Particle Grid Tracking System for Stereovision Based Obstacle Perception in Driving Environments

Radu Danescu, Cosmin Pantilie, Florin Oniga, and Sergiu Nedevschi, *Member, IEEE*

***Abstract*— This paper presents an occupancy grid tracking system based on particles, and the use of this system for dynamic obstacle detection in driving environments. The particles will have a dual nature – they will denote hypotheses, as in the particle filtering algorithms, but they will also be the building blocks of our modeled world. The particles have position and speed, and they can migrate in the grid from cell to cell depending on their motion model and motion parameters, but they will also be created and destroyed using a weighting-resampling mechanism specific to particle filter algorithms. An obstacle grid derived from processing a stereovision-generated elevation map is used as measurement information, and the measurement model takes into account the uncertainties of the stereo reconstruction. The dynamic occupancy grid is used for improving the quality of the stereovision-based reconstruction as oriented cuboids. The resulted system is a flexible, real-time tracking solution for dynamic unstructured driving environments, and a useful tool for extracting intermediate dynamic information that can considerably improve object detection and tracking.**

I. Introduction

Advanced Driving Assistance Systems include complex functions of perception, decision and action. Because decision, and ultimately action, relies on the results of the perception, the perception function becomes extremely important, and considerable effort should be put into making sure that the results it produces are consistent with the reality.

Some of the most important bits of reality that need to be perceived in the driving scenario are the surrounding obstacles. The position, size and speed of the obstacles are determined by sensorial data processing, which includes feature selection, feature grouping (obstacle detection), and object tracking. When the sensor is not capable of providing dynamic information directly, tracking becomes the only way of estimating the objects' speed.

When the observed environment is composed mainly of obstacles having a standard geometry (such as cars and trucks), model-based tracking is the natural choice to be employed. In this case, the obstacles can be modeled as cuboids having position, size and speed vectors. The highway and most of the urban and rural sections of road are suitable for model-based tracking.

The conditions change when the environment to be tracked is an intersection, a busy urban center, or an off-road scenario. Even if parts of this environment can be tracked by simple geometric models, many essential parts of the environment will not fulfill these models, or they are incompletely observed by the sensors and the models are inaccurately matched. In order to get speed information for these cases, tracking must be employed at intermediate level, and not after the objects are reconstructed.

There are several approaches for intermediate level tracking, and these approaches are mostly defined by what kind of intermediate feature they track. The simplest 3D feature to be tracked is the 3D point, and that is the feature of choice for the solution called 6D Vision [1]. In this approach, the stereovision extracted 3D information is combined with the image space optical flow, and relevant points are tracked independently using Kalman filtering. Another intermediate representation of the 3D scene that is used in tracking is the dynamic stixel [2], a dynamic extension of the ground-based vertical entity called stixel that was first introduced in [3]. In the dynamic stixel world the visible obstacle areas are divided into vertical segments, and these segments are tracked from frame to frame using optical flow and Kalman filtering. Another approach based on fusion of stereo vision and tracked image features is presented in [4], where obstacles are represented as a rigid set of 3D points that are tracked in terms of feature displacements and depth measurements. In [5], the stereovision provided point clouds are used for cuboid reconstruction, but the position and speed are refined to a very accurate value by the use of radar along with stereo.

Occupancy grid based tracking is another popular choice for intermediate-level estimation of the dynamic properties of the scene, and this type of intermediate representation and tracking is the main focus of this paper. An occupancy grid is a probabilistic map of the driving environment, which encodes the present and past knowledge available from processing the sensor data, and which can be dynamically updated when new information becomes available.

One of the first uses of occupancy grids, in the context of sonar based navigation, is presented by Elfes in [6], and the probabilistic mechanism for updating the grid from sensorial data is described in [7]. The initial grids were static representations of the environment, and were therefore unsuitable for perception of dynamic entities. One approach

Manuscript received October 15, 2010. This work was supported by CNCSIS –UEFISCSU, project number PNII – IDEI 1522/2008.

Radu Danescu, Cosmin Pantilie, Florin Oniga and Sergiu Nedevschi are with the Technical University of Cluj-Napoca, Computer Science Department (e-mail: radu.danescu@cs.utcluj.ro). Department address: Computer Science Department, Str. Memorandumului nr. 28, Cluj-Napoca, Romania. Phone: +40 264 401457.

for dynamic occupancy grid modeling is presented in [8], where the speed components along the coordinate axes are added as extra dimensions in the grid, leading to a 4D representation.

Another dynamic occupancy grid tracking solution is presented in [9], where the speed of each cell is modeled as a distribution of possible values, and the tracker computes the probability of each value, along with the occupancy probability of the grid cell. The Bayesian reasoning process uses a set of possible antecedents, which are the cells that can influence the current cell based on the speed hypotheses, and the probabilities of the antecedents are combined with the sensor information. In [8] the dynamic Bayesian occupancy filter solution introduced in [9] is combined with the use of map information, which guides the hypotheses of the cell speeds to the allowed trajectories on the road. This solution prevents the system to consider unreachable positions, and enables it to better predict the vehicle paths when the road is curved.

Tracking dynamic occupancy grids is difficult, because the speed information attached to the grid cells increases the complexity of the state space significantly. For this reason, there are several mixed solutions, which use static grids while identifying the moving objects as occupancy trails [10] or inconsistencies [11] and tracking them separately.

Besides the dynamic versus static nature of the grid, and the way the dynamic grids represent their speed probability distribution, the grid solutions may differ in terms of their cell geometry. Three types of occupancy grid geometries are presented in [12], in the context of environment tracking based on stereo measurements: Cartesian (rectangular cell), column/disparity, and polar. Grid solutions differ also in the way sensorial data is used for grid updating, as some are based on the faster inverse sensor model [11], and some use the more accurate forward sensor model, whose advantages are described by Thrun in [13].

The occupancy grid model of the world is well suited for collaborative updating, using the information from multiple sensors or multiple observers. A solution that integrates the observations of multiple mobile observers into a unified description of the environment is presented in [14].

This paper presents an occupancy grid tracking solution based on particles. The particles will have a dual nature – they will denote hypotheses, as in the particle filtering algorithms such as CONDENSATION [15], but they will also be the building blocks of our modeled world. The tracking algorithm described in this paper will be particle-oriented, not cell oriented. The particles have position and speed, and they can migrate from cell to cell depending on their motion model and motion parameters, but they will also be created and destroyed using the same logic as the weighting-resampling mechanism described in [15]. The measurement data is the raw obstacle grid obtained by processing the elevation map, as described in [16], a measurement source which we have previously used for model-based object tracking, a technique described in [17].

