Obstacle Detection Using Dynamic Particle-Based Occupancy Grids

Radu Gabriel Danescu
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Radu.Danescu@cs.utcluj.ro

***Abstract*—Due to the complex nature of the driving environment, obstacle tracking systems are required to rely on intermediate dynamic information, before the obstacle is fully reconstructed. This paper presents an obstacle estimation system which uses the advantages of a particle-based occupancy grid tracking solution. The initial measurement data is a raw occupancy map extracted from dense stereovision-derived elevation maps. The occupancy grid tracking system is able to use the raw occupancy data to derive a filtered occupancy probability for each grid cell, along with dynamic information. The process of grouping the grid cells into individual obstacles takes advantage of the dynamic grid information to discriminate between nearby objects that have different speeds, and is able to produce oriented cuboids, having position, size and speed, without the need of an additional tracking step.**

Keywords- tracking; occupancy grid; obstacle detection

I. INTRODUCTION

Any autonomous system, or system that performs autonomous functions related to navigation, has to be aware of the existence of obstacles. When such a system is a driving assistance system, the obstacle problem becomes critical, as the tolerance for failure is zero. In order to avoid or to follow an obstacle, it is not enough to know where the obstacle is, but also to know where the obstacle will be in near future. For this reason we need to perceive the obstacles as dynamic entities.

Most of the obstacles in the driving environment can be modelled as cuboids having position, size, orientation and speed. The parameters of these cuboids can be tracked using any model-based tracker, and a variety of sensorial information. When the available measurement data does not contain dynamic information, but is a series of position measurements arriving at discrete time intervals, the estimation of the obstacle speed must rely only on the quality of the tracker, and especially on the quality of data association (matching between the parts of the target to be tracked and the measurement information). A system that tries to achieve successful measurement-association results in the presence of incomplete measurements (parts of the object not visible, sensor errors, etc) is presented in [8]. Unfortunately, the complexity of the driving environment, especially in the case of urban traffic, and the large degrees of freedom associated to the need of tracking heterogeneous obstacles at any orientation, leads to the need of having some kind of intermediate dynamic information independently of the model-based tracking process. Any obstacle tracking system can be greatly improved if the dynamic properties of the environment can be estimated independently from the choice of object representation.

In order to achieve the goal of extracting the speed independently from object model, intermediate tracking solutions are devised. Such solutions can directly track 3D points (the 6D vision technique, presented in [1]), compact dynamic obstacle primitives such as the stixels [2], or they can use track the occupancy and speed of a cell in the map, such as in the case of occupancy grids.

The dynamic occupancy grid is a good choice for the driving related dynamic environment, as it is capable of concisely describing the relevant aspects while maintaining a decent level of computation complexity. Maybe one of the first uses of occupancy grids, under the name of probabilistic local maps, is presented by Elfes in [3], in the context of sonar based robot navigation, and the probability inference mechanism for handling the uncertainty of range sensors in computing the probability of each cell's occupancy state is presented in [4]. The initial occupancy grids, such as those presented in [3] and [4], are simple 2D maps of the environment, each cell describing the probability of it being occupied or free. By adding the speed factor in the environment estimation, the complexity increases significantly, as the cells are now strongly interconnected. The work of Coué et al, presented in [5], uses a 4D occupancy grid, where each cell has a position and two speed components along each axis. By estimating the occupancy of each cell in the 4D grid, the speeds for the classical cells in the 2D grid can be computed. Another solution for the representation of speeds is presented by Chen et al, in [6]. Instead of having a 4D grid, this solution comes back to 2D, but uses for each cell a distribution of speeds, in the form of a histogram.

Several object tracking solutions are tailored to use the benefits of available intermediate dynamic information. For example, a solution based on grouping dynamic stixels is presented in [9], and a solution based on optical flow combined with stereovision is presented in [10].

This paper presents an obstacle estimation system which uses the advantages of a particle-based occupancy grid

tracking solution. The initial measurement data is a raw occupancy map extracted from dense stereovision-derived elevation maps. The occupancy grid tracking system is able to use the raw occupancy data to derive a filtered occupancy probability for each grid cell, along with dynamic information. The process of grouping the grid cells into individual obstacles takes advantage of the dynamic grid information to discriminate between nearby objects that have different speeds, and is able to produce oriented cuboids, having position, size and speed, without the need of an additional tracking step.

