Modeling and Tracking the Driving Environment With a Particle-Based Occupancy Grid

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

***Abstract*—Modeling and tracking the driving environment is a complex problem due to the heterogeneous nature of the real world. In many situations, modeling the obstacles and the driving surfaces can be achieved by the use of geometrical objects, and tracking becomes the problem of estimating the parameters of these objects. In the more complex cases, the scene can be modeled and tracked as an occupancy grid. This paper presents a novel occupancy grid tracking solution based on particles for tracking the dynamic driving environment. 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 be also created and destroyed using a weighting-resampling mechanism that is specific to particle filtering algorithms. The tracking algorithm will be centered on particles, instead of cells. 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 resulting system is a flexible real-time tracking solution for dynamic unstructured driving environments.**

***Index Terms*—Environment modeling, occupancy grids, particle filtering, stereovision, tracking.**

I. Introduction

THE TASKS of modeling and perceiving the driving environment are a continuous challenge because there are multiple types of scenarios of different degrees of order and complexity. Some environments are well regulated, and the types of static and dynamic objects are easily modeled and tracked using geometrical models and their parameters. The obstacles can be modeled as cuboids having position, size, and speed, and the driving surface delimiters can be modeled as parametrical curves. The highway and most of the urban and rural sections of roads are usually suitable for geometrical modeling and 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 estimating the parameters of a geometrical model, many essential parts of the environment will not fulfill the constraints of the models. In addition, sometimes a driving assistance application needs to have static and dynamic information about the environment before a model can be instantiated and tracked, or it may use this additional information in model fitting and model-based tracking. For these reasons, solutions for intermediate-level representation and tracking are devised. These intermediate representation and tracking solutions can be based on occupancy grids or directly on the 3-D points (the 6D-vision technique [1]) or on compact dynamic obstacle primitives called stixels [2], or they can be replaced with specialized techniques of detecting critical motion [3]. In what follows, we will focus on works related to occupancy grids.

Maybe one of the first uses of occupancy grids, under the name of probabilistic local maps, is presented by Elfes in [4], in the context of sonar-based robot navigation. Another paper by the same author [5] names the occupancy maps occupancy grids and describes the probability inference mechanism for handling the uncertainty of a range sensor in computing the probability of each cell's occupancy state. In the same reference, we find a definition of the occupancy grid: "the occupancy grid is a multidimensional random field that maintains stochastic estimates of the cells in a spatial lattice."

The initial occupancy grids, such as those presented in [4] and [5], are simple 2-D maps of the environment, each cell describing the probability of it being occupied or free. However, for many tracking applications, particularly in the driving assistance field, there is a need for estimating the dynamic parameters of the environment, namely, the speed of each grid cell. By adding the speed factor in the environment estimation, the complexity significantly increases, as the cells are now strongly interconnected. The work of Coué *et al.* [6] uses a 4-D occupancy grid, where each cell has a position and two speed components along each axis. By estimating the occupancy of each cell in the 4-D grid, the speeds for the classical cells in the 2-D grid can be computed.

Another solution for the representation of speeds is presented in Chen *et al.* [7]. Instead of having a 4-D grid, this solution comes back to 2-D but uses for each cell a distribution of speeds, in the form of a histogram. The Bayesian inference mechanism relies on sensor data and antecedent cells, the list of antecedents being decided by the speed hypotheses.

A simpler but limited way of handling the dynamic aspects of the environment is presented in [8]. Instead of estimating the speed of each cell, this solution relies on "occupancy trails," which are specific patterns, similar to the motion blur of the

Manuscript received June 2, 2010; revised October 7, 2010 and April 4, 2011; accepted May 20, 2011. Date of publication July 7, 2011; date of current version December 5, 2011. This work was supported in part by Consiliul National al Cercetarii Stiintifice din Invatamantul Superior (CNCSIS)–Unitatea Executiva pentru Finantarea Invatamantului Superior si a Cercetarii Stiintifice Universitare (UEFISCSU) under Project Programul National II (PNII)-IDEI 1522/2008 and in part by the Programul Operational Sectorial Dezvoltarea Resurselor Umane (POSDRU) Program under Contract POSDRU/89/1.5/S/62557. The Associate Editor for this paper was S. Sun.

The authors are with the Department of Computer Science, Technical University of Cluj-Napoca, 400114 Cluj-Napoca, Romania (e-mail: radu.danescu@cs.utcluj.ro).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2011.2158097

camera, which can be used to derive the trajectory, and therefore the speed, of the moving objects. A more sophisticated method is presented in [9], where the inconsistencies in the static grid are detected as soon as they appear, and a multimodel Kalman filter tracker is initialized to track the dynamic object.

We can attempt a first classification of the dynamic occupancy grid solutions (not the grids themselves) into fully dynamic, such as those presented in [6], [7], and [10], and static–dynamic hybrids, such as those presented in [8] and [9].

One of the most important features of an occupancy grid tracking solution is the way the sensor model is used for grid update. The most time efficient way of updating a grid is to rely on the inverse sensor model, which derives the probability of a cell being occupied directly from the sensor readout, assuming the occupancy of each cell is independent of its neighbors. This solution is still maybe the most popular, mainly in static grids [9]. However, the work of Thrun [11] proved that forward sensor probability models are preferable, even in the case of static grids, even if this significantly increases the complexity of computation.

The occupancy grids can have multiple spatial representations, and in [12], we are shown a comparison between three types of grids, namely, the Cartesian (classic), the polar (distance and angle), and the column/disparity grids. All these grids have advantages and drawbacks. A Cartesian grid is closer to the real-world representation and can handle velocities easier, whereas the other types of grids are more "sensor friendly," making the computation of the sensor uncertainties easier.

The occupancy grid is a flexible representation of the environment, and this flexibility allows powerful integration of multiple information sources. For instance, map information can be mapped on the grid, when available, as presented in [10]. The map can associate to each cell a terrain type (such as road, curb, or sidewalk), and the terrain type is translated into a reachability probability for the cell. The use of terrain information can greatly improve the prediction of the position of dynamic objects on the road. The flexibility of the occupancy grid makes it well suited for collaborative updating, using the information from multiple sensors or multiple observers. A solution that uses the occupancy grid (named obstacle map) to integrate laser and radar information is presented in [13], and in [14], the grids are used to fuse stereo and optical flow information. A solution that integrates the observations of multiple mobile observers into a unified description of the environment is presented in [15].

This paper presents a driving environment tracking solution based on a particle occupancy grid. This solution is defined by a new and original approach for the representation of the occupancy and velocity probability distribution of each grid cell and by the original updating algorithm derived from the proposed representation. 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, such as in the particle filtering algorithms such as CONDENSATION [16], but can also be regarded as physical building blocks of our modeled world. The tracking algorithm described in this paper is particle oriented and 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 are also created and destroyed using the same logic as the weighting–resampling mechanism described in [16]. The measurement data is the raw obstacle grid obtained by processing the elevation map, as described in [17]. Building a sufficiently dense elevation map requires accurate dense stereo information, which is computed using the techniques described in [18]. Other techniques for dense stereo processing are presented in [19].