Our solution is a fully dynamic, forward model-based Cartesian grid tracking system, which we believe comes as an improvement over the existing techniques, because due to the use of moving particles the representation of the speed probability distribution and the estimation of this distribution are no longer a concern. These distributions do not have to be approximated as histograms [9], Gaussian mixtures [8] or higher dimensions [18], and we don't have to assume that one cell belongs to only one object with only one velocity. The speed probability distribution of one grid cell results naturally from the surviving particles assigned to that cell.

The solution presented in this paper is a practical approach for probabilistic occupancy grid tracking, which has the benefit of simple integration of motion and measurement models, provides an easy mechanism for introducing additional constraints or information, and, by controlling the number of particles, allows the user to reach a tradeoff between accuracy and time performance.

The results of the grid based environment tracking are used as input data for an obstacle reconstruction algorithm, where the stereovision-provided 3D points are grouped into cuboids. The grid tracking algorithm adds dynamic information (magnitude and orientation of a speed vector) to the 3D points which will be grouped. The benefits of the improved obstacle detection algorithm are threefold. First, at individual cell level, object boundaries are more accurately detected. The detected obstacles are confidently described by using the cuboidal model. Next, by exploiting motion at obstacle level, the obstacle's orientation is more accurately and more easily determined. And finally, each obstacle carries speed information, a valuable cue for tracking and classification. The grid-based obstacle reconstruction algorithm is an extension of the one presented in [20].

II. System Overview

An overview of our solution is shown in figure 1. Each block is a key component of the overall algorithm, and the arrows show the dataflow. The *prediction* and *measurement based update* blocks form the main grid tracking cycle. The update is based on the *measurement model*, which transforms the *raw measurement*, unfiltered occupied cells extracted from elevation map processing, into conditional probabilities. The results of the measurement model block can also be used for grid cell *initialization*.

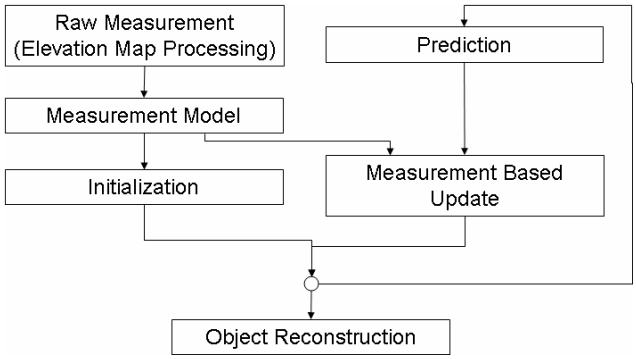

Fig. 1. Algorithm flow overview

As a final step, the results of the grid tracking process are used for cuboidal *object reconstruction*.

III. The Grid World Model

The particle grid tracking process is defined by our choice of the occupancy grid probabilistic representation. This chapter will describe our grid modeling solution.

The world is represented by a 2D grid, mapping the bird-eye view 3D space into discrete 10 cm x 10 cm cells. The size of the grid is 400 rows x 128 columns (this corresponds to a scene size of 40x12.8 meters). Like in other grid tracking solutions, the aim is to estimate the occupancy probability of each grid cell, and the speed components on each axis. However, these values are not key concepts in the workings of the algorithm that will be proposed in this paper, but they will be derived from a particle-based tracking mechanism.

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 $S=\{p_i \mid p_i=(x_i, z_i, vx_i, vz_i), i=1...N_S\}$, each particle i having a position in the grid, described by the row z_i (a discrete value of the distance in the 3D world) and the column x_i (discrete value of the lateral position), and a speed, described by the speed components vx_i and vz_i. The total number of particles N_S is not fixed, but depends on the number of obstacles in the scene. Having the population of particles in place, the occupancy probability of a cell C is 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.

$$P_O(C)=\frac{|\{p_i \in S \mid x_i=x_c, z_i=z_c\}|}{N_C} \tag{1}$$

The number of allowed particles per cell N_C is a constant of the system. In setting its value, a tradeoff 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 estimation of a grid cell is the average speed of its associated particles.

$$(vx_C, vz_C)=\frac{\sum_{p_i \in S, x_i=x_c, z_i=z_c}(vx_i, vz_i)}{|\{p_i \in S \mid x_i=x_c, z_i=z_c\}|} \tag{2}$$

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 goal of the tracking algorithm can now be stated: using the measurement information to create, update and destroy particles such that they accurately represent the real world.

IV. Prediction

This processing step will derive the present particle distribution from the past information, preparing the particle set for measurement. The prediction equations will use odometry and motion model information.

The basic odometry information available through the CAN bus of a modern car is the speed v and the yaw rate $\dot{\psi}$. Together with the time interval Δt elapsed between measurements, these parameters can be used to compensate for the ego-motion, and separate it from the independent motion of the objects in the scene. Between measurements, the ego-vehicle rotates with an angle ψ, and travels a distance d.

$$\psi=\dot{\psi}\Delta t \tag{3}$$

$$d=\frac{2v\Delta t \sin\frac{\psi}{2}}{\psi} \tag{4}$$

The origin of the grid representation is displaced along the two coordinate axes by d_x and d_z.

$$d_x=d\cos\psi / DX \tag{5}$$

$$d_z=d\sin\psi / DZ \tag{6}$$

We denote by DX and DZ the cell size of the grid. A point in the grid is displaced by the following equation:

$$\begin{bmatrix} x_n \\ z_n \end{bmatrix}=\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} x \\ z \end{bmatrix}-\begin{bmatrix} d_x \\ d_z \end{bmatrix} \tag{7}$$

The prediction is achieved using equation 8, which combines the deterministic drift caused by the ego-motion compensation and the particle's own speed, with the stochastic diffusion caused by the uncertainties in the motion model. The quantities δx, δz, δvx and δvz are randomly drawn from a Gaussian distribution of zero mean and a covariance matrix **Q** equivalent to the state transition covariance matrix of a Kalman filter.

$$\begin{bmatrix} x \\ z \\ v_x \\ v_z \end{bmatrix}=\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_n \\ z_n \\ v_x \\ v_z \end{bmatrix}+\begin{bmatrix} \delta x \\ \delta z \\ \delta vx \\ \delta vz \end{bmatrix} \tag{8}$$

The stochastic diffusion process allows the tracking system

to cope with changes in the environment. For example, if an obstacle is accelerating, the particles whose speed was increased by the stochastic process will be favored by the measurement based update process, and the average speed of the object's cells will be increased.

From the grid model point of view, the prediction has the effect of moving particles from one cell to another, as seen in figure 2. The occupancy probability is thus dynamically adjusted using the particle's motion model and the vehicle odometry.

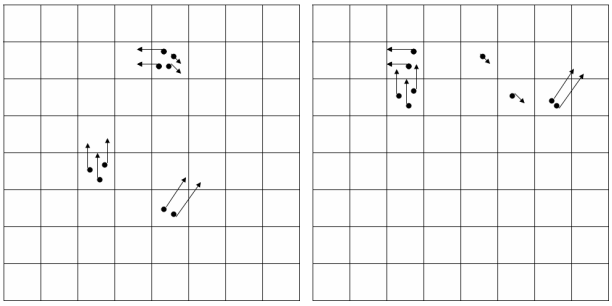

Fig. 2. Particles in the grid, before and after prediction.