II. PARTICLE OCCUPANCY GRID

The dynamic occupancy grid that we use as an intermediate environment representation depicts the obstacles as composed of particles having individual position and speed. Each particle is assigned to a discrete cell in a grid. The occupancy probability of each grid cell is described by the number of particles in that cell, and the particles have a dual nature – they describe occupancy hypotheses, as in the particle filtering algorithms such as CONDENSATION [7], but can also be regarded as physical building blocks of our modelled world. The grid tracking algorithm is particle-oriented, not cell oriented. The particles can migrate from cell to cell depending on their motion model and motion parameters, but they are also created and destroyed using a weighting-resampling mechanism similar to the one described in [7].

The world is represented by a 2D grid, mapping the bird-eye view 3D space into discrete 20 cm x 20 cm cells. The size of the grid is 250 rows x 120 columns, corresponding to a scene size of 50x24 meters. The aim of the grid tracking algorithm is to estimate the occupancy probability of each grid cell, and the speed components on each axis.

Considering a coordinate system where the z axis points towards the direction of the ego-vehicle, and the x axis points to the right, the obstacles in the world model are represented by a set of particles:

$$S = \{p_i \mid p_i = (c_i, r_i, vc_i, vr_i, a_i), i = 1...N_S\}, \qquad (1)$$

each particle i having a position in the grid, described by the row r_i (a discrete value of the distance in the 3D world z) and the column c_i (discrete value of the lateral position x), and a speed, described by the speed components vc_i and vr_i. An additional parameter, a_i, describes the age of the particle, since its creation. The purpose of this parameter is to facilitate the validation and the speed estimation process, as only particles that survive in the field for several frames are taken into consideration. The total number of particles in the scene N_S is not fixed, but dependent on the occupancy degree of the scene, that is, the number of obstacle cells in the real world. Having the population of particles in place, the occupancy probability of a cell C is estimated as the ratio between the number of particles whose position coincides with the position of the cell C and the total number of particles allowed for a single cell, N_C.

$$P_O(C) = \frac{|\{p_i \in S \mid r_i = r_c, c_i = c_c\}|}{N_C} \qquad (2)$$

The number of allowed particles per cell N_C is a constant of the system. In setting its value, a trade-off between accuracy and time performance should be considered. A large number means that on a single cell multiple speed hypotheses can be maintained, and therefore the tracker can have a better speed estimation, and can handle fast moving objects better. However, the total number of particles in the scene will be directly proportional with N_C, and therefore the speed of the algorithm will decrease.

The speed of a grid cell can be estimated as the average speed of its associated particles, if we assume that only one obstacle is present in that cell. Of course, the particle population can handle the situation when multiple obstacles, having different speeds, share the same cell, and in this case the speed estimate of the cell must be computed by clustering.

$$(vc_C, vr_C) = \frac{\sum_{p_i \in S, x_i = x_c, z_i = z_c} (vc_i, vr_i)}{|\{p_i \in S \mid r_i = r_c, c_i = c_c\}|} \qquad (3)$$

Thus, the population of particles is sufficiently representative for the probability density of occupancy and speed for the whole grid. Multiple speed hypotheses can be maintained simultaneously for a single cell, and the occupancy uncertainty is represented by the varying number of particles associated to the cell. The tracking algorithm can now be defined: using the measurement information in the form of elevation maps, it will create, update and destroy particles such that they accurately represent the real world.

The first step of the algorithm is the *prediction*, which is applied to each particle in the set. The positions of the particles are altered according to their speed, and to the motion parameters of the ego vehicle (see figure 3). Also, a random amount is added to the position and speed of each particle, for the effect of stochastic diffusion. The second step is the *processing of measurement* information. This step is based on the raw occupancy cells provided by dense stereo processing, and provides the measurement model for each cell.