Based on the surveyed literature, the occupancy grid tracking solution presented in this paper can be classified as having a *Cartesian representation*, using a *forward sensor probability model*, and producing a *fully dynamic grid*. The proposed method is most closely related to the works presented in [7] and [10], which use a speed probability distribution for each cell in the grid instead of modeling the dynamic grid as a high-dimensional space, as in [6]. We believe that our solution comes as an improvement over these 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. We do not have to approximate the velocity as a histogram [7] or as a mixture of Gaussians [10], we do not have to assume that one cell belongs to only one object with only one velocity, and neither are we concerned with estimation of this speed, as this results naturally from the survival or elimination of the particles. The particles in a cell can have different speeds, and therefore, they can handle the situation of overlapping objects or the most likely situation when the objects are too close and the uncertainty of one overlaps over the uncertainty of the other. The complexity of the algorithm is linear with the number of cells in the grid and with the maximum number of particles in a cell, a tradeoff between accuracy and response time being always available as a simple parameter. In addition, integrating other motion parameters such as acceleration does not increase the complexity of the tracking algorithm because it only alters the way the position of the particles in time is computed.

The remainder of this paper is organized as follows: First, the particle grid model is presented, and then, the steps of the filtering algorithm are detailed, i.e., prediction, measurement, and initialization. Then, this paper describes the way the particle grid results can be used to extract 3-D cuboids that have position, size, and speed. This paper ends with the testing and results section, followed by conclusions.

II. World Model

The world is represented by a 2-D grid, mapping the bird's-eye view 3-D space into discrete 20 cm $\times$ 20 cm cells. The size of the grid is 250 rows $\times$ 120 columns (this corresponds to a scene size of 50 m $\times$ 24 m). The aim of the tracking algorithm is to estimate the occupancy probability of each grid cell and the speed components on each axis. The tracking goals are achieved by the use of a particle-based filtering mechanism.

Considering a coordinate system where the z-axis points toward 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 $S = \{p_i | p_i = (c_i, r_i, vc_i, vr_i, a_i), i = 1, \ldots, N_S\}$, each particle i having a position in the grid, as described by

row r_i (a discrete value of distance in the 3-D world z) and column c_i (a discrete value of lateral position x), and a speed, as described by 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 process, which will be described in a subsequent section of this paper. The total number of particles in scene N_S is not fixed. This number depends on the occupancy degree of the scene, i.e., the number of obstacle cells. 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 cell C and the total number of particles allowed for a single cell N_C, i.e.,

$$P_O(C) = \frac{|\{p_i \in S | r_i = r_c, c_i = c_c\}|}{N_C}. \tag{1}$$

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 to N_C, and therefore, the time consumption will increase.

The speed estimation 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.

$$(vc_C, vr_C) = \frac{\sum_{p_i \in S,\, x_i = x_c,\, z_i = z_c} (vc_i, vr_i)}{|\{p_i \in S | r_i = r_c, c_i = c_c\}|}. \tag{2}$$

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 simultaneously maintained 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.

III. Algorithm Overview

The first step of the algorithm is *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. In addition, a random amount is added to the position and speed of each particle for the effect of stochastic diffusion. The second step is *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. The measurement model information is used to *weight* the particles and *resample* them in the same step. 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 and to group the cells into 3-D oriented objects for result evaluation.

IV. Prediction

This 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 controller area network bus of a modern car is speed v and 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. Thus

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

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

The origin of the grid representation is displaced along the two coordinate axes by d_c and d_r, i.e.,

$$d_c = d \, \sin \frac{\psi}{2} / DX \tag{5}$$

$$d_r = d \, \cos \frac{\psi}{2} / DZ. \tag{6}$$

We denote by DX and DZ the cell size of the grid (in the current implementation, it is 0.2 m). A point in the grid, at row r and column c, is displaced by the following equation:

$$\begin{bmatrix} c_n \\ r_n \end{bmatrix} = \begin{bmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{bmatrix} \begin{bmatrix} c \\ r \end{bmatrix} - \begin{bmatrix} d_c \\ d_r \end{bmatrix}. \tag{7}$$

The prediction is achieved using (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. Quantities δc, δr, δvc, and δvr are randomly drawn from a Gaussian distribution of zero mean and a covariance matrix $\mathbf{Q}$ equivalent to the state transition covariance matrix of a Kalman filter. The covariance matrix is diagonal, with the standard deviations for the speed components corresponding to a real-world amount of 1 m/s and the standard deviations for the position corresponding to a real-world value of 0.1 m. These values will ensure that the system is able to cope with fast moving objects even at a frame rate of 10 frames/s.

$$\begin{bmatrix} c \\ r \\ v_c \\ v_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_n \\ r_n \\ v_c \\ v_r \end{bmatrix} + \begin{bmatrix} \delta c \\ \delta r \\ \delta vc \\ \delta vr \end{bmatrix}. \tag{8}$$

From the grid model point of view, the prediction has the effect of moving particles from one cell to another, as shown in Fig. 1. The occupancy probability is thus dynamically adjusted using the particle's motion model and the vehicle odometry.

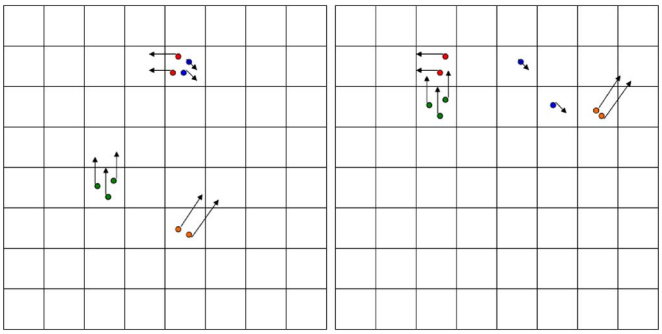

Fig. 1. Particles in the grid before and after prediction.

V. Measurement Model

The measurement model will relate the measurement data, which is a binary occupied/free condition derived from the stereovision-generated elevation map [10], to conditional probabilities $p(\text{measurement}|\text{occupied})$ and $p(\text{measurement}|\text{free})$, which will weight the particles. To compute these probability values, we have to pass through several steps.

A. Uncertainty of the Stereo Measurement

To compute these probabilities, we start by computing the uncertainty of the stereo reconstruction. First, the uncertainty of the distance reconstruction, in the case of a rectified system, is given by

$$\sigma_z = \frac{z^2 \sigma_d}{bf}. \tag{9}$$

In the preceding 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 lateral position x (in the real-world coordinates) and distance z, i.e.,

$$\sigma_x = \frac{x \sigma_z}{z}. \tag{10}$$

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

$$\sigma_{\text{row}} = \frac{\sigma_z}{DZ}$$
$$\sigma_{\text{column}} = \frac{\sigma_x}{DX}. \tag{11}$$

The values of σ_{row} and σ_{column} are computed offline, at the initialization phase, for each cell in the grid.