V. Measurement Based Update

The predicted particles are confronted to the measurement data, processed through the measurement model block, and multiplied or deleted depending on how well the data from the sensors support the occupied or the free hypotheses of each cell. The classical steps of a particle filter based tracker are resampling, drift, diffusion, and measurement (weighting). This behavior replaces a population of a fixed number of particles with an equal number of particles, which approximates an updated probability density function over a space of parameters. However, this approach works when the particles are hypotheses of the state of a system, not when the particles are the system itself (we can see our tracked world as physically composed of particles).

Our algorithm tries to use the particles in a dual form – as hypotheses, and as building blocks of the world that we track. Their role as building blocks has been already explained. However, if we restrict our reasoning to a single cell in the grid world, we can see *that the particle is also a hypothesis*. A particle in a grid cell is a hypothesis that this cell is occupied, and that the cell has the speed equal to the speed of the particle. More particles in the cell mean that the hypothesis of occupancy is strongly supported. Less particles in the cell means that the hypothesis of the cell being free is supported. We can regard the difference between the number of particles in a cell and the total number of particles allowed in a cell as the number of particles having the occupancy hypothesis zero.

A. Weighting the particles

If we regard the number of particles in the cell to be constant, and some of them having the occupancy value "true" while some having it "false", we can apply the mechanism of weighting and resampling.

If we assume that the measurement data does not contain speed information, the weight of the particle depends only on the "occupied" hypothesis. Also, this means that all the particles having the same occupied hypothesis will have the same weight. The equations 9-14 apply to a single cell in the grid, and therefore the measurement, the coordinates *x, z* and the weights are cell specific.

$$w_{occupied} = p(m(x,z) \mid occupied) \tag{9}$$

$$w_{free} = p(m(x,z) \mid free) \tag{10}$$

The computation of the measurement conditional probabilities *p(m(x,y)|occupied)* and *p(m(x,y)|free)*, that is, the application of the forward sensor model, is detailed in section VI.

The number of particles having the "occupied" hypothesis true is the number of "real" particles in the cell.

$$N_{OC} = \mid \{ p_i \in S \mid x_i = x_c, z_i = z_c \} \mid \tag{11}$$

The number of particles (hypotheses) having the "occupied" value false is the complement of N_{OC}.

$$N_{FC} = N_C - N_{OC} \tag{12}$$

The total posterior probability of a cell being occupied and of a cell being free can be computed from the number of free/occupied hypotheses, and their corresponding weights:

$$P_{OC} = \frac{w_{occupied} N_{OC}}{w_{occupied} N_{OC} + w_{free}(N_C - N_{OC})} \tag{13}$$

$$P_{FC} = \frac{w_{free}(N_C - N_{OC})}{w_{occupied} N_{OC} + w_{free}(N_C - N_{OC})} \tag{14}$$

The aggregate (total) particle weights P_{OC} and P_{FC} are used for particle resampling. The resampling of the particle population is done at the end of the measurement step, so that the next cycle can start again with an updated population of particles without concerning about their weight.

B. Resampling

The classical resampling makes N_C random draws from the previous particle population of a cell, and the weight of each particle controls its chances of being selected. Because we don't care for the "cell free" hypothesis particles, our resampling will instead decide for each real particle (particle having the occupied hypothesis true) whether it is destroyed or multiplied (and, if multiplied, how many copies of it are created).

The following algorithm describes the process of resampling, which is materialized as duplication or removal of particles from the particle set.

Algorithm Resample

For each cell *C*

Compute N_{OC} and P_{OC}

Compute resampled number of particles N_{RC}

$N_{RC}=P_{OC}N_C$

Compute ratio between actual number of particles and the number of resampled particles

$$f_C = \frac{N_{RC}}{N_{OC}}$$

End For

For each particle p_i

Find corresponding cell *C*

If (f_C>1) – *number of particles will increase*

F_n = Int(f_C) Integer part

$F_f = f_C$ -Int(f_C) Fractional part

For k=1 **to** F_n-1

S.Add(p_i.MakeCopy)

End For

r = random value between 0 and 1

If ($r<F_f$)

S.Add(p_i.MakeCopy)

End if

End if

If (f_C <1) – *number of particles will decrease*

r = random value between 0 and 1

If ($r> f_C$)

S.Remove(p_i)

End if

End if

End For

The system will compute the number of particles that each cell should have after the process of resampling has been completed. The ratio f_C between this number and the existing number of particles in the cell will tell us if the particles have to be duplicated or removed. If f_C is higher than 1, the number of particles has to be increased. The integer part of the difference between f_C and 1 tells us the number of certain duplications a particle must undergo (for instance, if f_C is 2, each particle will be doubled). The fractional part of the difference is used for chance duplication: each particle will have a probability of being duplicated equal to the fractional part of this difference.

If *f* is lower than 1, the number of particles has to be decreased, by removing some of the particles. Each particle has 1- f_C chances of being eliminated.

At this point the cycle is complete, and the tracking algorithm can process a new frame. Secondary estimations for occupancy, speed, or clustering the cells into objects can be performed at the end of this step.

VI. Measurement Model

The measurement model will relate the measurement data, which is a binary occupied/free condition derived from the stereovision-generated elevation map [16], to the conditional probabilities *p(m(x,z) | occupied)* and *p(m(x,z) | free)*, probabilities that will weight the real and the virtual particles presented in the previous section. Building a sufficiently dense elevation map requires accurate dense stereo information, which is computed using a GPU optimized variant of the semi-global matching technique, described in [19].

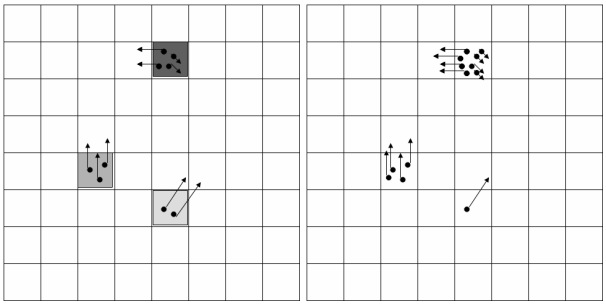

Fig. 3. Weighting and resampling. The weight of the occupied hypothesis is encoded in the darkness of the cell of the left grid.

In order to compute these probabilities, we start by computing the uncertainty of the stereo reconstruction, for each real world position x, z corresponding to a cell. First, the uncertainty of the distance reconstruction, in the case of a rectified system, is given by:

$$\sigma_z = \frac{z^2 \sigma_d}{bf} \tag{15}$$

In the above equation, *z* denotes the distance, *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 decent stereo reconstruction engine).