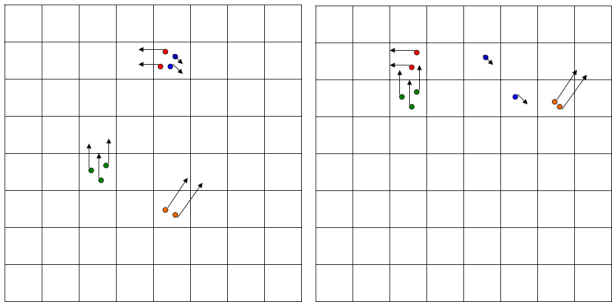

Figure 1. Migration of particles from one cell to another, as prediction is applied.

The measurement model information is used to *weight* the particles, and *resample* them in the same step (see figure 4). By weighting and resampling, the particles in a cell can be multiplied or reduced. The final step is to estimate the occupancy and speeds for each cell. A more detailed description of the particle grid tracking algorithm is given in [11] and [12].

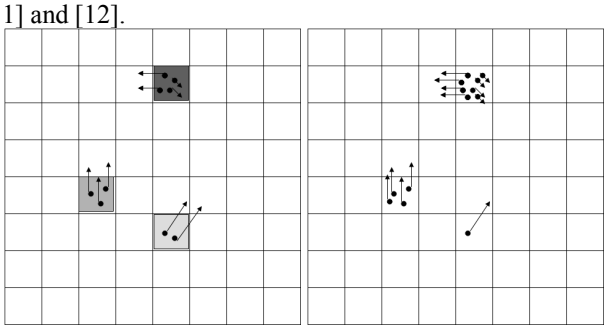

Figure 2. Weighting and resampling. The weight of the occupied hypothesis is encoded in the darkness of the cell of the left grid. In the right grid, the effect of resampling is shown, as particles are multiplied or deleted.

A typical grid result is described in figure 3. The intensity of the grid cell is proportional to the occupancy value, and the speed is encoded in the saturation and hue of the colour. Figure 3 also shows a colour map for deciphering the colour grid information, and a projection of the grid results on the original perspective image, for comparison.

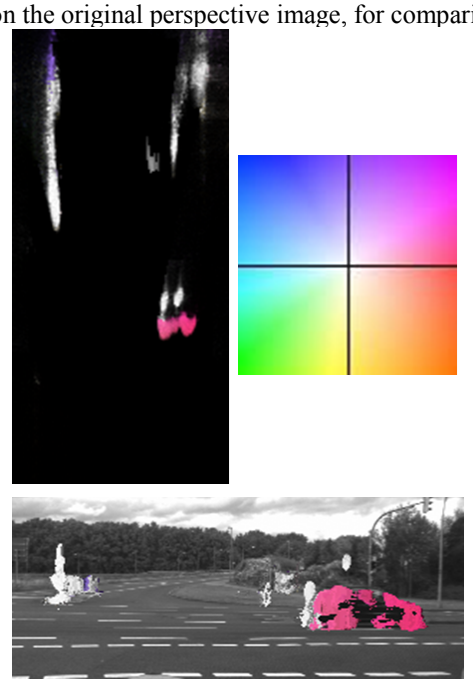

Figure 3. Typical dynamic occupancy grid result.

III. IDENTIFYING CONNECTED COMPONENTS

After each grid cell receives an occupancy probability and a speed estimation, the next step is to use these results for extracting the individual objects in the scene. For this purpose we'll use a labelling algorithm that is able to take advantage of the dynamic properties of the grid.

Algorithm Labelling
Input: Cell – grid of cells, with occupancy value and speed
Output: Label – grid of labels
Uses Queue – queue of pairs of row, column coordinates
While ((r,c) = *FindUnlabeledOccupiedCell()* != NULL)
CurrentLabel = *MakeNewLabel()*
Label (r, c) = CurrentLabel
Queue.*Insert*((r,c))
Area (CurrentLabel)=1
MinRow(CurrentLabel)=r;
MaxRow(CurrentLabel)=r;
MinColumn(CurrentLabel)=c;
MaxColumn(CurrentLanel)=c;
While Not Queue.*IsEmpty()*
(rr, cc) = Queue.*Remove()*
Update MinRow(CurrentLabel)
Update MaxRow(CurrentLabel)
Update MinColumn(CurrentLabel)
Update MaxColumn(CurrentLabel)
Increment Area(CurrentLabel)
If Not *AreaRatioCheckOk()*
Queue.*ForceEmpty*()
Break()
End if
For Each (rri, cci) **in** *Neighborhood*(rr, cc)
If *UnlabeledOccupiedCell*(rri, cci)
If *Compatible* (Cell(rr, cc), Cell(rri, cci))
Label(rri, cci)= CurrentLabel
Queue.*Insert* (Point(rri, cci))
End if
End if
End For
End While
End While

