Real Time Environment Representation in Driving Scenarios Based on Object Delimiters Extraction

Andrei Vatavu[1], Sergiu Nedevschi[1], and Florin Oniga[1],

[1] Computer Science Department, Technical University of Cluj-Napoca,
26-28 G. Baritiu Street, Cluj-Napoca, Romania
{Andrei.Vatavu, Sergiu.Nedevschi,Florin.Oniga }@cs.utcluj.ro

Abstract. In this paper we present and evaluate several methods for real-time environment representation by extracting object delimiters from the traffic scenes using a Dense Stereovision System. The delimiters detection is based on processing the information provided by a 3D classified occupancy grid obtained from the raw dense stereo information. One of the problems in representing the environment through the occupancy grid is a large volume of data. Therefore we propose a more compact 2.5D model by representing the environment as a set of polylines with associated features. Two approaches to extract object delimiters are presented: an improved contour tracing called 3A Tracing and a polyline extraction method through the radial scanning of the occupancy grid. We discuss the advantages and drawbacks for each of these methods.

Keywords: Environment Representation, Contour tracing, Border Scanning, Environment Perception, Polyline Extraction, Object Delimiters.

1 Introduction

In the context of in-vehicle navigation systems, the environment perception and its convenient representation is an important requirement [1]. The process of environment representation building has to be accurate and characterized by a low computational cost.

Usually, the Driving Assistance Applications detect the objects through 2D or 3D points grouping processes. The detected objects are represented by geometric primitives such as 2D bounding boxes [2] or 3D cuboids [3]. As an alternative approach, the objects may be represented by polylines. One of the advantages of the polyline based objects representation is the close approximation of the object contour by the polygonal model while having a number of vertices as small as possible. In the same time the polyline could inherit the type, position and height properties of the associated object.

The polyline object representation may lead to the creation of subsequent algorithms that are computationally fast due to the fact that only a small subset of points is employed.

The road feature identification through the object delimiters detection can be used in the unstructured environments as an alternative solution to the lane detection algorithms.

The object delimiters extraction is studied in some areas like mobile robots [4], [5], [6], [7], [8] or autonomous vehicle systems [9], [10], [11]. The polyline representation is very common in many algorithms, such as localization and mapping [6], [7], [8], [10] contour tracking [12] and path planning [10].

The polyline extraction methods differ by the nature of the information as well as by the sensors used for data acquisition process. Current systems use laser [4], [8], [9], [10], sonar [11], [7] or vision sensors [11].

Two main directions can be distinguished for the delimiters extraction:

- **The contour processing of already detected objects from the scene** [13];
- **The radial scanning of the environment**. This method is common for the systems based on sonar or laser sensors [4], [9].

A method for map representation as a set of line segments or polylines is described in [7]. An occupancy grid is created here from sonar information. The data is converted to a list of vertices using the Douglas Peucker line reduction algorithm.

In [8] a method that learns sets of polylines from laser range information is presented. The polylines are iteratively optimized using the Bayesian Information Criterion.

The polyline representation was chosen in [10] for terrain-aided localization of autonomous vehicle. The new range data obtained from the sensor are integrated into the polyline map by attaching line segments to the end of the polyline as the vehicle moves gradually along the tunnel.

In this paper we present and evaluate several methods for real-time environment representation by extracting object delimiters from the traffic scenes using a Dense Stereovision System [3]. The delimiters detection is based on processing the information provided by a 3D classified occupancy grid obtained from the raw dense stereo information. One of the problems in representing the environment through the occupancy grid is a large volume of data. Therefore we propose a more compact 2.5D model by representing the environment as a set of polylines with associated height features. We present two approaches to extract object delimiters:

- **The 3A Tracing.** The classical algorithm for contour tracing is improved by developing a new method named 3A Tracing Algorithm;
- **The radial scanning of the occupancy grid.** We have developed a Border Scanning method that is able to detect delimiters of complex objects taking into account the nature of information from the traffic scene (curb, object, and road).

A polyline map is generated as the result of the delimiters extraction process. Each polyline element inherits the type (object, curb), position and height properties of the associated objects from the occupancy grid.