The error in lateral positioning (usually much smaller than the error in *z*), can be derived from the distance error:

$$\sigma_x = \frac{x \sigma_z}{z} \tag{16}$$

The 3D errors are mapped into grid cell errors, by dividing them with the grid cell size on *x* and *z*.

$$\sigma_{grid\_z} = \frac{\sigma_z}{DZ}$$
$$\sigma_{grid\_x} = \frac{\sigma_x}{DX} \tag{17}$$

In order to compute the conditional probability of the measurement cell, under the occupied or free assumption, we have to take into account a reality that is specific to stereovision sensors. The stereo sensor does not perform a scan of the scene, and therefore it does not output a single bird-eye view point for a real-world obstacle cell. We'll take as example a pillar, which has almost no width, and no depth spread. The representation of a pillar in the occupancy grid should be a single cell. If the pillar were observed by a scanner-type sensor, this sensor will output a cell, displaced from the true position by an amount specific to the sensor error. For the stereo sensor, things are different, because the

camera observes the whole height of the pillar, and therefore each pillar pixel will get a distance and a lateral position. This means that once we "collapse" the pillar information in the 2D grid representation, each part of the pillar may fall in a different cell, and the pillar will generate a spread of cells. The size of the spread area is controlled by the grid uncertainties on the *x* and *z* axes.

This property leads us to find a reasonable approximation for the conditional probabilities of the measurement cells under the occupied/free assumption. We'll count the obstacle cells in the measurement grid around the current cell position, in an area of σ_{grid_z} height and σ_{grid_x} width, and divide the number of found obstacle cells by the total number of cells in the uncertainty area. This ratio will be our approximation for *p(m(x,z) | occupied)*.

$$p(m(x,z)\mid occupied)=\frac{\sum_{row=z-\sigma_{z\_grid}}^{row=z+\sigma_{z\_grid}}\sum_{col=x-\sigma_{x\_grid}}^{col=x+\sigma_{x\_grid}}O(row,col)}{(2\sigma_{z\_grid}+1)(2\sigma_{x\_grid}+1)} \quad (18)$$

By *O(row, col)* we denote the "occupied" value of the measurement grid, at position *row* and *col*. This value is 1 when an obstacle cell is present and 0 when not.

The conditional probability of the measurement given the "free" assumption is:

$$p(m(x,z)\mid free)=1-p(m(x,y)\mid occupied) \quad (19)$$

These conditional probability values will be used to weight the particles. A graphic comparison between the raw measurement data and the conditional probability of the measurement under the "occupied" assumption is given in the following figure.

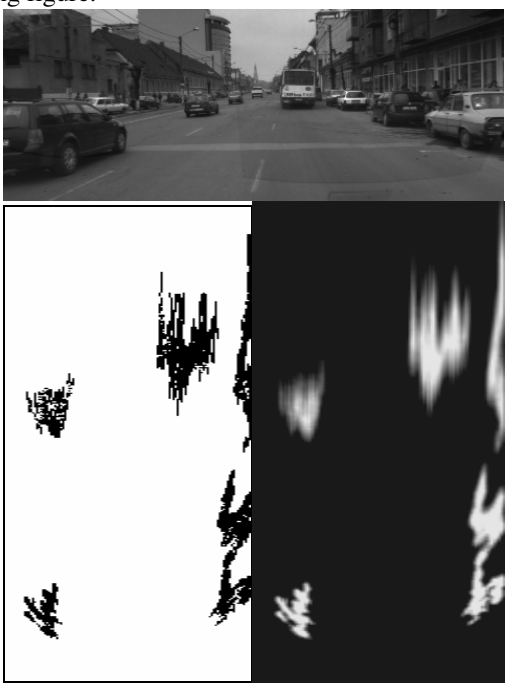

Fig. 4. From the occupancy grid to the particle weights. The bottom-right image encodes the weights of the occupied hypothesis.

VII. Initialization

Although the measurement step takes care of particle creation and deletion, this step only works if there are particles to be duplicated or deleted. For the prediction-measurement cycle to work, the particle population has to be initialized.

From a strictly probabilistic point of view, each cell's state is unknown at startup, which means that the cell has equal probability of being occupied or free. In our tracking system, this would mean that each cell should be assigned a number of particles equal to half the total number of particles allowable in a cell. However, this approach would significantly reduce the speed of the system, and would require permanent re-initialization.

Our solution is to use the measurement occupancy grid to create particles. If a measurement cell is of type obstacle, its *p(m(x,z) | occupied)* is high, and there are no particles in the corresponding tracked grid cell, a small number of particles will be created. The initial speed components *vx* and *vz* of the created particles will be sampled randomly from an initial range of possible values, and the initial position is confined to the creation cell. In this way, the initialization is a continuous process.

Particles are automatically removed when they go outside the grid area, in the prediction phase. Another case of "administrative" removal (removal not caused by the probability mechanism described in section V) is when, due to particle drifting, the number of particles in a cell exceeds the allowed value.

VIII. Object Reconstruction using Grid Tracking Results

The grid tracking system provides probabilistic dynamic occupancy results, each cell in the grid receiving a probability of being occupied and free, and an average speed. These results are now used to extract individual dynamic objects, in the shape of cuboids, having position, size, speed and orientation.

A. Obstacle localization

The speed information for each cell in the tracking grid is used to augment each 3D point obtained from the dense stereo sensor as follows. For each frame, after the grid tracking cycle is completed and the speed estimates for each tracking grid cell are available, the 3D points are projected onto the tracking grid. This way each 3D point is associated with the cell of the tracking grid it falls onto. The 3D point is augmented to hold the speed of the cell it was associated with. The set of augmented 3D points holding position (x, y, z) and motion (magnitude, orientation) information is the input of the object reconstruction procedure described in the next paragraphs.

The augmented 3D points will be processed by a polar occupancy grid, with variable cell size, which ensure that point density is independent of the range. A full description of the method is presented in [21].

The obstacle localization algorithm entails a two-stage

procedure. All the described processing steps are performed on the polar occupancy grid which relates to the tracking grid solely through the use of the set of augmented 3D points.

First the polar grid is populated with range and motion information, from the 3D points (Fig. 5). The computed motion information is integrated in the polar occupancy grid by projecting the 3D points and their associated motion vectors onto cells of the grid. In effect, each cell of the polar grid enclosing one or more points for which motion information was computed will also entail a motion descriptor for its own lateral or longitudinal motion in the form of magnitude and orientation. The two components are computed using a circular histogram also known as a histogram in polar coordinates (shown in Fig. 6).