The labelling algorithm is based on the generic solution of breadth first exploration of graphs, as the occupied cells can be regarded as nodes in the graph and the neighbourhood relationship between them as edges. In the remaining of this section we'll describe the main elements of the cell grouping algorithm.

The input to the labelling algorithm is the occupancy grid, an array of *Cell*-s, indexed by the row and column. Each cell has the following properties: *occupancy*, ranging from 0 to 1, expressing the probability that the cell is occupied, *speed*, the absolute value of the estimate speed for the cell, and *orientation*, which describes the orientation of the estimated speed vector.

The output of the algorithm will be the 2D array *Label*, having a unique identifier for each connected component, which hopefully will correspond to a real-world object. Initially, this array is initialized with zero.

The algorithm starts by finding occupied cells that have no label assigned to their position yet. A cell is considered to be occupied if the *occupancy* value is above a fixed threshold of 0.5. A new label is generated, and assigned to this initial

point. A set of parameters are initialized at this step: the *area*, which is the number of cells assigned to a connected component, and the extreme coordinates of this component, minimum row, minimum column, maximum row and maximum column. These parameters will be updated as the labelling algorithm will gradually cover the whole object.

The main labelling loop will execute as long as the queue is not empty. A coordinate pair (which corresponds to a cell already labeled) is extracted from the queue, and the points in its *Neighbourhood* are analyzed. The size of the neighbourhood is defined by the uncertainty of the stereo measurement, as the neighbourhood is a rectangle of size $2\sigma_{row}$ by $2\sigma_{column}$, centred in the current point. The row and column uncertainty are computed from the distance uncertainty of the stereo measurement, divided by the grid cell size *DX* and *DZ*. These values are computed offline, at the initialization phase, for each cell in the grid.

$$\sigma_{row} = \frac{\sigma_z}{DZ} \tag{4}$$

$$\sigma_{column} = \frac{\sigma_x}{DX} \tag{5}$$

The distance uncertainty of a stereovision system can be approximated as:

$$\sigma_z = \frac{z^2 \sigma_d}{bf} \tag{6}$$

In the previous equation, z denotes the distance (in the real world coordinates), b is the baseline of the stereo system, f is the focal distance in pixels, and σ_d is the error in disparity computation (usually about 0.25 pixels, for a good stereo reconstruction engine). The error in lateral positioning (usually much smaller than the error in z), can be derived from the distance error. This error depends on the lateral position x (in the real world coordinates) and the distance z.

$$\sigma_x = \frac{x\sigma_z}{z} \tag{7}$$

Each neighbour (rri, cci) of the current position is tested for compatibility before it is added to the connected component. An ordinary labelling algorithm takes into account only the vicinity relations, but this can sometimes lead to false connection between different, but closely positioned objects, as shown in figure 4.

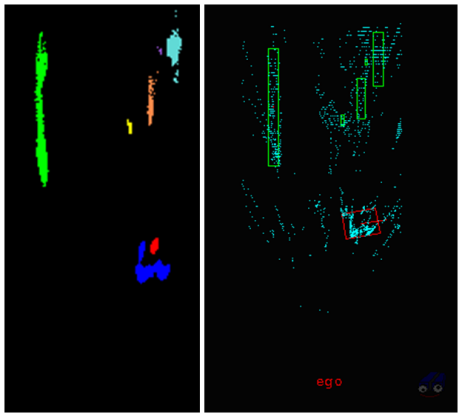

Figure 4. Vicinity-only labelling, without taking into account the dynamic cell information. Left – label grid, right – resulted 3D objects.