In the next section, we describe the proposed Delimiters Extraction architecture. The delimiters detection approaches are presented in section 3. Experimental results are given in section 4, and section 5 concludes the paper with final remarks.

2 System Architecture

Our delimiters detection approaches have been conceived for an urban driving assistance system. We extended our Dense Stereo-Based Object Recognition System (DESBOR) by developing an Object Delimiters Detection component. A detailed description about the DESBOR system is presented in [3].

The Object Delimiters Detection system architecture consists in the following modules (see Fig. 1):

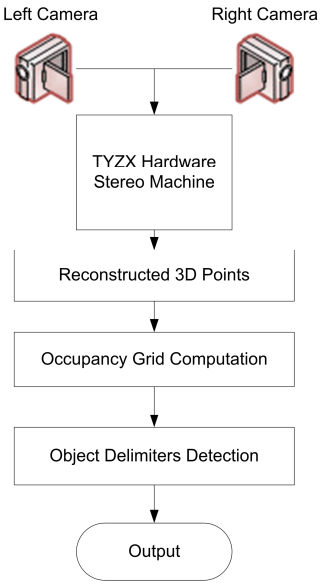


Fig. 1. System Architecture.

TYZX Hardware Stereo Machine The 3D reconstruction is performed by the "TYZX" hardware board [14].

Reconstructed 3D Points The reconstructed 3D points are used for the occupancy grid generation.

Occupancy Grid Computation The occupancy grid (see Fig. 2.c) represents a description of the scene, computed from the raw dense stereo information represented as a digital elevation map (see Fig. 2.b). The occupancy grid cells are classified into road, traffic isle and object cells. A detailed description about the occupancy grid computation is presented in [15]

Object Delimiters Detection The Object Delimiters detection uses the occupancy grid results as the input and generates a set of unstructured polygons approximated with the objects contour. The delimiters can be extracted from the occupancy grid through both 3A Tracing and Border Scanning algorithms.

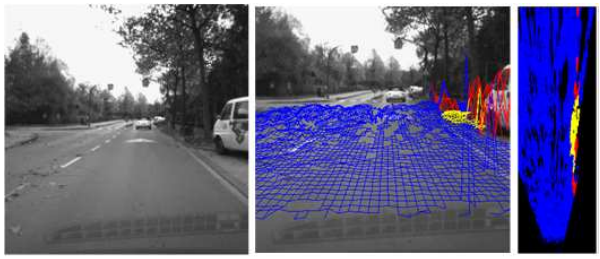


Fig. 2. The Occupancy Grid (c) is computed from the Elevation Map (b) of a scene (a). The occupancy grid cells are roughly classified (blue – road, yellow – traffic isle, red – obstacles)

Object Delimiters Detection Output A polyline map is generated as the result of delimiters extraction process. For each polyline element we keep the following information: a list of vertices, the delimiter type (object, curb), and the height of the object for which we apply the polyline extraction.

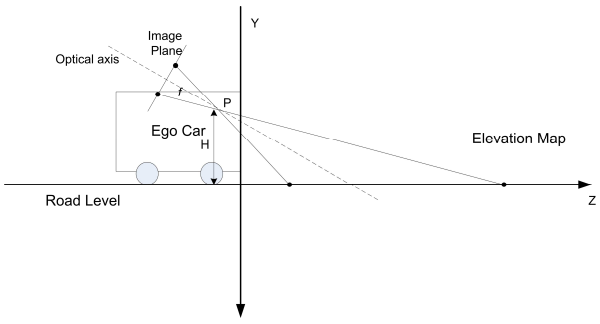


Fig. 3. The car coordinate system.

It must be noted that the car coordinate system coincide with the world coordinate system having its origin on the ground in front of the car (see Fig. 3). The position and orientation of the stereo cameras are determined by the absolute extrinsic parameters [16].

3 Object Delimiters Extraction Methods

A set of steps have been identified for the delimiters extraction:

Step 1: Object Labeling. In this step each object from the occupancy grid is labeled with a unique identifier.

Step 2: The contour extraction. We compute the contours of the non-drivable blobs (objects, traffic isles) from the occupancy grid. Each contour point will represent a single cell in the grid map.

Step 3: The polygonal approximation. Given a curve C we will find a polygon that closely approximates C while having as small a number of vertices as possible.