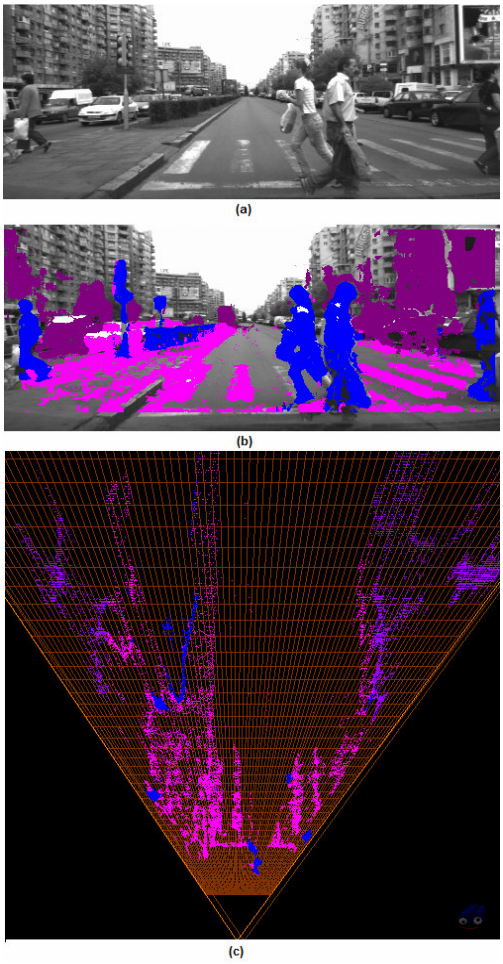

Fig. 5. Polar grid. (a) Gray scale image; (b) 3D points superimposed on the perspective image; Road points, points outside of the space of interest and obstacle points are classified and distinctively colored; (c) Top view of the scene: 3D points projected on the cells of the polar grid; Grid cell size is proportional to the distance from the origin of the world coordinate system.

The highest peak of the histogram is extracted and it represents the dominant direction (orientation) for a given polar grid cell. To account for angular errors of the velocities estimated by the grid tracking algorithm, each bin of the histogram holds a 5° interval of orientations.

The second stage of the method involves the use of a labeling algorithm that is able to exploit not only vicinity but also motion magnitude and orientation criteria.

A depth coherency constraint imposes that only neighboring cells can be grouped into an obstacle. Additionally, a candidate cell at the fringe of the partially discovered/built obstacle should be grouped to the same object only if the speed of the candidate cell is in agreement with that determined for the (partial) obstacle, up to that moment.

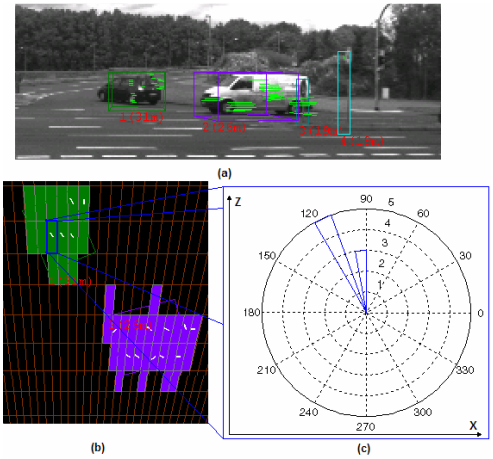

Fig. 6. Motion orientation is determined for each cell for which motion information is available. (a) Perspective view of the scene, detected obstacles with superimposed motion vectors; (b) close-up view of the grid cells corresponding to the two cars, with motion information (lines inside cells show determined orientation); (c) orientation histogram for the highlighted cell; orientation is in agreement with the travelling direction of the car.

As each new cell is added to the obstacle, the cell's contribution to the obstacle's overall motion signature is processed and the obstacle's properties are updated. Cells rejected due to motion differences are at the boundary of two closely positioned obstacles, and will be therefore grouped by the labeling algorithm into different obstacles.

```
procedure Add_cell_to_obstacle(obstacle, cell)
if cell ∈ fringe(obstacle)
   if holdsMotionInformation(obstacle)
      m_obst = |d_obst|
      o_obst = d_obst / |d_obst|
      if holdsMotionInformation(cell)
         m_cell = |d_cell|
         o_cell = d_cell / |d_cell|
         if |o_cell − o_obst| < th_orient and |m_cell − m_obst| < th_mag
            merge(obstacle, cell)
            updateMotionInformation(obstacle, m_cell, o_cell)
         end if
      else
         merge(obstacle, cell)
      end if
   else
      merge(obstacle, cell)
      if holdsMotionInformation(cell)
         m_cell = |d_cell|
         o_cell = d_cell / |d_cell|
         setMotionInformation(obstacle, m_cell, o_cell)
      end if
   end if
end if
```

Fig. 7. Vicinity and motion based decision logic used when adding a cell to a partially constructed obstacle.

This vicinity and motion based decision logic is described in pseudo code in Fig. 7 and offers insight on how to deal with special cases when either the cell or the partial obstacle do not hold motion information. A situation when the motion information can successfully discriminate between two obstacles that are very close together is shown in figure 8.

B. *Obstacle orientation*

This section introduces a new method of extracting the orientation of the non-stationary obstacles based on motion information. An accurate fitting of the cuboid on the detected obstacle is important for many subsequent algorithms that take as input the detected objects.

Prior to this approach obstacle orientation was determined by computing the envelope of the obstacle's 3D points and by analyzing the envelope's visible sides. Due to stereo reconstruction limitations the shape of the point cloud can sometimes lead to an erroneous orientation of the obstacle.

For non-stationary obstacles, motion can provide additional cues for orientation computation. Intuitively, the orientation of the obstacle should coincide with the orientation of the obstacle's motion vector (see Fig. 9).

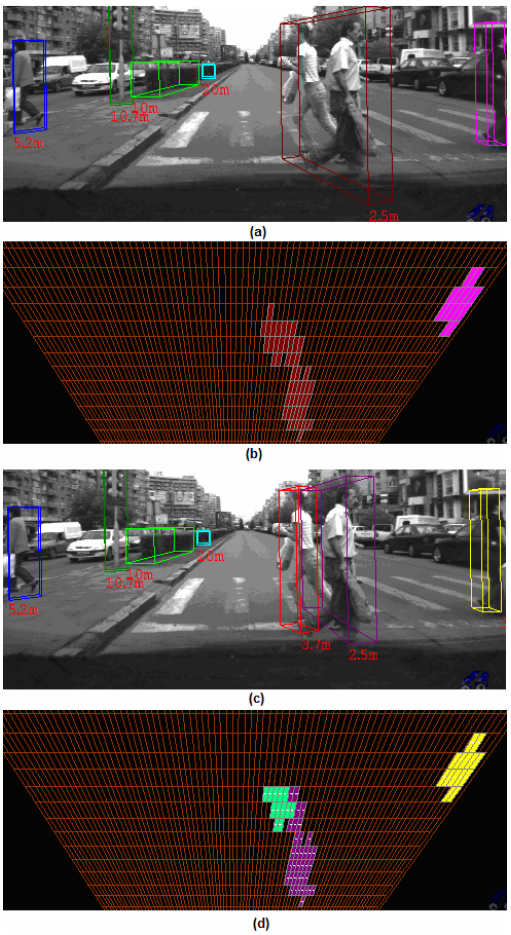

Fig. 8. (a) Detection result without the use of motion information; (b) Close-up on the grid cells of the merged pedestrians; (c) Improved detection results obtained with the use of motion information; (d) Close-up view on the cells of the two obstacles.