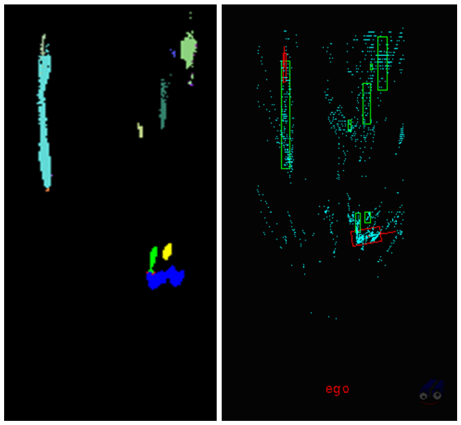

Figure 5. Labelling using the speed compatibility criteria.

The compatibility test is used exactly for the purpose of preventing such false associations. The neighbours of the current cell will receive the current label only if the speed characteristics of these cells are similar. Two cells are said to be compatible if:

- The difference in the orientation of the speed vectors in the two cells is less than 30 degrees.
- The difference in speed vector magnitudes is less than 30% of the value of the largest magnitude of the two cells.

Another problem of the classical labelling approach for object extraction is that the algorithm will connect anything as long as the compatibility is valid, and sometimes the resulted objects have a considerable size. Also, most of the huge objects are static, and therefore their orientation is difficult to estimate (extracting the orientation from shape is not very robust). The most problematic case is the one described in figure 6, when a non-oriented cuboid will be completely unsuited for the large structure.

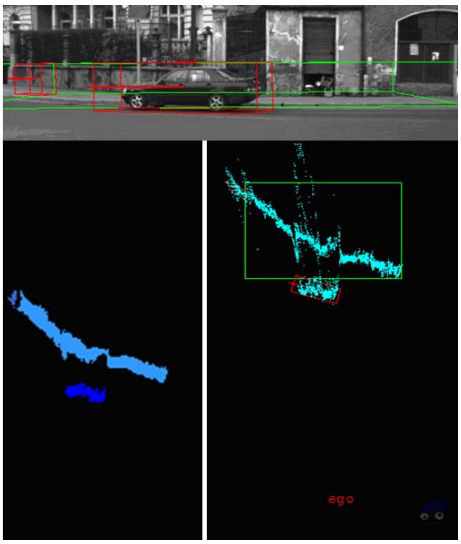

Figure 6. Massive static objects

In order to prevent the generation of such deformed objects, we have designed the *AreaRatioCheck* test. The following coefficient is computed any time a cell position is extracted from the queue, and the area and the coordinate limits are updated:

$$\rho = \frac{A}{(MaxRow - MinRow)(MaxColumn - MinColumn)} \quad (8)$$

When the object's length or width is above four meters, the area ratio is tested against a 0.5 threshold. This condition means that the labelled object should fill its non-oriented box at least 50 %. If the test fails, the labelling process is interrupted by forced emptying of the queue, and then resumed with a new label. The effect is shown in figure 7, where the large static object is broken into smaller pieces which depict the real world geometry more accurately.

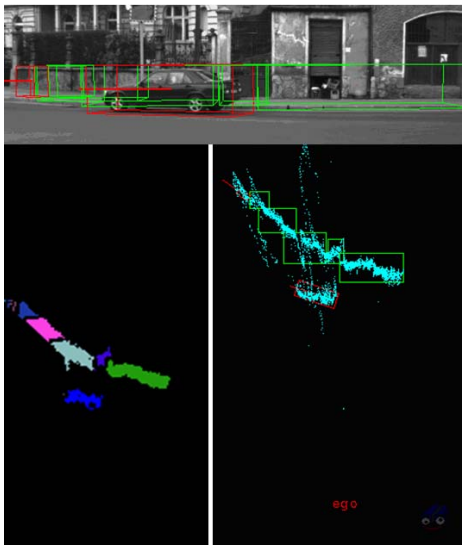

Figure 7. Large objects are split into smaller pieces

IV. Extraction of Object Properties

After the labelling process is finished, each object is identified by its label L. The first property that is computed is the speed vector $\vec{v}_L$. In order to estimate the speed of an object, we'll perform a weighted average of the speeds of the object's cells, and the occupancy probability will fulfil the role of weight.