B. Raw Occupancy Density Cue

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's-eye view point for a real-world obstacle cell. We will take as an 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, which is 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 2-D 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 c and r axes (real world x and z).

This property leads us to find a good cue, which will contribute to the conditional probabilities of the measurement cells under the occupied/free assumption. We will count the obstacle cells in the measurement grid around the current cell position, in an area of σ_{row} height and σ_{column} width, and divide the number of found obstacle cells by the total number of cells in the uncertainty area. We will denote this ratio as $p_{\text{density}}(m(r,c)|\text{occupied})$, i.e.,

$$p_{\text{density}}\left(m(r,c)|\text{occupied}\right) = \frac{\sum_{\text{row}=r-\sigma_{\text{row}}}^{\text{row}=r+\sigma_{\text{row}}} \sum_{\text{col}=c-\sigma_{\text{column}}}^{\text{col}=c+\sigma_{\text{column}}} O(\text{row},\text{col})}{(2\sigma_{\text{row}}+1)(2\sigma_{\text{column}}+1)}. \tag{12}$$

We denote the "occupied" value of the measurement grid at position row and col by $O(\text{row},\text{col})$. This value is 1 when an obstacle cell is present and 0 when not.

The density cue for the "free" assumption is

$$p_{\text{density}}\left(m(r,c)|\text{free}\right) = 1 - p_{\text{density}}\left(m(r,c)|\text{occupied}\right). \tag{13}$$

A graphic comparison between the raw measurement data and the density cue (conditional probability) of the measurement under the "occupied" assumption is given in Fig. 2.

C. Handling the Occlusions

Not all cells in the grid can be directly observed, and this fact must be taken into consideration by the tracking algorithm. Due to the limitations of the primary source of information, i.e., the stereovision-based raw occupancy grid, some of the cells are never observed. The raw occupancy grid only covers a longitudinal distance from 0 to 40 m and a lateral span of 13 m. In addition, the field of view of the camera (angular span) limits the areas that are visible at close distance. The cells that are excluded by the field of view and distance limitations are marked as obstructed (unobservable) by default.

Another way for a cell to become unobservable is if it is obstructed by an obstacle cell that is located between it and the observation origin (camera position). To decide if a cell is in such a situation, we switch to polar coordinates. Each cell is mapped to a polar grid. Then, for each angle, the cells are

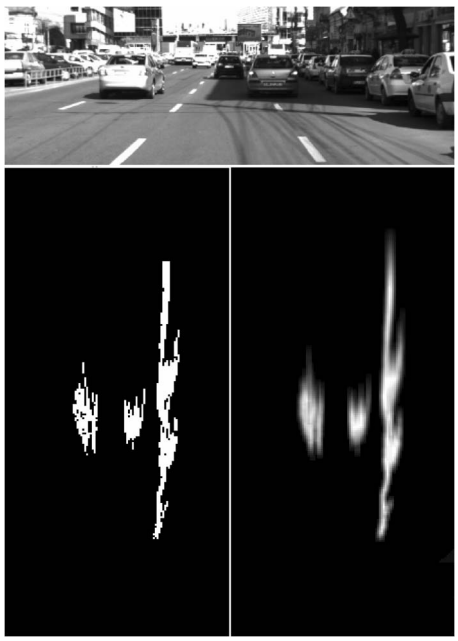

Fig. 2. From the raw occupancy grid to the raw measurement density cues. (Bottom left) Raw occupancy grid. (Bottom right) Density cue for the occupied cell hypothesis.

scanned in the order of their distance. Once a raw occupied cell is found, an obstruction counter is incremented for every cell that is behind the first occupied cell. Then, the obstruction values are remapped into the Cartesian grid.

Once each cell has an obstruction value, the final analysis is performed. Each cell that has an obstruction value higher than 10 is considered obstructed and considered as such in the particle weighting and resampling phase (to be described in the next section). However, this is not the only way we use the obstruction property. If a raw measurement cell is marked as "occupied," but from the obstruction analysis, it is found to be obstructed, the occupied cell is removed. This will make the raw occupancy map look more like a scanner-derived map. This reduction of measurement information must be performed before the computation of the other particle weighing cue, which relies on the distance from measurement.

The obstruction-related processing steps are shown in Fig. 3. The left panel shows the raw measurement data, the middle panel shows the obstruction value for each cell (the lighter, the more obstructed), and the right panel shows the measurement data that remains after the obstructed cells are removed. This data set is used for the next cue computation.

D. Distance From the Measurement Cue

For each cell in the grid, we need to compute the distance to the nearest occupied cell in the measurement grid. For that, we will use a modified version of the distance transform algorithm presented in [20]. The main issue is that we need to know not only the distance to the nearest measurement point but the distance components on the two coordinate axes as well, i.e., the row and the column. The reason for this requirement is that the standard deviations for the positioning errors are different

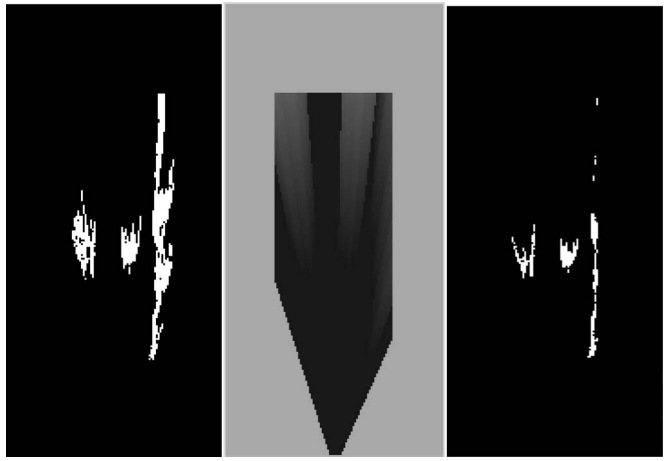

Fig. 3. Handling the occlusions. (Left) Original measurement information. (Middle) Obstruction value for each cell. (Right) Unobstructed measurement.

on the row and on the column, and therefore, one cannot be substituted for another.

Our distance transform algorithm performs like the classical two-pass L1 norm one, but instead of updating only the cell distance to the nearest measurement, the position of the nearest measurement is updated as well. The following algorithm updates a distance matrix $D(r,c)$ initialized with zero for measurement occupancy cells, and with 255 for the free cells, and two position matrices M_r and M_c that hold the row and the column of the nearest occupied measurement cell. The values of M_r and M_c are initialized to the current row and column of each cell.

