A Particle Based Solution for Modeling and Tracking Dynamic Digital Elevation Maps

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

***Abstract*— Digital elevation maps are simple yet powerful representations of complex 3D environments. These maps can be built and updated using various sensors and sensorial data processing algorithms. This paper describes a novel approach for modeling the dynamic 3D driving environment, the particle-based dynamic elevation map, each cell in this map having, besides height, a probability distribution of speed in order to correctly describe moving obstacles. The dynamic elevation map is represented by a population of particles, each particle having a position, a height and a speed. Particles move from one cell to another based on their speed vectors, and they are created, multiplied or destroyed using an importance-resampling mechanism. The importance-resampling mechanism is driven by the measurement data provided by a stereovision sensor. The proposed model is highly descriptive for the driving environment, as it can easily provide an estimation of the height, speed and occupancy of each cell in the grid. The system was proven robust and accurate in real driving scenarios, by comparison with ground truth data.**

***Index Terms*— digital elevation map, particle filtering, environment modeling, tracking, stereovision.**

I. INTRODUCTION

The traffic scene is a complex 3D environment, with many relevant objects, which can be difficult to model by the classic 3D oriented bounding box. Many objects do not fit well in boxes, and sometimes a more detailed description of the object's shape is required [1], or they may not be observable enough to accurately fit a box to them, and the data association problem specific to box-based tracking is not easily solved [2]. The nature of the relevant traffic objects is hugely heterogeneous: cars, pedestrians, bicycles, traffic isles, shrubbery, curbs, and many more. They can also be static or dynamic, or they can quickly change between static and dynamic. For these reasons, many researchers have tried to find generic representations, which are not bound to the object's type or nature. One such a solution is to simply store the raw sensorial points, and use them to discriminate between road and obstacles, or even for mapping, provided that these points are highly accurate, such as those delivered by a laser scanner [3]. 6D vision [4] is an ambitious attempt to model the environment as a set of 3D points, each point having its own speed vector. A more compact representation is the dynamic stixel set [5], which models the visible sides of obstacles as a set of dynamic vertical structures.

Digital elevation maps (DEM) are a simple yet powerful way of modeling complex 3D environments. The environment is represented as a 2D grid, each cell in the grid being described by its height. The digital elevation maps can be large data structures, used for terrain mapping [6], a function which makes them extremely useful for planetary exploration tasks [7], but they can also be local structures, used for robotic navigation [8], environment representation for driving assistance systems [9], or even indoor pedestrian tracking [10]. The digital elevation maps can be built in real time, using multiple types of 3D sensors, the most popular being of the laser [2] and of the stereo vision [9] [10] family. The cells of the elevation map can be then analyzed and classified into traversable, obstacles and others [8] [9].

The elevation map representation of the environment is sometimes described as having 2.5 dimensions [11], because the description is not complete – bridges and tunnels, for example, cannot be fully represented. For this reason, researchers have proposed several extensions. One of the problems of elevation maps, described in [11], is that when they are built using the average (or maximum) height of the sensorial points in each grid cell, structures such as bridges and tunnels will appear as non-traversable. Assuming the overhanging structure is of no concern, the same paper presents an optimized map building algorithm which looks for gaps in the vertical structures and generates the map of the drivable surface below. In [12], we find a further extension of the elevation map, called Multi Level Surface Map, which can successfully model the overhanging structures. The environment is organized as a 2D map, but instead of storing occupancy or height, each cell stores a set of surface patches, which are defined as Gaussian distributions of height and depth. This way, the surface under a bridge will be one surface patch, and the bridge itself another. The heights are defined by their mean and standard deviation, which are updated in a probabilistic fashion. An even more general extension, presented in [13], is the multi-volume occupancy grid, which is a probabilistic representation of height volumes for each map cell, each volume having a starting position from the ground and a height, the crucial difference being that the volume can be either occupied or empty, the occupancy being a probability value. This way, free and occupied volumes can be modeled, and uncertainty can be associated. Another solution that combines elevation and occupancy, using the uncertainty element (called

Manuscript received March 14, 2013. This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS – UEFISCDI, project number PN-II-ID-PCE-2011-3-1086.

Radu Danescu and Sergiu Nedevschi are with the Technical University of Cluj-Napoca, Computer Science Department (e-mail: radu.danescu@cs.utcluj.ro, sergiu.nedevschi@cs.utcluj.ro). Department address: Computer Science Department, Str. Memorandumului nr. 28, Cluj-Napoca, Romania.

"credibility"), can be found in [1].

Another class of solutions for representing and perceiving freeform 3D environments as 2D bird-eye view maps is the occupancy grid family. Similarly to the elevation map, the occupancy grid is also based on discrete cells, but, instead of height, it holds the probability that the cell is occupied or free - a more pressing concern for real-time robotic applications. Due to their simpler nature, and to the fact that they can be adapted to work with a wide variety of sensors, the occupancy grids are featured in multiple scientific contributions, and many sophisticated probabilistic techniques have been developed around the concept.

Maybe one of the first uses of occupancy grids, under the name of probabilistic local maps, is presented by Elfes in [14], in the context of sonar based robot navigation. Adding speed information to the environment representation can significantly increase the complexity of the probabilistic reasoning, as the grid cells are now strongly interconnected. The work of Coue et al, presented in [2], uses a 4D occupancy grid, where each cell has a position and two speed components along each axis. By estimating the occupancy of each cell in the 4D grid, the speeds for the classical cells in the 2D grid can be computed. Another solution for the representation of speed is presented by Chen et al, in [15]. Instead of having a 4D grid, this solution uses the classical 2D representation, but each cell has 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.

Several occupancy grid approaches handle the dynamic obstacle regions of the environment separately from the static regions. In [16], the Dempster-Shafer evidence theory is employed in order to update the occupancy grid from laserscanner measurement data, and the obstacle regions are identified as regions with conflicting evidence and high uncertainty. A similar approach, using stereovision instead of laser, is presented in [17]. The obstacle regions, identified from evidence uncertainty, are grouped and tracked as oriented box dynamic objects, separated from the grid. A method that computes the occupancy grid directly in the stereovision disparity space, and uses GPU parallelization for real-time performance, is presented in [18].