In order to improve the accuracy of the determined orientation the following technique was devised. Instead of working with the orientation approximated for each cell of the obstacle, we recompute the orientation of the obstacle by taking all of its vectors into account. Each motion vector that falls within the already computed three dimensional un-oriented bounds of the obstacle is added to a circular/polar histogram that is constructed for the analyzed obstacle.

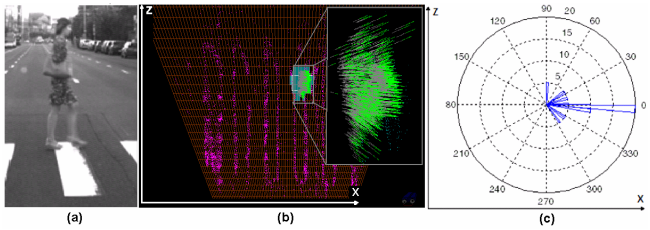

Fig. 9. (a) Pedestrian crossing the road from left to right, almost parallel to the longitudinal axes x; (b) Top-view of the 3D points and their associated motion vectors; (c) Orientation histogram for the pedestrian; the dominant direction of movement can be correctly.

The peaks of the circular histogram designate the possible orientations of the obstacle.

Each detected peak P_k is characterized by its weight and by its center of weight:

$$Weight_{P_k} = \sum_{i \in P_k} h[i] \qquad (20)$$

$$CenterOfWeight_{P_k} = \frac{1}{Weight_{P_k}} \sum_{i \in P_k} i * h[i]$$

The largest peak, P_{max} encompasses the prevailing orientation of the motion vectors and it is used to designate the orientation of the obstacle and the magnitude of the motion in that direction:

$$| \vec{d}^{\,obst} | = \frac{1}{|P_{\max}|} \sum_{v \in P_{\max}} |\vec{d}| \qquad (21)$$

$$\frac{\vec{d}^{\,obst}}{|\vec{d}^{\,obst}|} = CenterOfWeight_{P_{\max}}$$

When the motion based analysis cannot determine a predominant orientation the obstacle orientation must be estimated using the envelope based method.

IX. Tests and Results

A. *Testing the grid tracking algorithm*

We have designed two types of tests in order to validate the particle grid tracking algorithm: qualitative tests and quantitative (numerical) tests.

The qualitative assessment proves that the system is capable of building an occupancy probability grid from the measurement data, and is capable of identifying the motion associated with the cells in the grid. The qualitative assessment has been performed on real traffic scenes.

Some results, from two traffic scenes, are shown in figure

10. In the top row, the perspective image of the scene is given, and in the bottom row the three panels are, respectively: the measurement data (obstacle cells computed from the elevation map), the occupancy probability, and the speed labels, identifying the motion of the cells. In the left scene, we can identify two incoming vehicles (red), one outgoing vehicle (blue), and a stationary parked vehicle (black). In the right scene, we can identify a stationary parked vehicle, an outgoing vehicle, the stationary traffic sign on the isle, and the pedestrian group with a combination of stationary and lateral motion.

For numerical evaluation, we have chosen a "follow the leader" scenario, with only one obstacle in the scene, so that a reasonable estimation of the object's speed can be done in the absence of a cell clustering algorithm. The ego-vehicle is following the target vehicle, matching its speed. Therefore, the speed of the ego-vehicle is a benchmark for the estimated speed of the target. Figure 11 shows that after an initial lag (10 frames, 0.5 seconds), the estimated speed converges to the ego-speed. The absolute mean error after the lag period is 1 km/h.

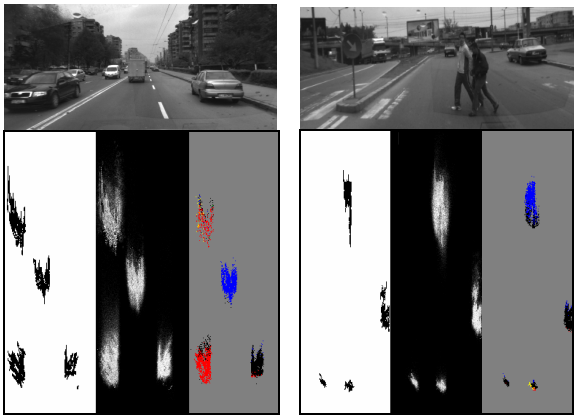

Fig. 10. Qualitative assessment of the algorithm performance. Speed labels: black – stationary, red – incoming, blue – outgoing, yellow – lateral motion.

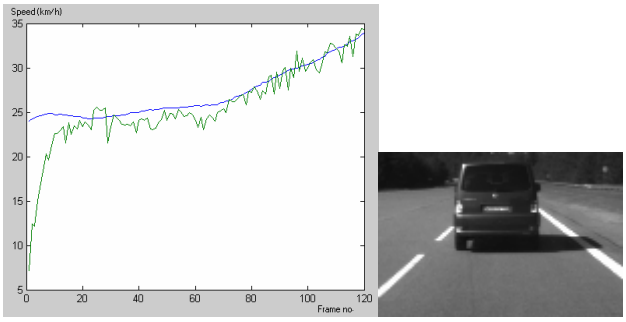

Fig. 11. Left – Ego-speed (blue) versus the estimated target speed (green), in km/h. Right – image of the target vehicle.

A second test implies a static object, observed from a moving vehicle traveling along a circular path. The circular path increases the difficulty of the estimation, due to the fact that the static object is in the field of view for only 3.5 seconds. The results of the speed estimation are shown in Figure 12, against the speed of the observing vehicle. The static nature of the object can be inferred almost immediately.

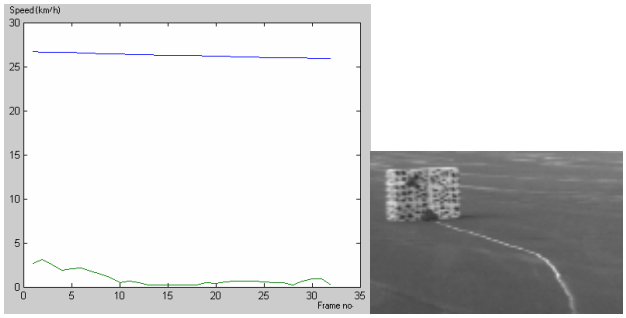

Fig. 12. Ego-speed (blue) versus the static object's estimated, in km/h. Right – the static object.