Next, we will present several algorithms developed by us for delimiters extraction. All these methods have in common the 1st and 3rd step. The 2nd step is different in each case. We have used two main approaches for the contour extraction:

1) The Contour Tracing for a given object - once an object cell has been identified, contour tracing is performed starting from this point, adding each traversed cell to the current contour. In this paper we present an improved version of contour tracing, the 3A Tracing Algorithm.

2) The Border Scanning – a radial scanning is performed with a given radial step, traversing the interest zone and accumulating the contour points at the same time. The main difference of this approach is that we scan only the visible parts from the ego-car position. Two main improvements of the Border Scanning method are discussed: the Border Scanning using a variable step, and the Combined Border Scanning, taking into account the occupancy grid blob's nature (traffic isles, obstacles).

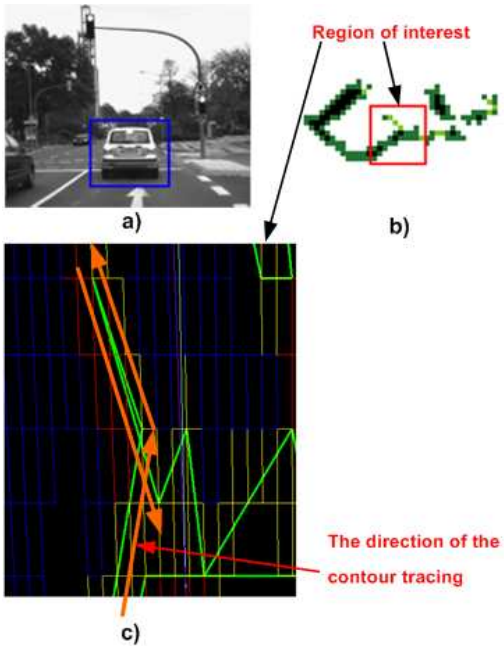


Fig. 4. Contour tracing of the care points (b) from the scene (a). There are cases when two polygonal segments can intersect each other (c), after the polygonal approximation of the car contour.

3.1 The 3A Tracing Algorithm

The classical contour tracing algorithm collects the contour points of an object by traversing the object boundary.

A disadvantage of the classical algorithm is that there are cases when the same delimiter point can be passed many times. This may lead to the incorrect representation, after the contour approximation step (see Fig. 4).

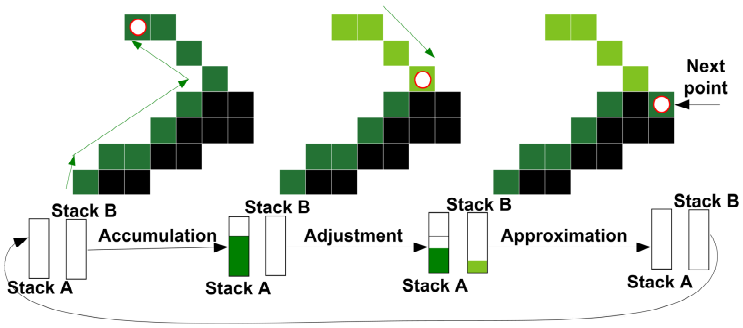


Fig. 5. 3A Tracing Algorithm. In the *Accumulation* phase, all traversed points are pushed onto the *Stack A*. In the *Adjustment* stage, the already passed points are extracted form the *Stack A* and pushed onto the *Stack B*. *Polygonal approximation* is applied in the last step of algorithm.

To avoid this problem we have developed an extended contour tracing algorithm named 3A Tracing. In this method we use two stacks, Stack A and Stack B. The name 3A Tracing comes from the next three main phases (see Fig. 5):

Phase 1: Accumulation. The tracing is made analogue to the Contour Tracing algorithm. All accumulated points are pushed onto the stack A. The traversed points are marked with a flag in order to know whether they were traversed or not at least one time. Once we found a terminal point (from which the tracing is made in the inverse sense) we pass to the 2nd phase of the algorithm.

Phase 2: Adjustment. In this phase the tracing continues in the inverse sense by extracting already passed points (drawn with light green) from the Stack A, and pushing them onto the Stack B. The Adjustment is repeated until we reach a contour point that has not passed yet. Once the new contour point is found we pass to the 3rd phase of the algorithm.