Algorithm DistanceTransform
For $r = 1$ **to** $\max\_r$
  For $c = 1$ **to** $\max\_c$
    Update $(r, c, -1, 0)$
    Update $(r, c, 0, -1)$
End For
End For
For $r = \max\_r$ **to** 1
  For $c = \max\_c$ **to** 1
    Update $(r, c, 1, 0)$
    Update $(r, c, 0, 1)$
  End For
End For

Function Update (r, c, n, k)
If $D(r,c) > D(r+n, c+k) + 1$
  $\mathrm{D}(r,c) = D(r+n, c+k) + 1$
  $M_r(r,c) = M_r(r+n, c+k)$
  $M_c(r,c) = M_c(r+n, c+k)$
End If

After the distance transform algorithm is applied, the distance-to-measurement-occupied on rows and columns for each cell can be found by

$$d_{\mathrm{row}}^{\mathrm{occupied}}(r,c) = |r - M_r(r,c)|$$
$$d_{\mathrm{column}}^{\mathrm{occupied}}(r,c) = |c - M_c(r,c)|\,. \tag{14}$$

The distance-to-measurement-free-cell is computed as the difference between the double of the distance standard deviation and the distance-to-measurement-occupied, saturated to zero. Thus

$$d^{\text{free}}_{\text{row}}(r,c) = \max\left(2\sigma_{\text{row}}(r,c) - d^{\text{occupied}}_{\text{row}}(r,c), 0\right)$$

$$d^{\text{free}}_{\text{column}}(r,c) = \max\left(2\sigma_{\text{column}}(r,c) - d^{\text{occupied}}_{\text{column}}(r,c), 0\right). \quad (15)$$

These distances are converted to a probability density value using the multivariate Gaussian equation [see (16)]. For the sake of readability, we have removed the row and column arguments for all the values involved. The same equation is applied for both free and occupied distances, and therefore, the condition status is a placeholder for both situations. Thus

$$p_{\text{distance}}(m|\text{status}) = \frac{1}{2\pi\sigma_{\text{row}}\sigma_{\text{column}}} \cdot e^{-\frac{1}{2}\left(\left(\frac{d^{\text{status}}_{\text{row}}}{\sigma^{\text{status}}_{\text{row}}}\right)^2 + \left(\frac{d^{\text{status}}_{\text{column}}}{\sigma^{\text{status}}_{\text{column}}}\right)^2\right)}. \quad (16)$$

At the end of this step, we have for each cell the values $p_{\text{distance}}(m(r,c)|\text{occupied})$ and $p_{\text{distance}}(m(r,c)|\text{free})$.

VI. Weighting and Resampling

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 and 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. In addition, this means that all the particles having the same occupied hypothesis will have the same weight.

For each cell at position r, c in the grid, the weights for the free and occupied hypotheses are obtained by fusing the cues computed from the measurement data using the methods described in Section V, i.e.,

$$w_{\text{occupied}}(r,c) = p_{\text{density}}\left(m(r,c)|\text{occupied}\right) \cdot p_{\text{distance}}\left(m(r,c)|\text{occupied}\right) \quad (17)$$

$$w_{\text{free}}(r,c) = p_{\text{density}}\left(m(r,c)|\text{free}\right) \cdot p_{\text{distance}}\left(m(r,c)|\text{free}\right). \quad (18)$$

Equations (17) and (18) hold if the cell in the grid is not marked as obstructed, as described in Section V-C. If the cell is obstructed, the weights of the occupied and free hypotheses will be equal, i.e., $w_{\text{occupied}}(r,c) = w_{\text{free}}(r,c) = 0.5$.

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

$$N_{\text{OC}}(r,c) = \left|\{p_i \in S | r_i = r, c_i = c\}\right|. \quad (19)$$

The number of particles (hypotheses) having the "occupied" value false is the complement of N_{OC}. We remind the reader that N_C is the maximum number of particles allowed in a cell, and this number is a constant of the algorithm. Thus

$$N_{\text{FC}}(r,c) = N_C - N_{\text{OC}}(r,c). \quad (20)$$

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. In the following equations, we have removed the row and column parameters, but they are implied:

$$P_{\text{OC}} = \frac{w_{\text{occupied}} N_{\text{OC}}}{w_{\text{occupied}} N_{\text{OC}} + w_{\text{free}}(N_C - N_{\text{OC}})} \quad (21)$$

$$P_{\text{FC}} = \frac{w_{\text{free}}(N_C - N_{\text{OC}})}{w_{\text{occupied}} N_{\text{OC}} + w_{\text{free}}(N_C - N_{\text{OC}})}. \quad (22)$$

Aggregate 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

A classical resampling algorithm would make N_C random draws from the previous particle population of a cell while the weight of each particle controls its chances of being selected. Because we do not 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 (see Fig. 4). The key solution for a real-time operation is that all the heavy computing tasks are executed at the cell level, mostly by the use of lookup tables, whereas particle-level processing is kept very light.

Algorithm Resample
For each cell C
Compute N_{OC} and P_{OC}
Compute resampled number of particles N_{RC}
$N_{\mathrm{RC}} = P_{\mathrm{OC}} N_C$
Compute ratio between the actual number of particles and the number of resampled particles

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

End For
For each particle p_i
Find corresponding cell C
If $(f_C > 1)$—*number of particles will increase*
$F_n = \mathrm{Int}(f_C)$ Integer part
$F_f = f_C - \mathrm{Int}(f_C)$ Fractional part
For $\mathrm{k} = 1$ **to** $\mathrm{F_n} - 1$
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 $1 - f_C$ chance 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.

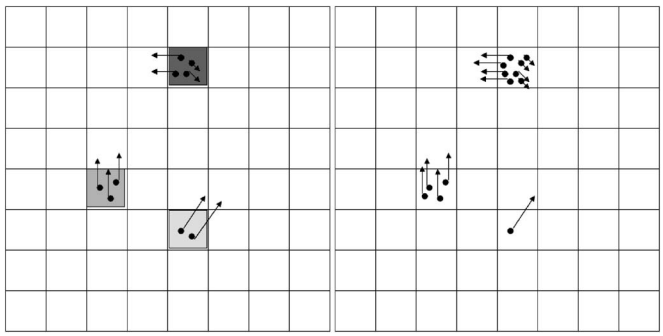

Fig. 4. Weighting and resampling. The weight of the occupied hypothesis is encoded in the darkness of the cell of the left grid.

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 start-up, 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 reinitialization.

Our solution is to use the measurement occupancy grid to create particles. If a measurement cell is of type obstacle, its $p(m(r,c)|\mathrm{occupied})$ 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 vr and vc of the created particles will be randomly sampled from an initial range of possible values, and the initial position is confined to the creation cell. This way, 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 VI) is when, due to particle drifting, the number of particles in a cell exceeds the allowed value.

VIII. Cell State Estimation and Object Extraction

The result of the tracking algorithm is the particle population itself. However, for testing and validation purposes, and to use the tracking results in further stages of processing, we will estimate the occupancy state and the speed of each cell in the grid.

The occupancy probability of each grid cell is approximated by the ratio between the number of particles in that cell and the total number of allowed particles in a cell [see (1)].