A combination between the digital elevation map and the probabilistic occupancy grid seems to be a natural extension of both environment modeling techniques. In [19], we find a stereovision-based solution which maintains two maps, one for occupancy and one for height, while in [20] the elevation map is used as an intermediary processing step towards achieving the occupancy grid representation.

This paper proposes a novel probabilistic solution for modeling and tracking dynamic 3D environments, which combines the elevation map's power of static 3D representation with the dynamic occupancy grid's power of cuboid-free representation of dynamic obstacles. While papers such as [12] and [13] present sophisticated techniques for probabilistic modeling of *static* 3D environments, and solutions such as [19] and [20] use occupancy grids along with elevation maps, this work describes a solution for modeling and tracking *fully dynamic elevation maps*. All cells in the map, static or dynamic, occupied or free, are updated using the same mechanism, and the tracking results can be easily estimated into static instantaneous elevation maps or dynamic occupancy grids.

The unified modeling and tracking solution is based on particles, which are not simply state hypotheses, as in classical particle-based tracking solutions, but are the building blocks of the 3D world. The particles can move from one cell to another, providing an elegant and intuitive mechanism for prediction, and can be created and destroyed based on their agreement with the measurement data (the state update). In [21], we used this particle mechanism for modeling and tracking dynamic occupancy grids. The moving particle, however, can carry many items of information: speed, position, and height. A population of particles having speed and height becomes a probabilistic model for a fully dynamic elevation map. This paper describes this original world model and its use for tracking, also highlighting the derived original contributions, necessary for transforming the concept into a working solution: the moving particle based prediction, the handling of discontinuous variations in the measurement height due to the observation vehicle's pitching, the computationally efficient integration of the stereovision measurement model in the particle weighting process, the particle resampling mechanism that reduces the particle population in a cell when these particles do not agree with the measurement.

The measurement data is an instantaneous, static elevation map constructed from dense stereo data, using a method described in [9]. However, the tracking solution is sufficiently general that any type of dense 3D sensor can be used.

The new world model and the dynamic environment tracking method based on the new model were evaluated for accuracy and reliability in describing the static and dynamic aspects of real traffic scenarios. High accuracy laser scanner data was used as ground truth, and standard metrics for dense 3D data accuracy evaluation, such as the percent of badly reconstructed heights or the root mean square error, were employed. The accuracy of the 3D reconstruction was considerably improved, in comparison to the raw map, as was the density (the number of cells having a valid state estimation). Also, the system was able to correctly estimate the speed of moving obstacles.

II. The Particle-Based Dynamic Elevation Map

An elevation map representation of a 3D scene in the coordinate system *XYZ* (consider a coordinate system with the origin on the ground in front of the vehicle, the *X* axis pointing forward, the *Y* axis pointing to the left, and the *Z* axis pointing up) is a function $Z(X,Y)$, which assigns to every point (X,Y) of the horizontal plane *XOY* a height coordinate *Z*. This continuous mapping is further approximated by dividing a finite region of the *XOY* plane into cells, each cell i being identified by a position in a finite matrix, described by a row r_i and a column c_i. A height value h_i is assigned to each cell, and thus the elevation map approximation becomes an array of height values.

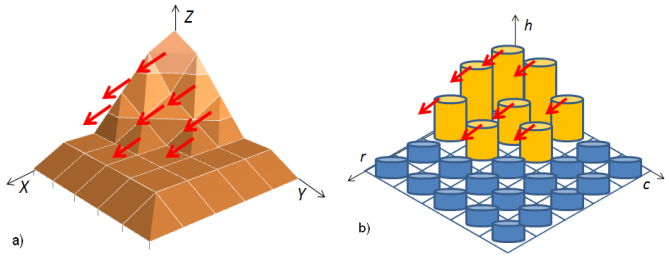

Fig. 1. a) The dynamic elevation map, a 3D surface with attached speed vectors; b) the continuous surface is approximated by a grid of fixed size cells, with heights and speeds for each cell.

If the 3D scene is dynamic, each cell of the discrete elevation map can have an assigned speed. If the application is limited to the driving scenario, it can be assumed that the objects in the scene move mainly in the horizontal plane, and the speed vector has only two components, v^x and v^y. Thus, in the continuous case we have two functions, $v^x(X,Y)$ and $v^y(X,Y)$, and in the discrete case we can speak of v^r_i and v^c_i – the row speed and the column speed for each cell i in the map, as shown in Fig. 1.

Thus, a dynamic digital elevation map can be described by three arrays of values, h_i, v^r_i and v^c_i. In an ideal world, all these values can be measured, and an accurate world description can be generated. In the real world the sensors have limited range, limited precision, limited reliability, and all these problems lead to some cells of the map to be unobservable, or to have a poor measurement of their height. All these limitations cause uncertainties, and these uncertainties have to be represented in the world model. Thus, instead of computing single values for height and speed components, the system must compute probability densities. A cell i in the dynamic elevation map is associated to a random variable $\mathbf{X}_i = (h_i, v_i^r, v_i^c)^T$, which has three dimensions (height, row speed, and column speed). The objective of the tracking algorithm is to compute the probability density of $\mathbf{X}_i$, for each cell i in the map, based on a sequence of measurements $\mathbf{Z}(0)\ldots\mathbf{Z}(t)$. The measurement $\mathbf{Z}$ includes all available sensorial data for the time instant t, not limited to the current cell.

$$p(\mathbf{X}_i(t) \mid \mathbf{Z}(0),\mathbf{Z}(1),...,\mathbf{Z}(t)) \propto p(\mathbf{Z}(t) \mid \mathbf{X}_i(t))p(\mathbf{X}_i(t) \mid \mathbf{Z}(0),...,\mathbf{Z}(t-1)) \quad (1)$$

The tracking problem is formulated as a Bayesian recursive estimation of probability densities, as described by (1). The past state density $p(\mathbf{X}_i(t-1) \mid \mathbf{Z}(0),...,\mathbf{Z}(t-1))$ and the state transition model $p(\mathbf{X}_i(t) \mid \mathbf{X}(t-1))$ are combined to form the predicted state density $p(\mathbf{X}_i(t) \mid \mathbf{Z}(0),...,\mathbf{Z}(t-1))$, and the sensorial information at time t is used to update the state through the observation model $p(\mathbf{Z}(t) \mid \mathbf{X}_i(t))$.