The time performance depends on the obstacle load of the scene, which influences the total number of particles. For a typical urban scene, and a total number of particles in a cell of 50, the total running time is about 40 ms per frame.

B. Testing the obstacle reconstruction algorithm

The most significant benefits that the obstacle reconstruction algorithm reaps from the particle grid tracking system are related to better identifying individual objects when they are close together, and a more stable and accurate estimation of object orientation.

The most common traffic situations when objects are close together and are difficult to separate tend to involve pedestrians. Their physical size is small, and they tend to leave little space between them, and therefore separating them as individual objects becomes difficult in the absence of dynamic information. Such is the case presented in figure 13, where two pedestrians pass each other at a small distance. Due to the uncertainties of the stereo measurement, they are sometimes grouped together.

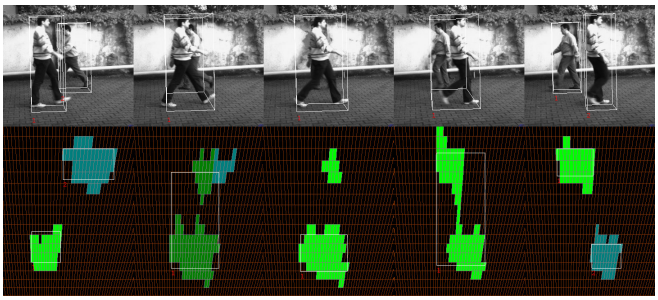

Fig. 13. Reconstruction based on feature position alone. Top – objects on the perspective image, bottom – grouped cells in the grid.

The outcome of the object reconstruction algorithm becomes significantly better when motion information, supplied by the grid tracking system, is used. The results shown in figure 14 do not exhibit the fusion problem any more. One of the main advantage of using a grid tracking system, as opposed to other techniques such as optical flow, is that when occlusion occurs (as in this case), the motion information is kept in the grid, and when the object appears again the motion vectors are readily available.

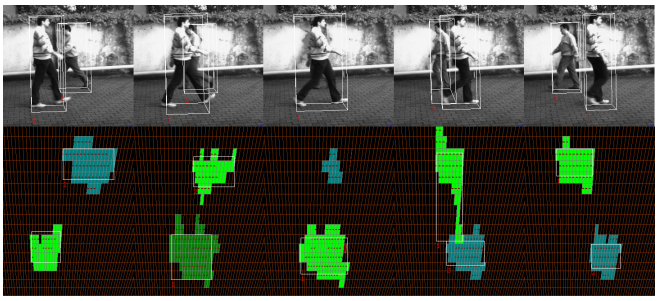

Fig. 14. Reconstruction results using the motion information. Top – objects on the perspective image, bottom – grouped cells in the grid.

The other performance parameter we wished to test was the improvement in orientation estimation. A test vehicle crossed our field of view at 135 degrees, as seen in figure 15.

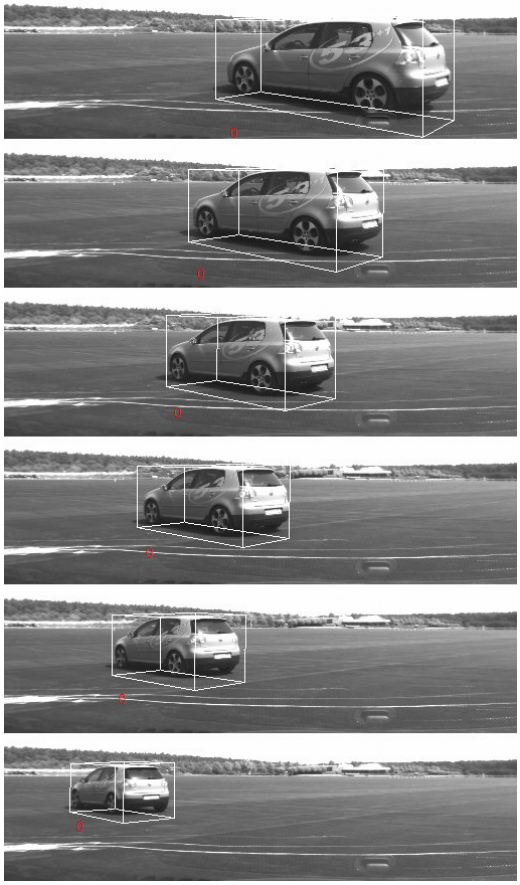

Fig. 15. Test sequence for orientation evaluation

The orientation estimated from the particle based motion information was compared to the orientation extracted from the shape of the 3D point cloud. The graph in figure 16 shows the results, which are clearly better for the motion based solution. The average of the geometry based results is 131.14 degrees, while the average of the motion-based results is 133.83, closer to the ground truth. The standard deviation of the geometry based estimation is 22.52 degrees, but if we remove the outliers it becomes 8.79 degrees. The standard deviation of the motion-based estimation is 3.39 degrees.

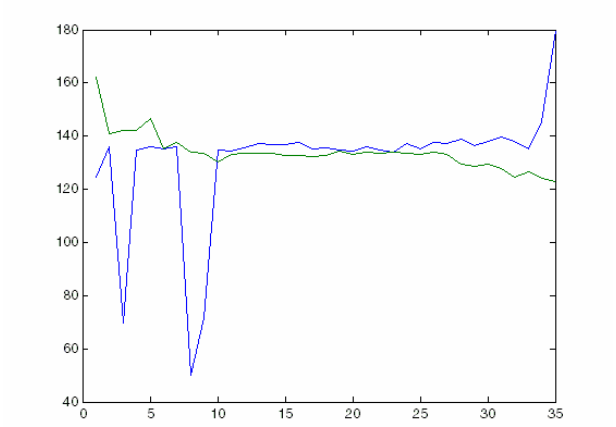

Fig. 16. Orientation from point cloud geometry (blue) versus the orientation estimated from motion (green). Orientation is in degrees, versus frame number.

The results show that the motion based object orientation, which relies on particle grid results, is clearly superior to the geometry-based estimation. Also, the comparison was made under conditions which are quite favorable to geometry-based estimation, as the object is quite close, and the stereo reconstruction quality is ideal. For objects farther away, in real traffic scenes, the difference between the methods will be even higher, but this scenario was chosen because of the availability of ground truth.

X. Conclusion and Future Work

We have presented a grid tracking technique that models and tracks the driving environment using a set of particles with position and speed. Our solution proves capable of identifying occupancy and motion in complex dynamic traffic scenes, without the need of feature grouping or obstacle model matching or data association.