$$\vec{v}_L = \frac{\sum_{r,c} P_O(r,c)\vec{v}(r,c)(Label(r,c) = L)}{\sum_{r,c} P_O(r,c)(Label(r,c) = L)} \quad (9)$$

The magnitude and the orientation of the speed vector are computed, and a static versus dynamic discrimination of the objects is performed (all objects having the speed vector magnitude above 1.5 m/s are dynamic). In figures 4-9, the static objects are depicted with green, and the dynamic objects with red. The orientation of the static objects is not computed, and their extremities are given by the maximum and minimum row and column numbers.

The orientation of the dynamic objects is given by the orientation of their speed vectors. The width and the length of the dynamic objects are computed by computing the distances of the object's labelled grid positions from the axis of orientation, already known.

V. Results

The system is able to identify the objects in the scene, computing their position and their scene, even when they are close together or they are temporarily occluded. An example is given in figure 8. A vehicle comes from our left, then turns left and proceeds to exit the scene. During this manoeuvre it occludes the static object near its left side, but does not become joined with this structure due to the speed-sensitive nature of the cell clustering algorithm. We can see how the occupancy becomes diffused as the object is occluded by a large truck, which then again occludes the static objects on the right.

We have also performed tests for speed and orientation estimation accuracy, on sequences acquired in controlled scenarios, with known target speed and orientation. We have performed four tests, with the same orientation, -45 degrees, but different speeds, 30 km/h, 40 km/h, 50 km/h, 60 km/h. The results that were evaluated are the estimated speed and orientation of the 3D cuboid resulted from clustering the occupied grid cells. These results are compared to the ground truth.

The speed estimation accuracy results are shown in Table 1, and the orientation results are shown in Table 2. The system was able to quickly identify the target cuboid, and once the obstacle became reasonably visible the estimated speed and orientation became accurate and stable, as shown by the Mean Absolute Error from the ground truth, and from the standard deviation of the results. The tests with the speeds near the urban speed limit show that even when using a medium to low accuracy primary sensor, at a quite low

frame rate of 10 frames per second, the system is able to accurately estimate the speed and orientation of the obstacles.

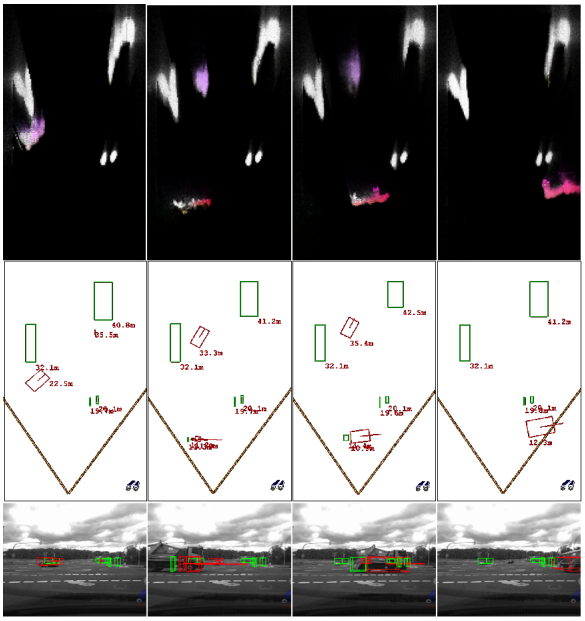

Figure 8. Sample results

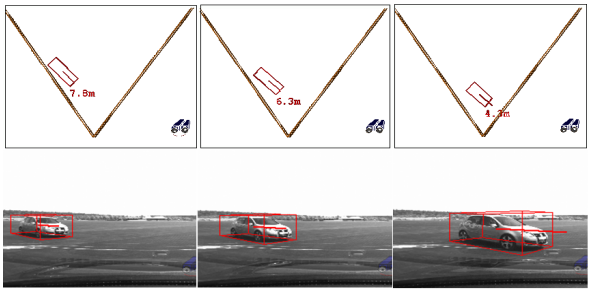

Figure 9. Test sequence for speed accuracy estimation

TABLE 1
NUMERICAL RESULTS – SPEED ESTIMATION ACCURACY

Speed of target	Estimation MAE	Estimation STDEV
30 km/h	0.9016	0.9731
40 km/h	1.0184	0.9730
50 km/h	2.4989	2.3370
60 km/h	2.1279	1.3858