Assuming that only the immediate past matters (the first order Markov model assumption), the prediction for a cell i can be computed from the past estimated states of all the cells j in the grid, $p(\mathbf{X}_j(t-1) \mid \mathbf{Z}(0),...,\mathbf{Z}(t-1))$, and the dynamic model $p(\mathbf{X}_i(t) \mid \mathbf{X}_j(t-1))$.

$$p(\mathbf{X}_i(t) \mid \mathbf{Z}(0),...,\mathbf{Z}(t-1)) = \sum_j p(\mathbf{X}_i(t) \mid \mathbf{X}_j(t-1))p(\mathbf{X}_j(t-1) \mid \mathbf{Z}(0),...,\mathbf{Z}(t-1)) \quad (2)$$

The most commonly used techniques for approximating probability densities in tracking applications are the Gaussian function (mostly used in Kalman filtering) and the particle (sample) set of values. A description of the most popular solutions for representing and tracking probability density functions (PDF) can be found in [22]. The particle-based solutions are preferred when the PDF is multi-modal or its shape is not known a priori. Another reason why this work is based on particles is that a mechanism for moving them from one cell to another can be intuitively defined.

The dynamic elevation map will be described, at time t, by a set $S(t)$of particles, each particle k being described by a state vector $\mathbf{x}_k(t)$:

$$S(t) = \{\mathbf{x}_1(t),\mathbf{x}_2(t),...,\mathbf{x}_{N_S(t)}(t)\}, \text{where}$$
$$\mathbf{x}_k(t) = ({}_pc_k(t), {}_pr_k(t), {}_ph_k(t), {}_pv^c_k(t), {}_pv^r_k(t))^T \quad (3)$$

Each particle k is located in the grid cell identified by the row ${}_pr_k$, and the column ${}_pc_k$. The grid is a map of 250 rows x 120 columns, and each cell in the grid is a rectangle of 20 cm x 20 cm. Thus, the grid spans a surface of 50x24 meters in the horizontal (XOY) plane. Each particle represents a hypothesis of the state of the cell: a possible height ${}_ph_k(t)$, a possible forward speed ${}_pv^r{}_k(t)$ and a possible lateral speed ${}_pv^c{}_k(t)$, as depicted in Fig. 2. The row, column and speed of a particle are expressed as multiples of the cell size D_X and D_Y (currently 20 cm), and the height is expressed as a multiple of the height element of size D_H (currently D_H=1 cm).

Based on the particle set $S(t)$, the probability densities involved in the tracking process can be approximated. The multi-modal probability density of the state of a cell i is derived from the particles whose position ${}_pc_k$ and ${}_pr_k$ coincides with the row and column of the cell i, r_i and c_i.

The dynamic model is described by (4). Assuming that the past state of a cell j is described by the particle value $\mathbf{x}_k(t-1)$, the current state can be described by a sample drawn from a normal distribution centered in $\mathbf{f}(\mathbf{x}_k(t-1))$ and having a covariance matrix $\mathbf{Q}_i(t)$. The function $\mathbf{f}$ encodes the uniform motion model of a particle and the translation and rotation motion of the observation platform, while the uncertainty matrix encodes the possible differences between the assumed models and the real world. The equations of the motion model, used for state prediction, will be described in section IV.

$$p(\mathbf{X}_i(t) \mid \mathbf{X}_j(t-1) = \mathbf{x}_k(t-1)) \approx N(\mathbf{f}(\mathbf{x}_k(t-1)), \mathbf{Q}_i(t)) \quad (4)$$

The prediction described by (2), based on the past state and the dynamic model, will take the form of altering the position and velocity of all particles, by applying the motion model equation $\mathbf{f}$ (a process called particle drift) and adding random quantities controlled by the matrix $\mathbf{Q}_i(t)$, a process called particle diffusion.

The measurement model $p(\mathbf{Z}(t) \mid \mathbf{X}_i(t) = \mathbf{x}_k(t))$ is defined for each cell i, and depicts the probability density of the measurement $\mathbf{Z}(t)$ under the assumption that the state of the cell is described by the particle k. This density is assumed to be a normal distribution centered in $(r_i, c_i, {}_p h_k(t))$ and having an error covariance matrix $\mathbf{\Sigma}_i(t)$, which describes the uncertainty of the sensor:

$$p(\mathbf{Z}(t) \mid \mathbf{X}_i(t)) = \mathbf{x}_k(t)) \approx N((r_i, c_i, {}_p h_k(t))^T, \mathbf{\Sigma}_i(t)) \quad (5)$$

The measurement model based update, described by (1), will take the form of assigning to each particle a weight proportional to the agreement between the particle's state and the measurement. This weight is not included in the proposed world model, because as soon as the particles are weighted they are re-sampled, a process which generates a new particle population from the weighted one, the particle's weight controlling the chances of it being replicated [23]. Thus, the new, weight-free particle population encodes the updated probability density.

III. Solution Overview

The purpose of the dynamic elevation map tracking algorithm is to continuously estimate the probability density for the height and speed of each cell in the grid. As the probability densities are represented by particles, the purpose of the tracking algorithm described in this paper is to continuously update the particle population of the scene, a process driven by the measurement data. A flowchart of the tracking algorithm is presented in Fig. 3.

In the following, a short description of each functional block of Fig. 3 is presented. A detailed description of the most important algorithm blocks will be provided in the next section of the paper. The main tracking steps follow the drift-weight-resample mechanism of the particle filter variant called CONDENSATION, described in [23].

The first step of the tracking cycle is the *Particle Drift*, a process that takes the particle population resulted at the end of the previous cycle and applies the motion equations of the host vehicle and of the particles themselves in order to predict their present positions. The particles are moved from one cell to another due to the motion of the host vehicle relative to the observed scene, and due to the speed values of the particles themselves.

After drift, the particles are subjected to the process of *Diffusion*. The states of the particles (position, height, and speed) are altered by small random amounts, which reflect the uncertainties that affect the evolution of the scene in time (or the difference between how a real scene alters its state as the time passes, and the way we predict the evolution of states). *Drift* and *Diffusion* form the prediction, preparing the particle population to meet the measurement and be altered by it. The details of the prediction process are presented in section IV.A.

