup on a rig with a baseline of 25 cm, as illustrated in Figure 6(b). For the stereo matching, in our implementation we use the ELAS matching algorithm [20]. This method provides a good compromise between small matching window sizes with high matching ratios, reducing the impact of blurring effects around borders that affects correlation-based stereo methods. The cameras were mounted on a John Deere's Gator (Figure 6(a)), an electric medium size utility vehicle, and driven in an industrial park in Australia. The area contains both urban and rural characteristics, with roads, trees, sidewalks, dense buildings and open fields. In this environment, the performance of the algorithm was assessed based on isolated obstacles (e.g., pallets) as well as curbs. The algorithm was also tested in public roads, for which the cameras were mounted on a Toyota Prado. The open road experiments were performed in suburban residential areas, and contain extremely challenging conditions due to occlusions and hard shadows. In all scenarios, the algorithm was tested under different lighting conditions considering overcast and sunny weather. The datasets have been made available for testing and can be downloaded from our server[1].

The computation was executed on recent hardware (Intel Core i7, 2.7 GHz Octa-core) and takes 20-55 ms per frame, depending on the amount of visible curbs and ground points (15-25 ms for the calculation of the DEM and the extraction of the ground points, under 1 ms for the ground plane estimation and 5-30 ms for the image processing and the extraction of the obstacles).

(a) (b)

Fig. 6. (a): John Deere's Gator, the test vehicle used for the experiments. (b): The point Grey Grasshopper cameras mounted on their stereo rig, approximately 1 meter above the ground. Obs.: The thermal camera and the inertial measurement unit in the center of the rig were not used in this work.

The parameters for the algorithm were set based on extensive experimental evaluation using test sets. The median filter in the ground plane estimation (Section III-C) is set with length $N_{med} = 10$. For the detection of obstacle line candidates (Section IV), the parameters were set to w_{window} = 50 pixels, w_{box} = 0.5m, h_{box} = 0.3m, α = 0.2m, κ = 0.1m, δ = 0.08m. The clustering (Section V) performs well with w_{lat} = 7, w_{long} = 0.2, w_θ = 2, ϵ = 4, N_{dep} = 20, K = 10 and $N_{min} = 3$. And finally, for the curve fitting (Section V-C), the parameters were fixet at $L_{max} = 10m$, w_{lin} = 1, w_{slope} = 0.1 and $l_{max} = 0.5m$. Most of these parameters have limited influence on the performance and generally work for all scenarios. The parameters used for the evaluation of the candidate lines, α, κ and δ, though, have more influence and should be set carefully. The values given above were fixed for all the experiments.

B. Results

1) Industrial Environment: In the industrial environment tests, the results are divided into isolated obstacles and continuous curbs.

a) Isolated low obstacles: In this first test, pallets were used to represent isolated low obstacles. We drove the robot towards the obstacle in different lighting conditions (sunny and overcast) and at different orientations ($0\,^\circ$ and $45\,^\circ$). For each test run, we recorded whether or not the obstacle was detected and the distance between the obstacle and the vehicle. The results are shown in Table I, illustrating that the obstacle was reliably detected in every test run. However, two facts are particularly relevant. Firstly, the distance at which it is detected is consistently small (approximately 2m) compared to the detection distance of a curb, for example. The reason for this lies in the small size of the obstacle. At a far distance, the

[1]Link to the datasets: http://dx.doi.org/10.4225/08/53D9DB330C4EB

TABLE I

DETECTION DISTANCE OF PALLETS. THE TEST VEHICLE APPROACHES THE PALLETS IN A STRAIGHT LINE. AS SOON AS THE PALLETS ARE DETECTED, THE VEHICLE IS STOPPED. THE DETECTION DISTANCE IS GIVEN BY THE DISTANCE BETWEEN THE PERPENDICULAR PROJECTION OF THE CAMERA LOCATION ONTO THE GROUND AND THE OBSTACLE ON THE GROUND.

	Overcast		**Sunny**	
	0 °	45 °	0 °	45 °
Number of samples	10	10	10	10
Samples detected	10	10	10	10
Mean distance [m]	1.93	1.90	2.01	1.99
Standard deviation [m]	0.064	0.141	0.054	0.054

amount of points in the DEM available to assess the already short obstacle lines is low, making it harder to distinguish between obstacle lines and non-obstacle lines. The shorter the distance, though, the bigger the obstacle appears in the image, yielding a bigger obstacle line and hence more points in the vicinity of the obstacle line. Secondly, as can be seen in Figure 8, only the sides that can be seen as straight lines in the image are detected by the algorithm.

b) Curbs: Curbs are present in most structured environments and are arguably one of the most relevant obstacles. To assess the curb detection performance, a region covering 3m to either side of the vehicle was analyzed. The success rate is given by the ratio of *detected* length of curbs and the *total* length of curbs passing through this area. Accordingly, the rate of the false positives is given by the ratio of false positive curb length detected to the total length of curbs. The curb detection tests in the industrial park contain 11 minutes of footage, corresponding to 1,070 meters of distance covered. The results of the curb detection are shown under 'Industrial' in Table II and illustrated in Figure 8. Note that the curbs are detected even at greater distances, in contrast to small obstacles.