The components of the speed vector for each cell are estimated using (2). However, due to the fact that the speed of a newly created particle is completely random, these particles are excluded from the speed estimation of a grid cell. For this purpose, we can use the *age* property of the particle. The *age*

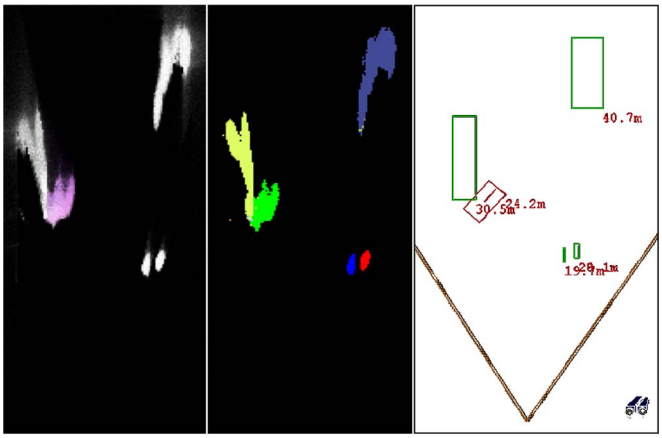

Fig. 5. Cell labeling and extraction of objects.

of the particle is set to 1 when the particle is created, and it is increased each time the particle's state (position and speed) is altered by prediction. Basically, the age of the particle tells us how many tracking cycles the particle has "survived" in the system.

All the particles in a cell that have an age higher than 2 become part of the speed estimation. They are counted, and the speed components on the row and the column are averaged. In addition, the standard deviation of these speed components is computed. If both the estimated speed components are lower in absolute value than the double of their standard deviations, the cell is declared static, because it means that either the speed is too low or it is too dispersed to draw a definite conclusion.

For further testing and evaluation, a subset of the grid cells is grouped into 3-D cuboids. A cell is considered for object grouping if its occupancy probability is at least 0.5, meaning that the particle count in the cell is at least $N_C/2$. The individual objects are identified by a generic algorithm of connected component labeling. The algorithm starts from a valid cell and recursively propagates a unique label to the cell's occupied neighbors until no more connections are found and a new label (which implies a new object) is generated. The difference between our labeling and a classical labeling algorithm is the way the neighborhood relationship is defined. Two cells are neighbors if three conditions are fulfilled.

1) The distance between them in the grid is less than 3, meaning that a one cell gap is allowed.
2) The difference in the orientation of the speed vectors in the two cells is less than 30°.
3) The difference in speed vector magnitudes is less than 30% of the value of the largest magnitude of the two cells.

The labeling process is shown in Fig. 5 (middle panel), where each color marks a different object. We can see that, by applying vicinity criteria only, the moving vehicle will be connected to the stationary structure. However, this does not happen due to the fact that we can use dynamic information provided by the grid to successfully discriminate the two objects.

The labeled connected components in the grid are used to generate the 3-D objects in the form of oriented cuboids (see Fig. 5, third panel). The objects are grouped into two categories, based on their average speed, computed from the speeds of each component cell, namely, static (shown in green) and dynamic (shown in red). Only the dynamic objects receive orientation, which is the orientation of their average speed.

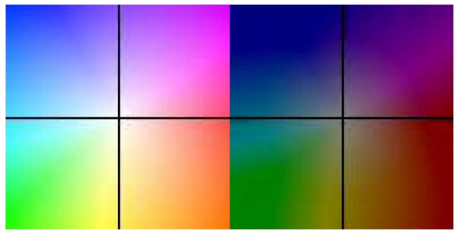

Fig. 6. Color coding for speed vectors (full and half occupancy).

IX. Tests and Results

A. Qualitative Assessment

The qualitative tests, which allow us to monitor the general behavior of the system in complex situations, are performed on video sequences recorded in real urban traffic. These tests show how the occupancy grid is computed, how the speed vector for each cell is estimated, and how the grid results are grouped into cuboidal objects having position, size, orientation, and oriented speed vector. The speed of the cells is displayed in color, using Hue for orientation and Saturation for magnitude. Due to the need for compact representation of the grid results, we have also encoded the occupancy probability as the color's Intensity, making full use of the whole hue, saturation, and intensity (HSI) color space (see Fig. 6).

Video files, describing results in different traffic situations, can be downloaded from the page http://users.utcluj.ro/~rdanescu/gridtrackingtests.htm. The main qualitative test is the sequence http://users.utcluj.ro/~rdanescu/long_sequence.avi, which shows the results over a significant distance through Cluj-Napoca. Some highlights of this sequence are shown in Fig. 7:

1) a crossing pedestrian, mixed with lateral traffic and static distant objects [see Fig. 7(a)];
2) an incoming vehicle, a static lateral scenery [see Fig. 7(b)];
3) two incoming vehicles, the most distant vehicle only visible for a couple of frames [see Fig. 7(c)];
4) a moving vehicle against a static wall, the ego vehicle performing a sharp turn left [see Fig. 7(d)];
5) a distant object, accurately tracked [see Fig. 7(e)];
6) a moving object against a static background. The protrusion from the static background near the moving object is actually an occluded stationary car. The ego vehicle is performing a sharp right turn, which causes the instability in the estimation of the static nature of the background in the top right corner. In addition, that area was previously occluded by the moving vehicle, which means that the static nature of the cells has not yet been detected due to the short observation time [see Fig. 7(f)];
7) a distant crossing vehicle going through stationary vehicles. The ego vehicle is turning right [see Fig. 7(g)];
8) tracking a moving target through a narrow corridor of stationary vehicles [see Fig. 7(h)].

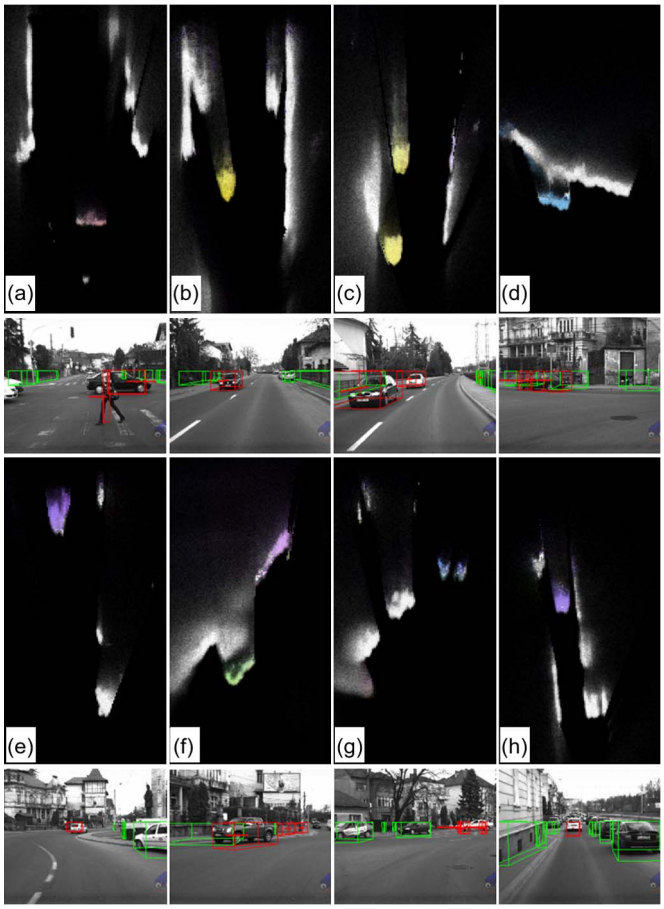

Fig. 7. Extended sequence in urban traffic—highlights.