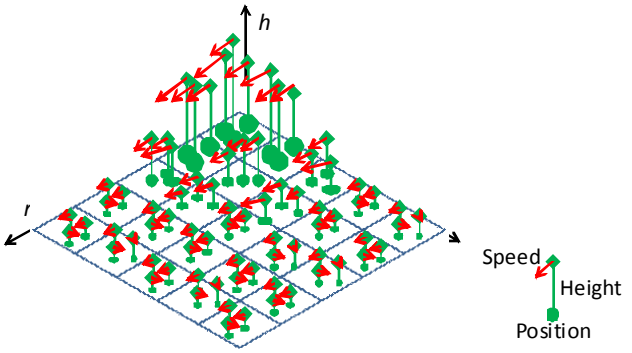

Fig. 2. The particle dynamic elevation map. Each cell has a population of particles, each particle having height and speed. The particle population can approximate a multi-modal probability distribution of heights and speeds for each cell in the map.

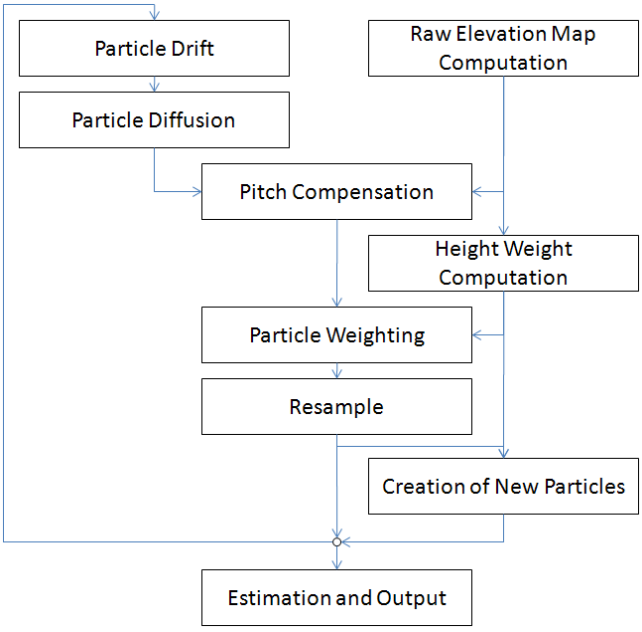

Fig. 3. Dynamic elevation map tracking algorithm.

The measurement comes in the form of a *Raw Elevation Map*, a static height map of the same size as the one that is tracked, derived directly from dense stereo information processing. This raw map is affected by the sensor-specific errors, which need to be taken into consideration. More details about this map are presented in section IV.B.

The first collision between the particle population and the measurement data is in the process of *Pitch Compensation*. The pitch angle can change quite abruptly, in an unpredictable way, due to imperfections in the road surface. Changes of this angle affect the height of the cells in the map significantly, and for this reason the system must estimate a pitch difference between frames, and adjust the particle heights to the new pitch, a method described in section IV.C.

After the pitch-based height adjustment, the particles are subjected to *Particle Weighting*. This process assigns to each particle a weight which reflects the quality of the match between the height of the particle in a specific cell and the measurement height of the raw elevation map. This process must take into account the specific uncertainties of stereovision (the observation model). For computation speedup, the

probabilistic observation model is built as a height weight look-up table for each grid cell, a process called *Height Weight Computation*. Then the particle weighting process becomes a simple assignment of a value from a Look-Up Table (LUT). Details about this process are found in section IV.D.

After each particle receives a weight based on their agreement with the measurement data, a new population of particles is generated for each cell, in the process of *Resampling*. The weight of a particle influences the chances of this particle to be selected. This way, the particles having the height closer to the height of the measurement survive and multiply, while the others are destroyed (section IV.E).

If a cell in the grid has very few particles (or none), the measurement data, already pre-processed in the form of a height weight LUT, will be used to *create new particles*, which will have random speeds, and a height distribution consistent with the height weight LUT (section IV.F).

With the updated particle population, the system is ready for the next tracking cycle. While the result of the tracking process is the particle population itself, the useful result is a dynamic elevation map, containing a height value and a speed vector for each cell. The *Estimation and Output* step will compute these values, and will generate a scene description using a popular 3D modeling language, for analysis and visualization (section IV.G).

IV. Algorithm Description

A. Particle Drift and Diffusion

The state transition probability model is implemented by the deterministic drift and the stochastic diffusion. The deterministic drift changes the state of the particles by taking into account two factors: the movement of the observing vehicle, which causes a relative movement of the whole scene in the vehicle's coordinate systems, and the movement of the mobile particles, according to their speed. The observing vehicle's movement in the horizontal, *XOY* plane, can be computed from its speed v, and its yaw rate $\dot{\psi}$, which are read from the CAN bus, and which are integrated over the time interval between two measurement frames, Δt. During this time, the vehicle rotates by an angle ψ and travels a distance d.

$$\psi = \dot{\psi}\Delta t \tag{6}$$

$$d = \frac{2v\Delta t \sin\frac{\psi}{2}}{\psi} \tag{7}$$

Taking into account the map cell size, D_X x D_Y (currently 0.2 m x 0.2 m), the origin of the coordinate systems is moved by d^c columns and d^r rows:

$$d^c = d\sin\frac{\psi}{2} \,/\, D_Y \tag{8}$$

$$d^r = d\cos\frac{\psi}{2} \,/\, D_X \tag{9}$$

Due to the motion of the observation platform, a particle k of the particle set $S(t\text{-}1)$, located at coordinates ${}_pr_k(t\text{-}1)$ and ${}_pc_k(t\text{-}1)$, will be moved to a new location, at coordinates ${}_pc_k{}^*(t)$ and ${}_pr_k{}^*(t)$ which can be computed as:

$$\begin{pmatrix} {}_pc_k^*(t) \\ {}_pr_k^*(t) \end{pmatrix} = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix} \begin{pmatrix} {}_pc_k(t-1) \\ {}_pr_k(t-1) \end{pmatrix} - \begin{pmatrix} d^c \\ d^r \end{pmatrix} \tag{10}$$

The speed vector of the particle, consisting of the two components ${}_pv^c{}_k(t\text{-}1)$ and ${}_pv^r{}_k(t\text{-}1)$, is also related to the observing vehicle's coordinate system. For this reason, when the observing vehicle rotates, the speed vector of the particle must rotate in the opposite direction, so that its direction in the scene remains unchanged. The new speed vector will have the components ${}_pv^c{}_k{}^*(t)$ and ${}_pv^r{}_k{}^*(t)$:

$$\begin{pmatrix} {}_pv_k^{c*}(t) \\ {}_pv_k^{r*}(t) \end{pmatrix} = \begin{pmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{pmatrix} \begin{pmatrix} {}_pv_k^c(t-1) \\ {}_pv_k^r(t-1) \end{pmatrix} \tag{11}$$

After the observation platform motion corrected particle positions and speeds are computed, the drift process is completed by adding the particles' motion caused by their own speed.