Phase 3: Approximation. Polygonal approximation is applied to each of the two stacks. After the polygonal approximation process the two stacks will be cleared and the algorithm is repeated from the Phase 1.

The algorithm stops when the start point is reached once again.

Although the 3A Tracing algorithm eliminates some particular cases in which two polygonal segments may intersect, like in the Contour Tracing, it works only on the connected components. Therefore this method does not take into account the cases of more complex objects, when a single obstacle is represented as many disjoint patches. Therefore we have elaborated an extraction method through the radial scanning of the Elevation Map.

3.2 The Border Scanner Algorithm

The Border Scanner algorithm performs a radial scanning with a given radial step. The scanning axis moves in the radial direction, having a fixed center at the Ego Car position. The scanning process is made into the limits of *Q_from* and *Q_to* angles, thus only the interest area are scanned, where the delimiters can be detected (see Fig. 6). Having a radial axis with a *Qrad* slope, *Q_from* < *Qrad* < *Q_to*, we try to find the nearest point from the Ego Car situated on this axis. In this way, all subsequent points will be accumulated into a *Contour List*, moving the scanning axis in the radial direction. At each radial step we verify that a new object has been reached. If a new label has been found then the polygonal approximation on the *Contour List* points is performed. The list will be cleared, and the algorithm will be continued finding a new polygon.

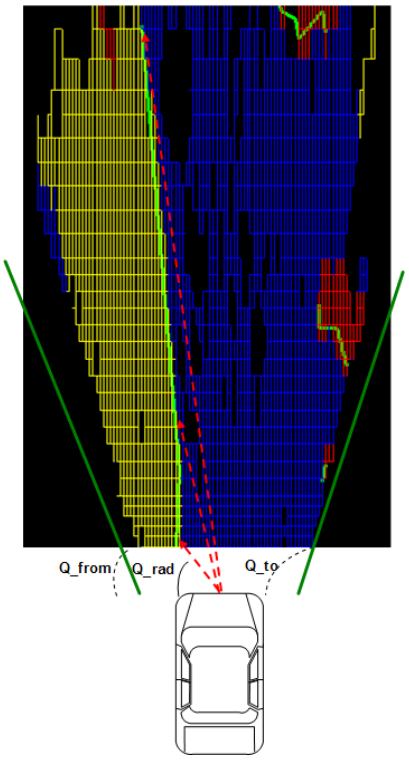


Fig. 6. Border Scanning on the Occupancy Grid Points.

Advantages: The obtained results are more close to the real obstacle delimiters from the scene. The problem of the complex objects presented in the case of Contour

Tracing algorithms is eliminated. Therefore many disjoint patches that belong to the same object can be enveloped by a single delimiter.

Disadvantages: A little obstacle (noise present in the occupancy grid) can occlude a great part from the scene, if this obstacle is too near to the Ego-Car. The scanning is influenced by the presence of such false obstacles.

The Border Scanning Algorithm Using Variable Step. Having a constant radial step, the detected pixel density will decrease with the depth distance. The distance between two consecutive detected pixels is greater at the far depths. The idea is that some important information about the delimiters can be lost at the far distances.

A good solution is to use a scanning method with a variable step, thus the radial step should be adapted with the distance.

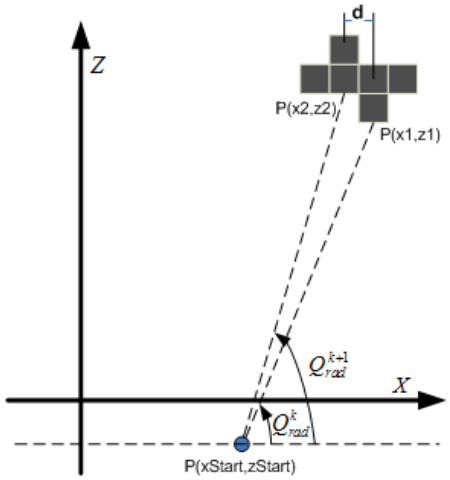


Fig. 7. Radial angle estimation for the next step in the Variable Step Border Scanning approach.