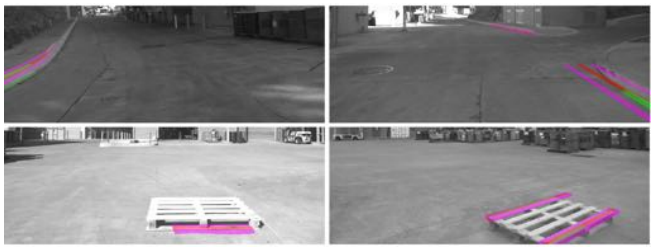


Fig. 8. Results of the obstacle detection and curb detection tests with the skeleton (red), the borders (pink) and the obstacle line clusters (green). Low obstacles are detected at various angles and different lighting conditions (two bottom images). Curbs of different curvatures are detected on either side of the road (two top images).

As with many outdoor vision applications, the main reason for missed detections is poor image quality under challenging lighting. This is the case when either the curb lies in a dark shadow , or the noise level in the image is high (e.g., the rough concrete in bright sunlight in Figure 7(a)).

The false positives are usually due to lines on the ground, like shadows (Figure 7(b)), street marks or road cracks. Most of these artefacts are filtered out because they do not satisfy

TABLE II

CURB DETECTION PERFORMANCE OF THE ALGORITHM IN AN INDUSTRIAL AND A SUBURBAN ENVIRONMENT, INCLUDING DATA FROM THE KITTI DATASET. THE SUCCESS RATE IS DETERMINED BY THE RATIO OF THE TOTAL LENGTH OF CURBS DETECTED TO THE TOTAL LENGTH PRESENT. THE FALSE POSITIVE RATIO IS THE LENGTH OF FALSE POSITIVES DETECTED DIVIDED BY TOTAL LENGTH PRESENT. KEY: TP (TRUE POSITIVE), FP (FALSE POSITIVE), NC (NO CLUSTERING).

	Industrial		**Suburban**		**KITTI**
	Overcast	*Sunny*	*Overcast*	*Sunny*	-
Distance [m]	1603	1869	425	395	-
TP	92%	84%	78%	55%	81%
TP - NC	80%	53%	80%	43%	64%
FP	1.4%	5.2%	0.5%	0.3%	3.2%
FP - NC	0.9%	4.1%	1.8%	2.2%	2.0%

the height difference criterion (Section II-B). However, the presence of faulty matches in the point cloud can trigger a false positive. As shown in Table II, this phenomenon occurs more often in sunny conditions, where hard shadows are present. Very rarely, the spline estimation discussed in Section V-C can be erroneous due to a numerical instability in the least squares approach (Figure 7(c)). This phenomenon, however, usually only lasts for 1-2 frames and disappears as soon as a new obstacle line is added to the cluster, therefore not realistically affecting the performance of the algorithm.

2) Public Roads: The second test environment is a suburban neighborhood in Brisbane, Australia. This dataset contains a number of very challenging aspects. As examples, grass growing over the curb edges and rough street/sidewalk surfaces can prevent the Hough transform from identifying lines in the images. Leaves in the gutter and slanted curbs smoothen out the height gradient along the curbs. Also, the issues faced with strong light-shadow contrasts are amplified as shadows become more scattered due to trees. This dataset also presents many discontinuities in the curbs due to parked cars and driveways. Despite the challenges, this is a very realistic scenario, which contains 17 minutes of footage, corresponding to 4,140 meters of distance covered. The results for this environment are shown in Table II under the 'Suburban' field.

In addition to our own data, we test the algorithm using the visual odometry[2] stereo data available in the KITTI dataset [25]. The nature of the issues faced in the suburban and KITTI environments are similar to those discussed above for the industrial park dataset. As the environments are more challenging, though, success rates are lowered, particularly in sunny cases. Counterintuitively, the amount of false positives in the 'Suburban' field in Table II is also reduced compared to 'Industrial' field in that table. This improvement is due to the fact that the higher noise levels not only hinder the probabilistic Hough transform from detecting actual obstacle lines, but also reduce the false positives rates. Also note that the line marks present in the suburban neighbourhood do not alter the false positive rate. Lane markers are reliably filtered

[2]From this dataset, our evaluation includes all the frames in "Sequences" 03, 06, 14 and 15.

(a) Noisy image prevents the detection of lines. (b) False positive due to shadow. (c) Faulty spline due to numerical instability.

Fig. 7. Illustration of failure cases. In difficult lighting conditions, curbs can be occasionally missed (a). Another source of error are lines on the ground combined with inaccurate point cloud measurements, yielding false positives such (b). Finally, numerical instabilities can occur in the curve fitting, yielding inaccurate splines (d).

out as they do not exhibit any height difference.

For comparison, we show results discarding the clustering and initial line estimation, performing the spline fitting directly on the 3D points from the DEM that are persistent, aiming at a similar approach to that of reference [15]. In this case, the results are presented in the rows tagged with 'NC' (No Clustering) in Table II. Depending on the density and quality of information, the performance between the proposed and the NC results are similar. In sunny cases, however, where the points are generally more segmented because of irregular lighting, the use of piecewise line-estimation and clustering performs better, on average. This is particularly noticeable in the 'Suburban' and KITTI footage.