After drift, the particles are subjected to diffusion. The state of each particle is altered by random quantities $\delta_{\,p}c(t)$, $\delta_{\,p}r(t)$, $\delta_{\,p}h(t)$, $\delta_{\,p}v^c(t)$ and $\delta_{\,p}v^r(t)$, drawn from a normal distribution of zero mean and experimentally adjusted covariance matrix $\mathbf{Q}_i(t)$, depicting the state transition uncertainty. The complete prediction process is described as:

$$\begin{pmatrix} {}_pc_k(t) \\ {}_pr_k(t) \\ {}_ph_k(t) \\ {}_pv_k^c(t) \\ {}_pv_k^r(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}_pc_k^*(t) \\ {}_pr_k^*(t) \\ {}_ph_k(t) \\ {}_pv_k^{c*}(t) \\ {}_pv_k^{r*}(t) \end{pmatrix} + \begin{pmatrix} \delta_{\,p}c(t) \\ \delta_{\,p}r(t) \\ \delta_{\,p}h(t) \\ \delta_{\,p}v^c(t) \\ \delta_{\,p}v^r(t) \end{pmatrix} \tag{12}$$

After (12) is applied for each particle in the scene, a final step is to ensure that each cell in the grid has a number of particles less or equal to N_C, the maximum allowed number of particles in a grid cell (a constant of the system, currently 200). For this reason, if prediction assigns to a cell more particles than the maximum allowed number, excess particles are destroyed. The destruction process is random, having no preference for existing particles in the cell or for newcomers.

B. The Sensorial Information: the Raw Elevation Map

The main source of sensorial data for elevation map tracking is a dense stereovision system [24], which is able to extract 3D information for the (mildly) textured areas in the stereo image pair. The 3D points are subsequently assigned to corresponding cells in the *XOY* grid, the height of a grid cell i being the Z coordinate of the highest point assigned to the cell. The density of 3D points per cell is also computed, and used for basic validation, under the assumption that road cells will have a lower point density than obstacle cells. This validation allows the elimination of erroneously high elevation values, which are mostly caused by stereovision mismatches. A

detailed description of the raw elevation map computation technique can be found in [9]. For the elevation map tracking algorithm, the following items of information from the raw elevation map are used:

- Measurement height of each cell i, denoted by z_i. For convenience, the heights are organized as a 2D array of values that can be accessed by specifying the row and the column coordinate, thus z_i is also written as $z(r_i, c_i)$.
- Data availability for each cell, denoted by d_i. d_i=1 means that height for this cell is available, and d_i=0 means that no measurement data is available for cell i. For convenience, the d_i values are organized as a 2D array that can be accessed by specifying the row and the column coordinate, thus d_i is also written as $d(r_i, c_i)$.

Due to the fact that not all pixels in the image will obtain 3D coordinates from the stereovision engine, and not all areas in the grid are visible, due to occlusions and limited field of view, not all cells in the raw map have a measured height. The tracking algorithm is made aware of this situation by d_i. Fig. 4. shows an example of raw elevation map.

C. Pitch Angle Compensation

According to the world model described in section II, a particle k is located in the elevation grid at the row coordinate ${}_pr_k$ and column coordinate ${}_pc_k$, and has a height ${}_ph_k$. Due to the continuous nature of the observation process, the heights of the map are not expected to change abruptly from one frame to another, except when the observation vehicle performs a pitching motion. The effect of a pitch angle variation between two frames, $\Delta\alpha$, translates into a difference between the height of a particle and the measured height for the cell containing the particle:

$$\Delta\alpha_k = \tan^{-1}\left(\frac{({}_ph_k - z({}_pr_k, {}_pc_k))D_H}{D_{X\,{}_pr_k}}\right) \qquad (13)$$

Naturally, (13) is valid only when the particle is part of a static structure, and its height is close to the true height of the structure in the scene. These conditions are not valid in the general case, but we can make several assumptions:

- The vast majority of the particles in the scene belong to static structures. These structures include the road surface.
- The average height in each cell is close to the real one, even if the individual particles have significant deviations from this height.
- The changes in height due to motion in the scene are minor compared to the abrupt changes due to pitching.

Based on these assumptions, equation (13) can be averaged for all particles located in cells that have valid data, and a reasonable estimate of the pitch variation between frames can be computed.

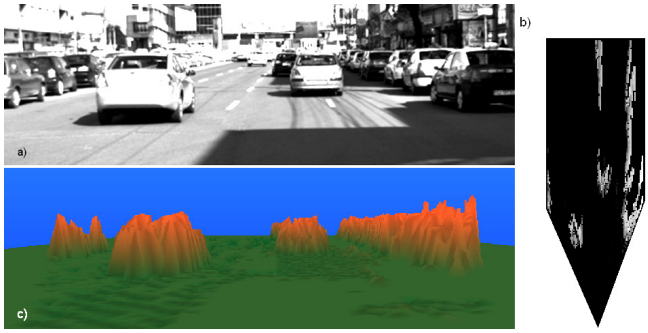

Fig. 4. The raw elevation map, extracted from dense stereovision: a) original grayscale image; b) top view of the grid, with heights encoded as grayscale values. The sensorial information covers only a part of the world map, and the areas that are not sensed are depicted in light gray; c) 3D representation of the raw map. The non-textured cells represent missing height data in the raw map, due to field of view limitations or errors of stereo reconstruction, such as those caused by dark shadows.

$$\Delta\alpha = \frac{\sum_{k=1}^{N_S} \tan^{-1}\left(\frac{({}_ph_k - z({}_pr_k, {}_pc_k))D_H}{D_{X\,{}_pr_k}}\right) d({}_pr_k, {}_pc_k)}{\sum_{k=1}^{N_S} d({}_pr_k, {}_pc_k)} \qquad (14)$$

After the pitch angle variation is estimated, the height ${}_ph_k$ of each particle k can be corrected:

$${}_ph_k = \frac{({}_ph_k D_H - \Delta\alpha\, {}_pr_k D_X)}{D_H} \qquad (15)$$

D. Height Weight Computation and Particle Weighting

The process of particle weighting is the particle filter instantiation of the measurement (observation) model $p(\mathbf{Z}(t) \mid \mathbf{X}_i(t) = \mathbf{x}_k(t))$, introduced in section II. This model describes the conditional probability density of the measurement $\mathbf{Z}(t)$ given a possible cell state value $\mathbf{X}_i(t) = \mathbf{x}_k(t)$. A particle is a state hypothesis, and the probability of the measurement given this state will be encoded as a particle weight, which will describe how well the particle hypothesis matches the measurement data.