If we have a point $P_1(x_1, z_1)$ of a given object and a radial axis containing the point P_1 with a radial angle Q^k_{rad} at the k step, then we estimate the radial angle at the $k+1$ step (see Fig. 7):

$$Q^{k+1}_{rad} = \arctan\left(\frac{z_1 - z_{Start}}{x_1 - x_{Start} - d}\right). \quad (1)$$

Where:

- x_1, z_1 are the coordinates of the P_1 point;;
- x_{Start}, z_{Start} are the ego-car point coordinates.
- d is considered the distance between any two adjacent points.

However, there are situations when no object point can be reached on the current scanning axis. Therefore we cannot estimate the radial angle for the next step, because we don't know the distance of the current object point from the Ego-Car. In this case, like in the simple Border Scanning method, we use a fixed step, until a new object point will be found.

The Combined Border Scanning. We know that the occupancy grid cells are classified into obstacles (cars, pedestrians etc.) or traffic isles (road-parallel patches). If we take into account only the first nearest point from the car, many relevant objects delimiters may be omitted. For example, the first obstacle from the car can be a curb. In this case, we are interested not only in the curb delimiters but also in the delimiters above the curb or behind the curb. Therefore we extended our Border Scanning algorithm by developing a method that takes into consideration the obstacle's nature making a decision based on two types of information "What have we found?" and "What we have to find?". The algorithm consists in two passes: one for the object delimiters detection, and second for the traffic isles delimiters detection.

In the Table 1 is presented the returned result when we want to find a delimiter taking in account the point type we have found.

Table 1. The Combined Border Scanning method. The result is returned, taking into consideration the found point type.

Delimiter's type we want to find	Point Type we have found	Returned result
OBJECT	OBJECT	FOUND
CURB	OBJECT	NOT FOUND
CURB	CURB	FOUND

4 Experimental Results

For the experimental results we have tested a set of 15 scenarios from the urban traffic environment using a 2.66GHz Intel Core 2 Duo Computer with 2GB of RAM.

Fig. 8 shows a comparative result between the Contour Tracing and 3A Tracing algorithms, using an approximation error of two points. One can notice that the polygonal segment intersection in the case of classical contour tracing algorithm (see Fig. 8.b) was eliminated by applying the 3A Tracing algorithm (see Fig. 8.c).

The difference between the result of delimiters detection in the case of Simple border Scanner and Combined Border Scanner algorithms is presented in the Fig. 9. It can be observed that in the case of Combined Border Scanner (see Fig. 9.b) the side fence's delimiter is detected in spite of his position behind the curb (see Fig. 9.c).

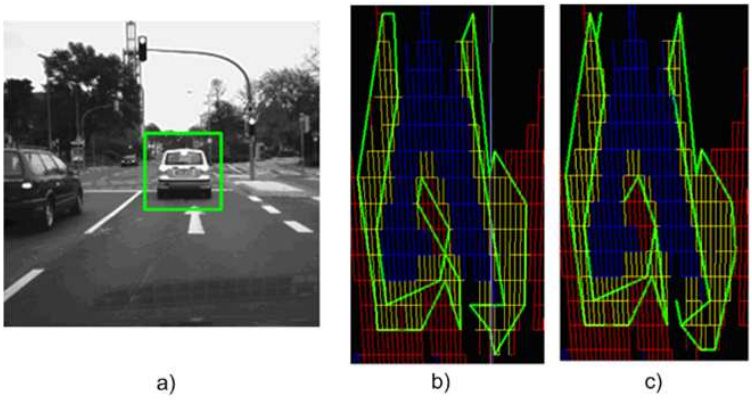


Fig. 8. Delimiters detection through the *Contour Tracing* algorithm (b) and *3A Tracing* algorithm (c). The detection is performed on the occupancy grid computed from the scene (a).