VII. Conclusions and Future work

A practical algorithm to detect low obstacles was proposed, implemented and tested. The algorithm is able to detect curbs and other types of low obstacles, making it applicable to a range of objects. Furthermore, it is able to deal with noisy and incomplete 3D range data. The algorithm was extensively tested in an industrial environment, presenting very good results in a scenario where the road surface is smooth and there are clear and strong lines to be detected. In very challenging environments, with video footage from and suburban roads, the majority of the curbs could also be successfully be detected. Results from the integration of the algorithm into a closed-loop system in an autonomous vehicle demonstrates the applicability of the method for robotics and navigation. All the datasets used in the experiments are available online as single images and as ROS archives.

References

[1] E. Pollard, J. Perez, and F. Nashashibi, "Step and curb detection for autonomous vehicles with an algebraic derivative-based approach applied on laser rangefinder data," in Intelligent Vehicles Symposium (IV), 2013 IEEE, June 2013, pp. 684–689.

[2] W. Zhang, "Lidar-based road and road-edge detection," in Intelligent Vehicles Symposium (IV), 2010 IEEE, June 2010, pp. 845–848.

[3] W. Wijesoma, K. R. S. Kodagoda, and A. Balasuriya, "Road-boundary detection and tracking using ladar sensing," Robotics and Automation, IEEE Transactions on, vol. 20, no. 3, pp. 456–464, June 2004.

[4] K. R. S. Kodagoda, W. Wijesoma, and A. Balasuriya, "Cute: curb tracking and estimation," Control Systems Technology, IEEE Transactions on, vol. 14, no. 5, pp. 951–957, Sept 2006.

[5] R. Aufrere, C. Mertz, and C. Thorpe, "Multiple sensor fusion for detecting location of curbs, walls, and barriers," in Intelligent Vehicles Symposium, 2003. Proceedings. IEEE, June 2003, pp. 126–131.

[6] O. Gallo, R. Manduchi, and A. Rafii, "Robust curb and ramp detection for safe parking using the canesta tof camera," in Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on, June 2008, pp. 1–8.

[7] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Communications of the ACM, vol. 24, no. 6, pp. 381–395, 1981.

[8] S. Se and M. Brady, "Stereo vision-based obstacle detection for partially sighted people," in Computer Vision ACCV'98, ser. Lecture Notes in Computer Science, R. Chin and T.-C. Pong, Eds. Springer Berlin Heidelberg, 1997, vol. 1351, pp. 152–159.

[9] R. Labayrade, D. Aubert, and 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, June 2002, pp. 646–651 vol.2.

[10] R. Turchetto and R. Manduchi, "Visual curb localization for autonomous navigation," in Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 2, Oct 2003, pp. 1336–1342 vol.2.

[11] X. Lu and R. Manduchi, "Detection and localization of curbs and stairways using stereo vision." in ICRA. IEEE, 2005, pp. 4648–4654.

[12] J. Siegemund, D. Pfeiffer, U. Franke, and W. Forstner, "Curb reconstruction using conditional random fields," in Intelligent Vehicles Symposium (IV), 2010 IEEE, June 2010, pp. 203–210.

[13] C. Fernandez, R. Izquierdo, D. Llorca, and M. Sotelo, "Road curb and lanes detection for autonomous driving on urban scenarios," in Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. IEEE, 2014, pp. 1964–1969.

[14] M. Kellner, M. E. Bouzouraa, and U. Hofmann, "Road curb detection based on different elevation mapping techniques," in Intelligent Vehicles Symposium Proceedings, 2014 IEEE. IEEE, 2014, pp. 1217–1224.

[15] F. Oniga, S. Nedevschi, and M. M. Meinecke, "Curb detection based on a multi-frame persistence map for urban driving scenarios," in Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on, Oct 2008, pp. 67–72.

[16] F. Oniga and S. Nedevschi, "Polynomial curb detection based on dense stereovision for driving assistance," in Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on. IEEE, 2010, pp. 1110–1115.

[17] ——, "Curb detection for driving assistance systems: A cubic spline-based approach," in Intelligent Vehicles Symposium (IV), 2011 IEEE, June 2011, pp. 945–950.

[18] M. Kellner, U. Hofmann, M. E. Bouzouraa, and N. Stephan, "Multi-cue, model-based detection and mapping of road curb features using stereo vision," in Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on. IEEE, 2015, pp. 1221–1228.

[19] F. Oniga and S. Nedevschi, "Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection," Vehicular Technology, IEEE Transactions on, vol. 59, no. 3, pp. 1172–1182, March 2010.

[20] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," Computer Vision–ACCV 2010, pp. 25–38, 2011.

[21] R. C. Gonzalez and R. E. Woods, Digital Image Processing (3rd Edition). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[22] G. Bradski, "The opencv library," Dr. Dobb's Journal of Software Tools, 2000.

[23] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in ICRA workshop on open source software, vol. 3, no. 3.2, 2009.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," International Journal of Robotics Research (IJRR), 2013.