The grid based tracking system is used as an intermediate method for dynamic parameters extraction, before the 3D features in the scene are grouped into cuboidal objects. The intermediate level tracking is not the only solution for attaching speed information to the primary features, because we can also achieve this using optical flow [19], but the use of tracking implies a higher stability of the speed estimation, robustness against temporary occlusions, and a lighter computation load for the same density of speed vectors.

The dynamic parameters extracted from the particle grid are used to augment the stereovision provided 3D points, which are then used for object reconstruction as a cuboid shape. This approach is able to significantly improve the feature grouping results, as the motion information can be used to discriminate between static and dynamic objects, and also between objects heading in different directions, which could have been grouped as a whole on vicinity criteria alone. Also, the orientation of the obstacles is more accurately determined using motion information, compared to using the shape of the 3D point cloud alone.

Future experiments with the particle grid size, speed and

position uncertainties of prediction, and a refinement of the measurement model to include the error of the occupant cell extraction besides the uncertainties of the stereo algorithm, will allow us to optimize this system's performance and accuracy.

REFERENCES

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

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

[3] H. Badino, U. Franke, D. Pfeiffer, "The Stixel World - A Compact Medium Level Representation of the 3D-World", *Lecture Notes in Computer Science*, Vol. 5748, 2009, pp. 51-60.

[4] A. Barth and U. Franke, "Estimating the Driving State of Oncoming Vehicles From a Moving Platform Using Stereo Vision," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 10, No. 4, pp. 560-571, 2009.

[5] W. Shunguang, S. Decker, P. Chang, T. Camus, J. Eledath, "Collision Sensing by Stereo Vision and Radar Sensor Fusion," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 10, No. 4, pp. 606-614.

[6] A. Elfes, "A Sonar-Based Mapping and Navigation System", in proc of *IEEE International Conference on Robotics and Automation*, April 1986, pp. 1151-1156.

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

[8] T. Gindele, S. Brechtel, J. Schroeder, R. Dillmann, "Bayesian Occupancy Grid Filter for Dynamic Environments Using Prior Map Knowledge", *IEEE Intelligent Vehicles Symposium,* 2009, pp. 669 – 676.

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

[10] T. Weiss, B. Schiele, K. Dietmayer, "Robust Driving Path Detection in Urban and Highway Scenarios Using a Laser Scanner and Online Occupancy Grids", *IEEE Intelligent Vehicles Symposium,* 2007, pp. 184-189.

[11] S. Pietzch, T. D. Vu, J. Burtlet, O. Aycard, T. Hackbarth, N. Appenrodt, J. Dickmann, B. Radig, "Results of a Precrash Application based on Laser Scanner and Short Range Radars", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 10, No. 4, 2009, pp. 584-593.

[12] H. Badino, U. Franke, R. Mester, "Free Space Computation Using Stochastic Occupancy Grids and Dynamic Programming", *Workshop on Dynamical Vision, ICCV,* 2007, pp. 1-12.

[13] S. Thrun, "Learning Occupancy Grids With Forward Sensor Models", *Autonomous Robots*, Vol. 15, No 2, 2003, pp. 111-127.

[14] J. Y. Chen, J. Hu, "Probabilistic Map Building by Coordinated Mobile Sensors", *IEEE International Conference on Networking, Sensing and Control,* 2006, pp. 807-812.

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

[16] F. Oniga, S. Nedevschi, "Processing Dense Stereo Data Using Elevation Maps: Road Surface, Traffic Isle, and Obstacle Detection", *IEEE Transactions on Vehicular Technology*, Vol. 59, No. 3, March 2010, pp. 1172-1182.

[17] R. Danescu, F. Oniga, S. Nedevschi, M.-M. Meinecke, "Tracking Multiple Objects Using Particle Filters and Digital Elevation Maps", *IEEE Intelligent Vehicles Symposium,* 2009, pp. 88-93.

[18] C. Coue, C.Pradalier, C.Laugier, T.Fraichard, 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.

[19] C. Pantilie, S. Nedevschi, "Real-time Obstacle Detection in Complex Scenarios Using Dense Stereo Vision and Optical Flow", *IEEE Conference on Intelligent Transportation Systems (IEEE-ITSC)*, 2010, pp. 439-444.

[20] I. Haller, C. Pantilie, F. Oniga, S. Nedevschi, "Real-time semi-global dense stereo solution with improved sub-pixel accuracy", *IEEE Intelligent Vehicles Symposium 2010 (IV 2010)*, pp. 369-376.

[21] C. Pocol, S. Nedevschi, and M.M. Meinecke, "Obstacle Detection based on Dense Stereovision for Urban ACC Systems," *5th International Workshop on Intelligent Transportation (WIT)*, 2008, pp. 13-18.

Radu Danescu received the Diploma Engineer degree in Computer Science in 2002 from the Technical University of Cluj-Napoca, Romania, followed by the M.S. degree in 2003 and the PhD (Computer Science) degree in 2009, from the same university. He is a Senior Lecturer with the Computer Science Department, TUCN, teaching Image Processing, Pattern Recognition, and design with microprocessors. His main research interests are stereovision and probability based tracking, with applications in driving assistance. He is a member of the Image Processing and Pattern Recognition Research Laboratory at TUCN.

Cosmin Pantilie received his M.S. and B.S. degrees in computer science from the Technical University of Cluj-Napoca, Romania in 2011 and 2009, respectively. He is currently working toward the PhD. degree in computer science, focusing on environment perception for intelligent vehicles using stereo vision. His research interests include image processing, pattern recognition, stereo vision and high performance computing on multi-core architectures.

Florin Oniga received the Diploma Engineer degree in Computer Science in 2002 from the Technical University of Cluj-Napoca, Romania, followed by the M.S. degree in 2003 from the same university. He is currently working towards the Ph.D. degree in Computer Science at Technical University of Cluj-Napoca, specializing in Computer Vision. He is a Lecturer with the Computer Science Department, Technical University of Cluj-Napoca, teaching Image Processing, Pattern Recognition, and Computer Architecture. His research interests include stereovision, digital elevation maps processing, and vision based automotive applications. He is a member of the Image Processing and Pattern Recognition Research Laboratory at TUCN.

Sergiu Nedevschi (M'99) received the M.S. and PhD degrees in Electrical Engineering from the Technical University of Cluj-Napoca (TUCN), Cluj-Napoca, Romania, in 1975 and 1993, respectively. From 1976 to 1983, he was with the Research Institute for Computer Technologies, Cluj-Napoca, as researcher. In 1998, he was appointed Professor in computer science and founded the Image Processing and Pattern Recognition Research Laboratory at the TUCN. From 2000 to 2004, he was the Head of the Computer Science Department, TUCN, and is currently the Dean of the Faculty of Automation and Computer Science. He has published more than 200 scientific papers and has edited over ten volumes, including books and conference proceedings. His research interests include Image Processing, Pattern Recognition, Computer Vision, Intelligent Vehicles, Signal Processing, and Computer Architecture.