The sensorial information, the raw elevation map, is just a conveniently modified version of the stereovision-derived 3D point data, therefore the probabilistic observation model will be derived from the observation model of stereovision, a three-dimensional normal distribution centered in the real world 3D coordinate, and having a covariance matrix defined by the distance error standard deviation σ_X, the lateral error standard deviation σ_Y and the vertical error standard deviation σ_z.

The expected error standard deviations of the three coordinates computed by a stereo reconstruction process depend on the system's baseline (distance between cameras) b, the focal distance in pixels f, and disparity computation uncertainty (matching uncertainty) σ_d. The stereo reconstruction process is seen as a non-linear transformation of the vector (u,v,d), containing a point's position (u,v) in the image space and the disparity d, into the vector of 3D coordinates (X,Y,Z).

The error is thus computed by propagating the covariance matrix of (*u,v,d*) through the Jacobian linearization of the 3D reconstruction transformation [25]. The error standard deviation for the distance coordinate *X* can be computed as:

$$\sigma_X = \frac{X^2 \sigma_d}{bf} \tag{16}$$

The error standard deviations for the lateral coordinate *Y* and for the vertical coordinate *Z* depend on the distance error standard deviation σ_X and the image position (pixel uncertainty) standard deviation σ_P.

$$\sigma_Y{}^2 = \frac{X^2}{f^2}\sigma_P{}^2 + \frac{Y^2}{X^2}\sigma_X{}^2 \tag{17}$$

$$\sigma_Z{}^2 = \frac{X^2}{f^2}\sigma_P{}^2 + \frac{Z^2}{X^2}\sigma_X{}^2 \tag{18}$$

The values of the first terms of the sums in (17) and (18) are much lower than the values of the second terms, therefore these terms can be ignored. Thus, σ_Y and σ_Z can be computed as:

$$\sigma_Y = \frac{Y\sigma_X}{X} \tag{19}$$

$$\sigma_Z = \frac{Z\sigma_X}{X} \tag{20}$$

Equations (16) to (20) refer to the 3D coordinates *X*, *Y*, and *Z* in the *camera* reference frame. In order to apply these equations to the elevation map, each cell in the map has to be associated to a 3D coordinate in the camera reference frame. A look-up table, computed offline with the help of the camera parameters (rotation and translation with respect to the world coordinate system) accomplishes this task. Also, the uncertainties for *X* and *Y* can be computed completely offline, as they are fixed for each position in the map and the height is the only thing that changes. The uncertainties of *X*, *Y* and *Z* are converted in uncertainties of row, column and height, using the following equations:

$$\sigma_r = \frac{\sigma_X}{D_X} + \sigma_{r0} \tag{21}$$

$$\sigma_c = \frac{\sigma_Y}{D_Y} + \sigma_{c0} \tag{22}$$

$$\sigma_h = \frac{\sigma_Z}{D_H} + \sigma_{h0} \tag{23}$$

The offsets σ_{r0}, σ_{c0} and σ_{h0} are added so that other measurement errors besides matching uncertainty (for example, incorrect stereo matching due to non-textured or repetitive surfaces), can be accounted for. These offsets are tuned experimentally.

After the measurement uncertainties for each cell are computed, they can be transformed into weights that will be assigned to the particles. The measurement model is depicted as a Gaussian distribution centered in the true row, column and height, having the covariance matrix formed by the three standard deviations, σ_r, σ_c and σ_h. As these standard deviations depend on the cell *i*, in the following equations they will be referred as $\sigma_{r,i}$, $\sigma_{c,i}$ and $\sigma_{h,i}$.

The classical approach for particle weighting, in this situation, is to compute the distance from the particle's position and height ($_pr_k$, $_pc_k$, $_ph_k$) to the measurements (*r, c, z*) inside an acceptable search zone, and transform this distance into a probability value using the Gaussian equation. This approach is not computationally efficient. Instead, the process of *Height Weight Computation* creates, for each cell *i* in the map, a weight LUT for all possible heights (which are represented as integers, multiples of 1 cm, thus a LUT size of 300 positions will accommodate all relevant heights found in the scene).

The creation of the weight LUT is a data-driven approach, which will approximate the computation of weight by distance to the measurement in a more computationally efficient way. For each cell *i*, an influence region of $4\sigma_{r,i}$ x$4\sigma_{c,i}$ around the central position (r_i, c_i) is analyzed. Each measured height *z* inside the search region will receive as weight the value of a Gaussian function G_i centered in (r_i, c_i), having the standard deviations $\sigma_{r,i}$ and $\sigma_{c,i}$. This is a straightforward application of the Gaussian observation model, in the horizontal plane. Formally, the histogram value for a height candidate *h*, at coordinates (r_i, c_i), for the cell *i*, is computed as:

$$H_i(h) = \sum_{\tau=r_i-2\sigma_{r,i}}^{r_i+2\sigma_{r,i}} \sum_{\kappa=c_i-2\sigma_{c,i}}^{c_i+2\sigma_{c,i}} d(\tau,\kappa) G_i(\tau - r_i, \kappa - c_i)\delta(z(\tau,\kappa) - h) \tag{24}$$

In (24), *d* is the data availability map, described in sub-section IV.B, and δ is the Kronecker delta function. The multiplication terms $d(\tau,\kappa)$ and $\delta(z(\tau,\kappa) - h)$ indicate that only the valid cells in the raw map, having the height *z* equal to the height candidate *h* will be taken into consideration.