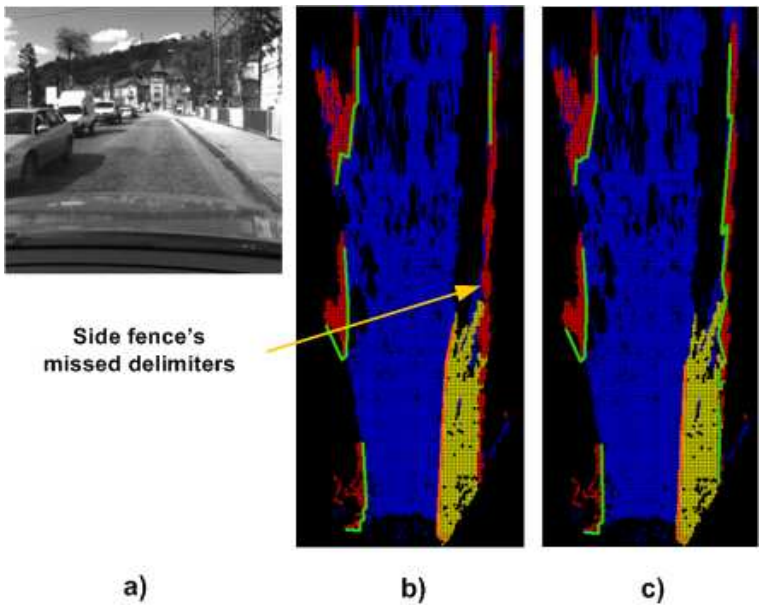


Fig. 9. Border scanning of a scene (a). The side fence's delimiters are missed in the case of the *Simple Border Scanning* (b) and have been detected in the case of the *Combined Border Scanning* algorithm (c).

In the Table 2 the results from the Variable Step Border Scanner and Fixed Step Border Scanner are computed for the same driving scene. It can be observed that the number of detected points is greater in the case of Variable Step Border Scanner

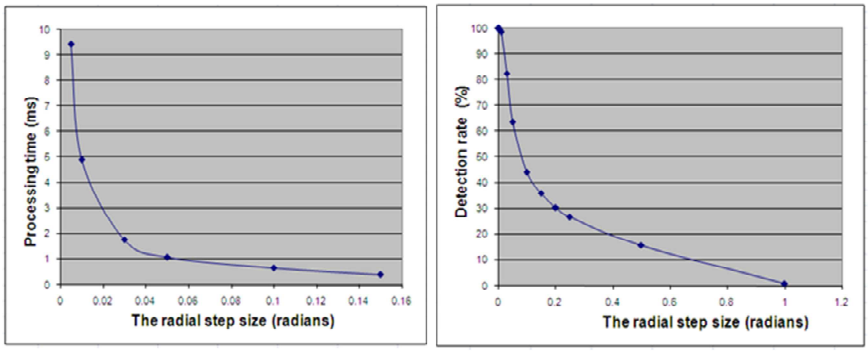
algorithm, thereby 11466 points, which means 28 detected points per frame in comparison with 22 detected points per frame in the case of Fixed Step Border Scanner algorithm.

Table 2. Fixed Step Border Scanner vs. Variable Step Border Scanner.

	Fixed Step Border Scanner	Variable Step Border Scanner
Number of Frames	406	406
Detected points	9058	11466
The radial step (radians)	0.01	variable
Points per Frames	22	28
Average processing time per frame	4 ms	5 ms

The average extraction time using the 3A Tracing algorithm is about 0.7ms per frame and depends on the angular resolution in the case of Border Scanner approach.

Fig. 10a shows how the radial step size variation affects the system response time using the Combined Border Scanning approach.



a) b)

Fig. 10. a) The processing time vs. the radial step size. b) The detection rate vs. the radial step size.

Fig. 10b is a diagram that shows the impact of radial step size on delimiters detection rate using the border scanner method. We can observe that, with a higher radial step size we obtain an increase in processing time while the detection rate decreases. The solution is a tradeoff between the system processing time and detection rate.

Fig. 11 presents results for various traffic scenes using the Combined Border Scanning method. For the border scanning algorithm with a radial step of 0.01 radians the average processing time is about 5ms and the delimiters detection rate is 98.66%.

Fig. 11. Object delimiters detection through the Combined Border Scanning algorithm for various traffic scenes. The delimiters are projected onto the Left Image and are represented as grids labeled as Traffic Isles (orange) or Objects (light green). The grid height is the same as the enveloped object by the current delimiter.