TABLE 2
NUMERICAL RESULTS – ORIENTATION ESTIMATION ACCURACY

Speed of target	Estimation MAE	Estimation STDEV
30 km/h	0.9728	0.8376
40 km/h	1.0321	0.8616
50 km/h	0.4695	0.2659
60 km/h	0.9343	0.6739

VI. CONCLUSIONS

This paper presents a method for extracting the parameters of dynamic objects, in traffic scenarios, based on the use of a particle-based dynamic occupancy grid. Based on the permanently updated occupancy and speed information of the grid cells, a grouping algorithm is able to extract individual objects, to avoid merging close objects when their motion parameters are different, and to extract the proper speed and orientation for these objects. The use of a dynamic occupancy grid removes the need for a model-based, data association tracking, or at least greatly simplifies the model-based tracking process, when the identity of the obstacle is needed to be known for longer periods of time.

VII. ACKNOWLEDGMENT

This work was supported by the POSDRU-EXCEL post-doctoral program, financing contract POSDRU/89/1.5/ S/62557.

REFERENCES

[1] U. Franke, C. Rabe, H. Badino, and S. Gehrig, “6d-vision: Fusion of stereo and motion for robust environment perception,” Proc. 27th Annual Meeting of the German Association for Pattern Recognition DAGM ’05, 2005, pp. 216-223.

[2] D. Pfeiffer and U. Franke, "Efficient Representation of Traffic Scenes by Means of Dynamic Stixels", Proc. IEEE Intelligent Vehicles Symposium (IEEE-IV), 2010, pp. 217-224.

[3] A. Elfes, “A Sonar-Based Mapping and Navigation System”, Proc. IEEE International Conference on Robotics and Automation, April 1986, pp. 1151-1156.

[4] A. Elfes, “Using Occupancy Grids for Mobile Robot Perception and Navigation”, Computer, vol. 22, No. 6, June 1989, pp. 46-57.

[5] C. Coue, C.Pradalier, C.Laugier, T.Fraichard, and P.Bessiere, “Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application”, The International Journal of Robotics Research, Vol. 25, No. 1, 2006, pp. 19-30.

[6] C. Chen, C. Tay, K. Mekhnacha, and C. Laugier, “Dynamic environment modeling with gridmap: a multiple-object tracking application”, Proc. International Conference on Automation, Robotics and Computer Vision (ICARCV) 2006, pp. 1-6.

[7] M. Isard and A. Blake, “CONDENSATION -- conditional density propagation for visual tracking”, International Journal of Computer Vision, Vol. 29, No. 1, 1998, pp. 5-28.

[8] R. Danescu, S. Nedevschi, M.M. Meinecke, and T. Graf, “Stereovision Based Vehicle Tracking in Urban Traffic Environments”, Proc. IEEE Intelligent Transportation Systems Conference (IEEE-ITSC), 2007, pp 400-404.

[9] A. Barth and U. Franke, “Tracking Oncoming and Turning Vehicles at Intersections”, Proc. IEEE Intelligent Transportation Systems Conference (IEEE-ITSC), 2010, pp. 861-868.

[10] S. Bota and S. Nedevschi, “Tracking Multiple Objects in Urban Traffic Environments Using Dense Stereo and Optical Flow”, IEEE Intelligent Transportation Systems Conference (IEEE-ITSC), 2011, Accepted, Unpublished.

[11] R. Danescu, F. Oniga, and S. Nedevschi, “Particle Grid Tracking System for Stereovision Based Environment Perception”, Proc. IEEE Intelligent Vehicles Symposium (IEEE-IV), 2010, pp. 987-992.

[12] R. Danescu, F. Oniga, and S. Nedevschi, “Modeling and Tracking the Driving Environment with a Particle Based Occupancy Grid”, IEEE Transactions on Intelligent Transportation Systems, in print, DOI 10.1109/TITS.2011.2158097.