According to the measurement model, the particle's weight will depend on its distance to the measurement along all three coordinate axes. Equation (24) only accounts for the displacement in the horizontal plane. The distance between the particle's height and one of the heights in the histogram H_i can be transformed into a probability value by using a Gaussian function. The same result can be achieved more efficiently by convolving the height histogram H_i with a 1D Gaussian kernel K_i, of standard deviation $\sigma_{h,i}$. Equation (25) transforms the sparse height histogram H_i into a continuous weight LUT, W_i:

$$W_i = K_i * H_i \quad (25)$$

A particle k, in a cell i, will get the following weight:

$$w_k = W_i(h_k) \quad (26)$$

E. Resampling

The resampling process creates a new population of particles, using the current population and their weights. This is the process that makes the particle population reflect the posterior probability of the state of the scene that is tracked, a probability which is the result of the prediction (drift and diffusion) and of the measurement (weighting).

Resampling is applied for each cell i, at each time instance t, after the particles are weighted. The total number of allowed particles for a cell is N_C, a constant which is currently set at 200. The real number of particles in the cell, $N_{R,i}$, resulted by drift and diffusion, may be lower than N_C. For re-sampling, it is assumed that the cell holds a higher number of particles, N_A = 1.25 N_C. The difference between the real number of particles in the cell, $N_{R,i}$, and the augmented maximum number of particles N_A is the number of "empty" particles, particles which are in fact empty places. The re-sampling mechanism will perform the following steps:

1. Weight the empty particles with a default low weight, which we chose to be the average value of the height weight LUT, W_i.
2. Normalize the weights of all N_A particles so that their sum becomes 1.
3. Perform N_C random extractions from the total particle population, real and empty. The weight of the particle controls its chances of being selected – high weight particles will be selected multiple times, lower weight ones may not be selected at all. If empty particles are selected, the final number of particles in the cell will be lower than N_C.

The resampling mechanism completely replaces the particle set at each measurement time t (each frame). Therefore, there is no need for particle deletion, as the particles with low weight (the unfit particles) will not be selected, and thus they will be automatically removed.

This resampling algorithm differs slightly from the classical solution [23] due to the presence of empty particles. The effects of having empty particles are the following:

1. If most of the real particles of a cell have a low weight due to their lack of fitness to the measurement data, the particle population in that cell decreases.
2. The difference between N_A and N_C ensures that even if a very good fit between the particles and the measurement data occurs, there is always a chance to make some room for new particles.

The main reason for designing the mechanism for reducing the number of particles in a cell, a mechanism that acts in an accelerated manner when the data does not fit the prediction, is the need for a mechanism to clear the way for the particles belonging to moving objects. If a vehicle moves across a road surface, the particles that have a low height must be cleared so that the particles of the moving object have a place to go. Without an accelerated elimination of the unfit particles, the convergence of the particle population to the new height of the obstacle will be slow, and the system will not react quickly enough to a dynamic scene.

F. Creation of New Particles

If the number of particles in a cell, $N_{R,i}$, is lower than $N_C/2$, and the cell i has a valid height $z(r_i,c_i)$ in the raw measurement map, the algorithm will create a number of $N_C/2$-$N_{R,i}$ new particles. The speeds of the new particles will be sampled from a normal distribution centered in 0, and the heights will be sampled from the multi-modal distribution of height measurement influencing cell i, represented by the weight LUT W_i. The process of creating new particles is applied after each resampling step.

G. Estimation and Output

If the particle population in a cell i is higher than a threshold that we set at $2N_C/3$, the height and speed of the cell can be estimated by averaging the values of all particles k that are located inside the cell, using equations (27) and (28). The operator $|S|$ denotes the cardinality of a set S. The particles involved in the estimation are those generated by resampling, therefore the particle weights are no longer needed.

$$h_i = \frac{\sum_{\mathbf{x}_k \in S,\, {}_p c_k = c_i,\, {}_p r_k = r_i} {}_p h_k}{|\{\mathbf{x}_k \in S \mid {}_p r_k = r_i,\, {}_p c_k = c_i\}|} \quad (27)$$

$$(v_i^c, v_i^r) = \frac{\sum_{\mathbf{x}_k \in S,\, {}_p c_k = c_i,\, {}_p r_k = r_i} ({}_p v_k^c,\, {}_p v_k^r)}{|\{\mathbf{x}_k \in S \mid {}_p r_k = r_i,\, {}_p c_k = c_i\}|} \quad (28)$$

The estimated dynamic elevation map is then transformed into a Virtual Reality Modeling Language (VRML) scene [26], for visual inspection, as seen in figure 5.a and 5.b. From the tracked elevation map one can also derive an occupancy grid, the occupancy probability for each cell being computed as the ratio between the number of particles having a height higher than a threshold T and the total number of particles in the cell. Figure 5.d shows the occupancy grid derived from the dynamic elevation map, for a sample threshold T=50 cm.

$$P_{Occ}(i) = \frac{|\{\mathbf{x}_k \in S \mid {}_p r_k = r_i,\, {}_p c_k = c_i,\, {}_p h_k > T\}|}{|\{\mathbf{x}_k \in S \mid {}_p r_k = r_i,\, {}_p c_k = c_i\}|} \quad (29)$$

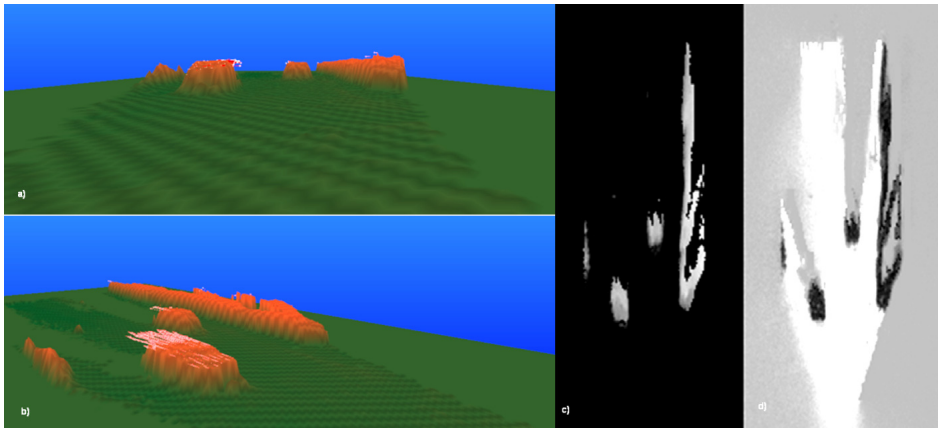

Fig. 5. Estimation results for the tracked scene: a) 3D rendering of the tracked elevation map, with speed vectors, perspective view; b) 3D rendering of the tracked elevation map, with speed vectors, lateral view; c) top view of the tracked map, with heights encoded as grayscale values; d) occupancy grid estimation from the tracked map, darker values encoding a higher occupancy probability.