The behavior of the system in the case of occlusions is highlighted by the sequence http://users.utcluj.ro/~rdanescu/cluj-occlusion.avi. Key points from the sequence are shown in Fig. 8. While the ego vehicle is performing a sharp right turn, a vehicle comes from our left and is occluded by a vehicle coming from our right. The occluded vehicle is also maneuvering, changing its heading to its left. While occluded, its particle distribution becomes diffuse, accounting for possible exit trajectories, and the correct heading is quickly identified as the object becomes observable again.

An extensive sequence, which was recorded while observing an intersection with the ego vehicle standing still, produced the results that are available in the file http://users.utcluj.ro/~rdanescu/wob-occlusion.avi. A highlight of this sequence is shown in Fig. 9. A vehicle comes from our left and then turns left and proceeds to exit the scene.

During this maneuver, 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.

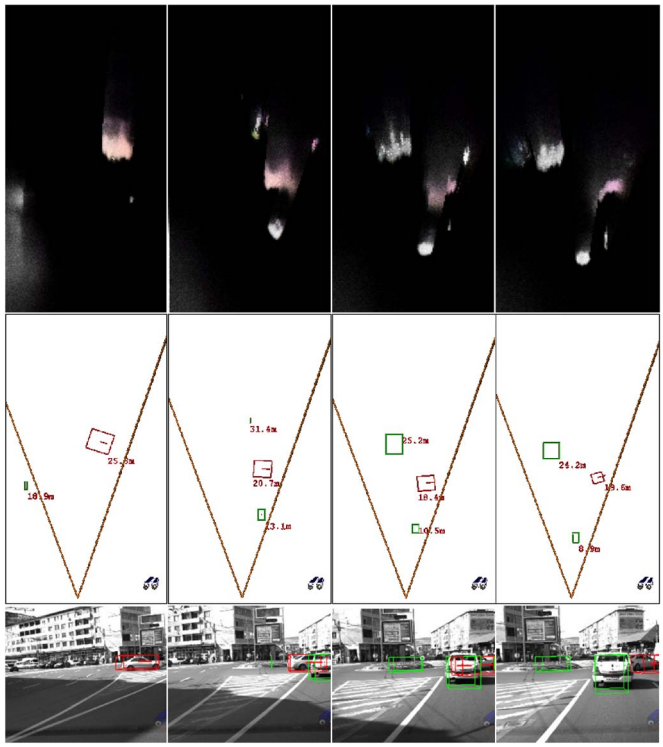

Fig. 8. Dynamic occlusion.

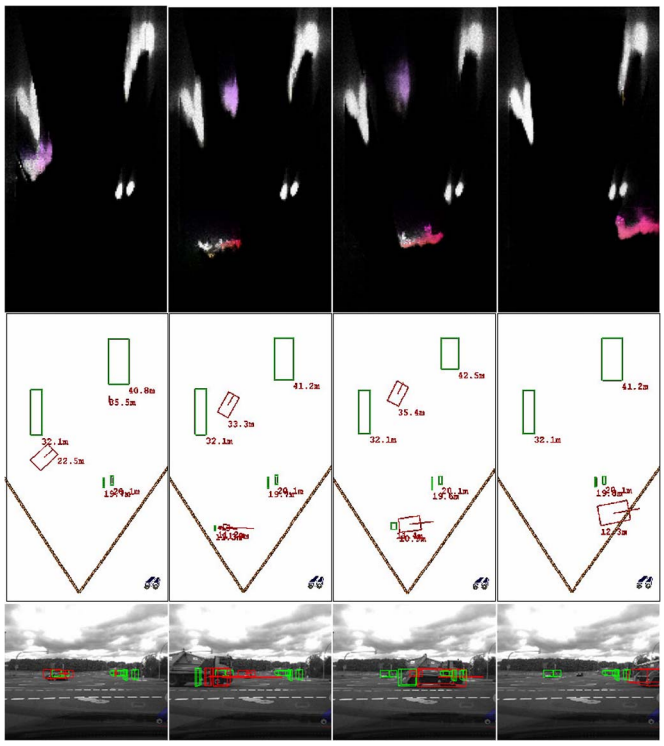

Fig. 9. Turning near a stationary object and occlusion.

B. Numerical Evaluation in a Controlled Environment

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

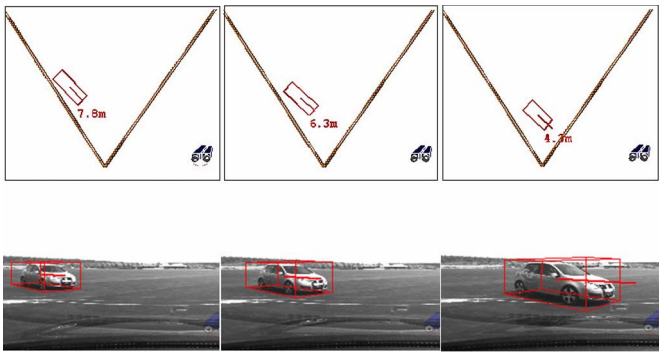

Fig. 10. Controlled test sequence.

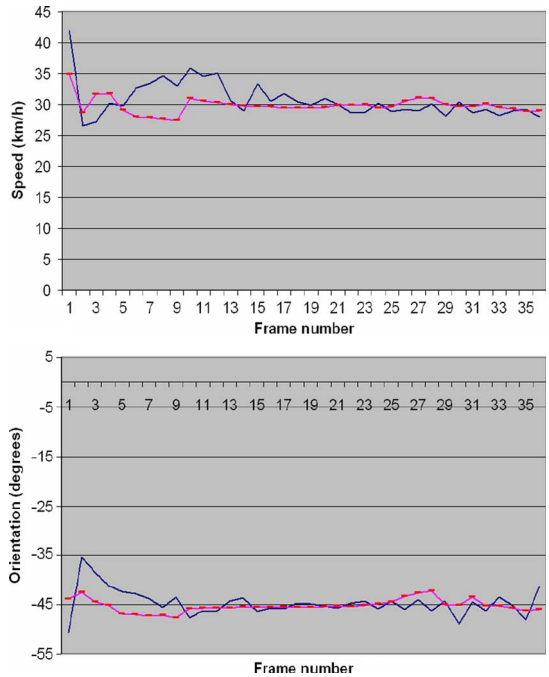

Fig. 11. Speed and orientation estimation: 30-km/h test.

truth, and they are also compared with the results of another means of intermediate extraction of 3-D dynamic information, namely, the optical flow combined with stereovision. The results of the optical flow that are taken into consideration are the speed and orientation of the 3-D cuboid obtained from grouping the points having 3-D and speed information [21]. The controlled test sequence is highly favorable to the optical flow approach, as the vehicle is clearly visible and has plenty of features that can be matched from one frame to another, which is a situation that provides plenty of good speed vectors to be averaged into an accurate vector of the cuboid (see Fig. 10).