5 Conclusions

In this paper we present and evaluate several methods for real-time environment representation through the object delimiter extraction and characterization from dense stereovision images. The delimiters detection is based on processing the information provided by a 3D classified occupancy grid obtained from the raw dense stereo information. The result is a more compact 2.5D model for representing the environment, as a set of polylines. Each polyline element inherits the type (object, curb), position and height properties of the associated object from the occupancy grid.

We have developed an improved Contour Tracing method named 3A Tracing algorithm that eliminates the situation when two polygonal segments can intersect each other.

Another approach presented in this paper is the polyline extraction through the radial scanning of the occupancy grid. Although the tracing approach is more computationally-efficient, the results provided by the Border Scanner algorithm are more appropriate for detecting the real obstacle delimiters from the scene. The algorithm is able to extract only the visible area from the ego-vehicle since the occluded points do not offer relevant information. Using the Border Scanner algorithm, our system is fast and achieves a high rate of detection: 98.66%.

As future work we propose to extend our approaches by using temporal filtering of delimiter points in order to improve the polyline representation accuracy. The final

goal of our research is to develop a system for complex environments that would achieve high performances with respect to accuracy, confidence and real-time capabilities.

References

1. Pijpers, M.: Sensors in adas. Universiteit Twente (2007)
2. Dellaert, F., Thorpe, C.: Robust Car Tracking using Kalman filtering and Bayesian templates. In: proceedings of Conference on Intelligent Transportation Systems, vol. 3207, pp. 72--83. (1997)
3. Nedevschi, S., Danescu, R., Marita, T., Oniga, F., Pocol, C., Sobol, S., Tomiuc, C. Vancea, C., Meinecke, M. M., Graf, T., To, T. B., Obojski, M. A.: A sensor for urban driving assistance systems based on dense stereovision. In: Intelligent Vehicles 2007, pp. 278--286, Istanbul (2007)
4. Harati, A., Siegwart, R.: A new approach to segmentation of 2d range scans into linear regions. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA (2007)
5. Magin, G., Russ, A.: Supporting real-time update of an environment representation for autonomous mobile robots real-time systems. In: EuroMicro Workshop on Real-Time Systems, pp. 34--39 (1994)
6. Joshi, R. R.: Novel metrics for map-matching in in-vehicle navigation system. In: IEEE Intelligent Vehicle Symposium, Vol. 1, pp. 36--43 (2002)
7. Laviers, K. R., Peterson, G. L.: Cognitive robot mapping with polylines and an absolute space representation. In: IEEE International Conference on Robotics and Automation, pp. 3771--3776. Hilton New Orleans Riverside, New Orleans, LA, USA (2004)
8. Veeck M., Burgard, W.: Learning polyline maps from range scan data acquired with mobile robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2004)
9. Kolski, S., Ferguson, D., Bellino, M., Siegwart, R.: Autonomous driving in structured and unstructured environments. In: IEEE Intelligent Vehicles Symposium (2006)
10. Madhavan, R.: Terrain aided localization of autonomous vehicles. In: Symposium on Automation and Robotics in Construction, Gaithersburg (2002)
11. Goncalves, A., Godinho, A., Sequeira, J.: Lowcost sensing for autonomous car driving in highways. In: ICINCO2007 - 4th International Conference on Informatics in Control, Automation and Robotics, Angers, France (2007)
12. Prakash, S., Thomas, S.: Contour tracking with condensation/stochastic search. In: Dept. of CSE. IIT Kanpur (2007)
13. Gonzales, R. C., Woods, R. E.: Digital Image Processing. Addison Wesley, second edition (2002)
14. Woodill, J. I., Gordon, G., Buck, R.: Tyzx deepsea high speed stereo vision system. In: IEEE Computer Society Workshop on Real Time 3-D Sensors and Their Use, Conference on Computer Vision and Pattern Recognition (2004)
15. Oniga, F., Nedevschi, S., Meinecke, M. M., To, T. B.: Road surface and obstacle detection based on elevation maps from dense stereo. In: IEEE Intelligent Transportation Systems Conference, pp. 859--865, Seattle, WA, (2007)
16. Marita, T., Oniga, F., Nedevschi, S., Graf, T., Schmidt, R: Camera calibration method for far range stereovision sensors used in vehicles. In: IEEE Intelligent Vehicles Symposium (IV2006), pp. 356--363, Tokyo, Japan (2006)