The results of speed and orientation estimation are displayed in the graphs shown in Figs. 11–14. The grid tracking results are shown with the red dotted line. We can see that both methods quickly converge toward the ground truth, but the grid tracking results are more stable (lower error standard deviation) and more accurate (lower mean absolute error). This fact is confirmed by Tables I and II.

The time performance depends on the obstacle load of the scene, which influences the total number of particles. For a

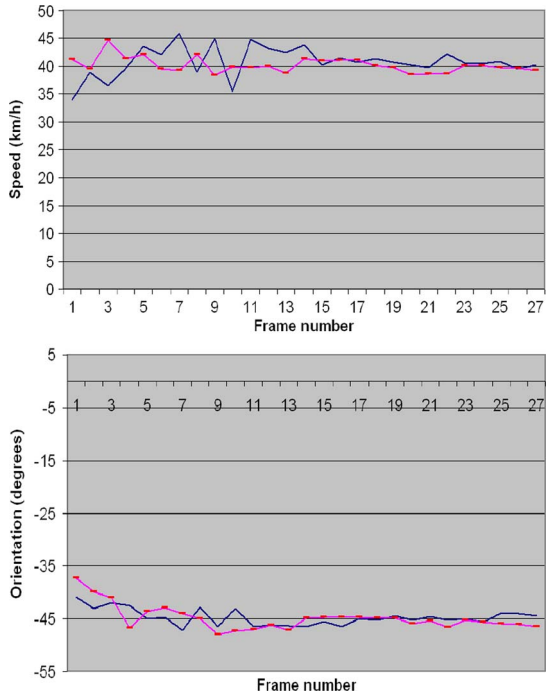

Fig. 12. Speed and orientation estimation: 40-km/h test.

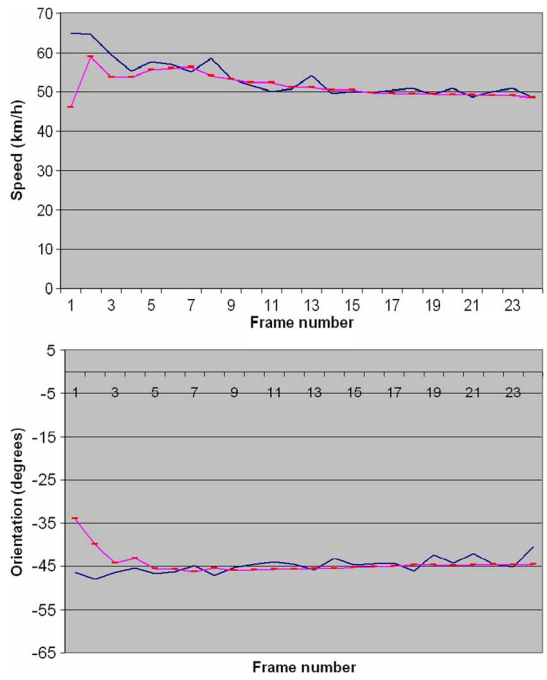

Fig. 13. Speed and orientation estimation: 50-km/h test.

typical urban scene, and a total number of particles in a cell $N_C = 50$, the total running time is about 40 ms/frame on an Intel Core 2 Duo processor at 2.1 GHz. Due to the fact that the particle tracking system shares the processor with other sensorial processing algorithms such as lane detection, object classification, and so on, the total frame rate is about 10 frames/s.

Note: Video files showing results in multiple traffic situations can be downloaded from the address http://users.utcluj.ro/~rdanescu/gridtrackingtests.htm.

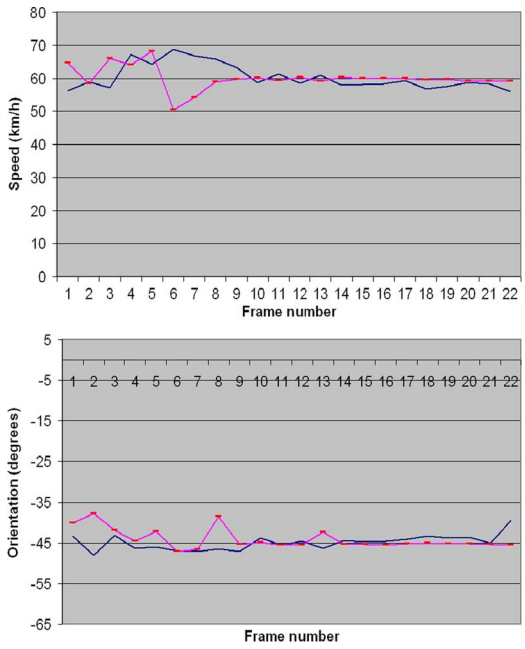

Fig. 14. Speed and orientation estimation: 60-km/h test.

TABLE I
NUMERICAL RESULTS—SPEED ESTIMATION ACCURACY

Speed of target	Particle grid MAE	Particle grid STDEV	Optical flow MAE	Optical flow STDEV
30 km/h	0.9016	0.9731	2.0141	2.3087
40 km/h	1.0184	0.9730	2.1181	1.9017
50 km/h	2.4989	2.3370	3.7329	4.4966
60 km/h	2.1279	1.3858	3.0677	2.2725

TABLE II
NUMERICAL RESULTS—ORIENTATION ESTIMATION ACCURACY

Speed of target	Particle grid MAE	Particle grid STDEV	Optical flow MAE	Optical flow STDEV
30 km/h	0.9728	0.8376	1.8219	2.0122
40 km/h	1.0321	0.8616	1.1962	1.0146
50 km/h	0.4695	0.2659	1.2775	1.1095
60 km/h	0.9343	0.6739	1.4554	1.1634

X. CONCLUSION AND FUTURE WORK

We have presented a solution for driving environment modeling and tracking, which employs particles to estimate the occupancy and speed of the cells of an occupancy grid. This flexible and real-time solution is capable of correctly tracking dynamic environments even at high relative speeds, without the need for a very high frame rate from the measurement system. The test sequences prove that the method is sensitive enough to detect and estimate not only the speed of a pedestrian but the speed of a fast moving vehicle as well. The accuracy of the speed and orientation estimation is proven by the tests conducted in controlled situations.

The particle grid tracking solution is an elegant extension of the dynamic occupancy grid solutions that were surveyed. The particle population approach relieves the designer of the choice of a speed probability distribution for each cell and can handle multiple divergent speed hypotheses. In addition, the speed distribution does not have to be estimated, and the measurement data only controls the creation or deletion of particles. We believe that the proposed technique is a new view of the occupancy grid problem, a view oriented toward practical implementation, and a view that can open the door to interesting extensions.

The presented technique is not a substitute for model-based tracking but a method for intermediate representation and processing of sensorial data. The occupancy probability and dynamic parameters of each cell can have subsequent algorithms of feature grouping, model-based object tracking, or even sensor fusion. The advantages of having a good dynamic intermediate representation are proven by the results of the experimental step of model-based object reconstruction. The quality of the particle grid tracking results as intermediate representation toward object detection and tracking are also proven by the comparison with the most used source of intermediate representation in computer vision, namely, the Lucas–Kanade optical flow mixed with stereo 3-D information, and the comparison was made in the most favorable case for the optical flow technique.

The solution leaves plenty of room for future work. For example, many of the calculations performed by the algorithm can be subjected to parallelization for a significant speed improvement. The particle-related computations, such as the prediction of the new position, can be subjected to massive parallelization, whereas the grid-related computations can be parallelized at the region level. Further work will be dedicated to the issue of optimization through parallelization.

We believe that the most important development for the future would be to use the capability of the particle to carry additional information. For example, the age information may be used for more than validation. One use of age is to adjust the variances of the random alterations of speed and position that are applied in the prediction phase—once a particle is older, its randomness can be decreased. The particles can be tagged with a unique ID, allowing us to reconstruct the trajectory of an object. Other parameters, such as height or the class of the object from which the particle is a part, can be added to the particle and be used by the tracking mechanism or by the applications developed on top of it.

ACKNOWLEDGMENT

The authors contributed equally to this work.

REFERENCES

[1] U. Franke, C. Rabe, H. Badino, and S. Gehrig, "6D-vision: Fusion of stereo and motion for robust environment perception," in *Proc 27th Annu. Meeting German Assoc. Pattern Recog. DAGM*, Vienna, Austria, Oct. 2005, pp. 216–223.

[2] D. Pfeiffer and U. Franke, "Efficient representation of traffic scenes by means of dynamic stixels," in *Proc IEEE-IV*, 2010, pp. 217–224.

[3] S. Cherng, C. Y. Fang, C. P. Chen, and S. W. Chen, "Critical motion detection of nearby moving vehicles in a vision-based driver-assistance system," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, pp. 70–82, Mar. 2009.

[4] A. Elfes, "A sonar-based mapping and navigation system," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1986, pp. 1151–1156.

[5] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun. 1989.

[6] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessiere, "Bayesian occupancy filtering for multitarget tracking: An automotive application," *Int. J. Robot. Res.*, vol. 25, no. 1, pp. 19–30, Jan. 2006.
[7] C. Chen, C. Tay, K. Mekhnacha, and C. Laugier, "Dynamic environment modeling with gridmap: A multiple-object tracking application," in *Proc. ICARCV*, 2006, pp. 1–6.
[8] T. Weiss, B. Schiele, and K. Dietmayer, "Robust driving path detection in urban and highway scenarios using a laser scanner and online occupancy grids," in *Proc. IEEE Intell. Veh. Symp.*, 2007, pp. 184–189.
[9] S. Pietzch, T. D. Vu, J. Burtlet, O. Aycard, T. Hackbarth, N. Appenrodt, J. Dickmann, and B. Radig, "Results of a precrash application based on laser scanner and short range radars," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 4, pp. 584–593, Dec. 2009.
[10] T. Gindele, S. Brechtel, J. Schroeder, and R. Dillmann, "Bayesian occupancy grid filter for dynamic environments using prior map knowledge," in *Proc. IEEE Intell. Veh. Symp.*, 2009, pp. 669–676.
[11] S. Thrun, "Learning occupancy grids with forward sensor models," *Auton. Robots*, vol. 15, no. 2, pp. 111–127, Sep. 2003.
[12] H. Badino, U. Franke, and R. Mester, "Free space computation using stochastic occupancy grids and dynamic programming," in *Proc. Workshop Dynamical Vis. ICCV*, 2007, pp. 1–12.
[13] M. S. Darms, P. E. Rybski, C. Baker, and C. Urmson, "Obstacle detection and tracking for the urban challenge," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 475–485, Sep. 2009.
[14] C. Braillon, K. Usher, C. Pradalier, J. Crowley, and C. Laugier, "Fusion of stereo and optical flow data using occupancy grids," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 2006, pp. 1240–1245.
[15] J. Y. Chen and J. Hu, "Probabilistic map building by coordinated mobile sensors," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, 2006, pp. 807–812.
[16] M. Isard and A. Blake, "CONDENSATION—Conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5–28, Aug. 1998.
[17] F. Oniga and S. Nedevschi, "Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection," *IEEE Trans. Veh. Technol.*, vol. 59, no. 3, pp. 1172–1182, Mar. 2010.
[18] I. Haller, C. Pantilie, F. Oniga, and S. Nedevschi, "Real-time semi-global dense stereo solution with improved sub-pixel accuracy," in *Proc. IEEE IV Symp.*, 2010, pp. 369–376.
[19] W. van der Mark and D. M. Gavrila, "Real-time dense stereo for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 38–50, Mar. 2006.
[20] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *J. Assoc. Comput. Mach.*, vol. 13, no. 4, pp. 471–494, Oct. 1966.
[21] C. Pantilie and S. Nedevschi, "Real-time obstacle detection in complex scenarios using dense stereo vision and optical flow," in *Proc. IEEE-ITSC*, 2010, pp. 439–444.

Radu Danescu received the Diploma Engineer, M.S., and Ph.D. degrees from the Technical University of Cluj-Napoca (TUCN), Cluj-Napoca, Romania, in 2002, 2003, and 2009, respectively, all in computer science.

He is currently a Senior Lecturer with the Department of Computer Science, TUCN, where he teaches image processing, pattern recognition, and design with microprocessors. He is also a member of the Image Processing and Pattern Recognition Research Laboratory, TUCN. His main research interests are stereovision and probability-based tracking, with applications in driving assistance.

Florin Oniga received the Diploma Engineer and M.S. degrees in computer science in 2002 and 2003, respectively, from the Technical University of Cluj-Napoca (TUCN), Cluj-Napoca, Romania, where he is currently working toward the Ph.D. degree in computer science, specializing in computer vision.

He is currently a Lecturer with the Department of Computer Science, TUCN, where he teaches image processing, pattern recognition, and computer architecture. He is also a member of the Image Processing and Pattern Recognition Research Laboratory, TUCN. His research interests include stereovision, digital elevation map processing, and vision-based automotive applications.

Sergiu Nedevschi (M'99) received the M.S. and Ph.D. 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 a Researcher. In 1998, he was appointed Professor of computer science and founded the Image Processing and Pattern Recognition Research Laboratory, TUCN. From 2000 to 2004, he was the Head of the Department of Computer Science, 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.