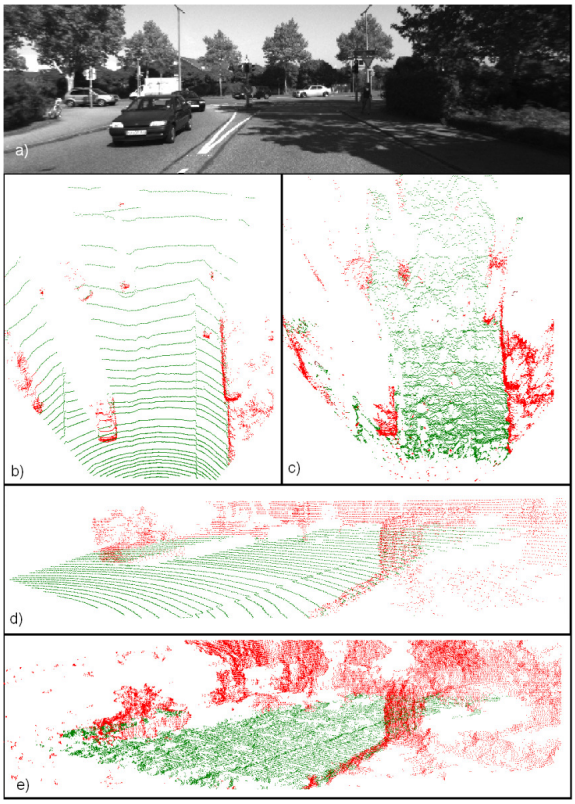

Fig. 6. A snapshot of the evaluation sequence. a) grayscale left image; b) laser 3D points, top view; c) stereo 3D points, top view; d) laser 3D points, lateral view; e) stereo 3D points, lateral view.

V. Tests and Results

A. Evaluation of Scene Reconstruction Accuracy

The most important goal of an elevation map algorithm is to accurately depict the scene being observed. For this reason, we have to compare the computed map with a ground truth map, a map generated by using a sensor that is both very precise and delivers a data density compared to that of stereo. Luckily, the Karlsruhe Institute of Technology compiled the KTTI Vision Benchmark Suite [27], a large set of data consisting of grayscale and color image pairs, Velodyne laser points, and GPS data, all synchronized and calibrated. For the evaluation process, we have performed the following steps:

1. Stereo reconstruction, using our algorithms, on the rectified image pairs of the KITTI dataset.
2. Raw elevation map computation, using the reconstructed stereo points.
3. Elevation map tracking, using the raw elevation map as measurement.
4. Raw elevation map computation, using the Velodyne points. This map is regarded as ground truth.
5. Computation of the differences between the ground truth map and the raw stereo map, and of the differences between the ground truth map and the tracked map. The differences are computed only when both the ground truth and the evaluated map have valid heights.

The error analysis presented in this section was done on the sequence 2011_09_26_drive_0009, part of the raw data sequences available for download at [28]. Fig. 6 shows a comparison between the stereo reconstructed 3D points and the laser 3D points, for a frame in this sequence.

Based on the differences between the stereo-derived maps (raw and tracked) and the laser ground truth map, we compute two types of errors:

- The *Badly Computed Heights* (BCH) percent, which is the ratio between the estimated heights that have the absolute difference from the ground truth higher than a threshold T (currently 0.15 m), and the total number of heights that can be compared to the ground truth.
- The *Root Mean Square Error* (RMSE), an average error of height estimation compared to the ground truth.
- The *Density* percent, the ratio between the number of estimated cell heights and the total number of cells that can have a height in the raw map (the theoretically observable cells).

The first two error indicators are inspired from [29], a paper which defines common accepted measures for evaluation of dense stereo algorithms, measures that we consider to be also suited for evaluation of dense elevation maps.

TABLE I
PERFORMANCE COMPARISON FOR HEIGHT ESTIMATION

Elevation Map	% Density	% BCH	RMSE (m)
Raw Map	41.12	27.99	0.19
Tracked Map	60.96	24.19	0.17

The results are presented in Table I. What is clearly visible is that tracking the elevation map provides a reduction in BCH, and in RMSE, while achieving a nearly 50% increase in density. This means that the tracked map is a much more complete description of the scene, increasing the number of cells that have a valid height, and this increase in data density comes with no precision cost, but with a slight precision gain.

The graphs in Fig. 7 show the BCH situation for different distances and heights. It is of no surprise that the errors increase with the distance, as this is an intrinsic property of stereovision. Also, the errors increase with the height being estimated, a phenomenon which we suggest is related in fact to distance uncertainties, a high structure being estimated at the wrong distance having a very large error with respect to the nearby ground.

The graphs in Fig. 8 show the RMSE situation for different distances and heights. As expected, the errors go up with the distance, and with the height.

From figures 7 and 8 it is apparent that the tracked elevation map improves the raw map results consistently, on almost all distances and heights.

B. Evaluation of Speed Estimation Accuracy

In addition to a denser and more accurate estimation of the heights of the perceived scene, the tracked elevation map adds dynamic information to this description. For each cell in the map we can estimate its speed vector magnitude and orientation using the speed vectors of its associated particles. In order to evaluate the quality of the speed estimation, we use several sequences recorded in controlled environments, with known speed of the tracked obstacles. In what follows, we present an analysis of eight sequences, four with the obstacle coming from the front at 45 degrees orientation (Fig. 9.), and four with the vehicle coming from behind us, at the same orientation (Fig. 10.). Each scenario was repeated with four different speeds, 30, 40, 50 and 60 km/h. The target vehicle's stable speed is reached outside of the observer's field of view.

Each test sequence is 2 to 5 seconds long (40 to 100 frames), and, depending on its speed, the vehicle is observed for 50 to 70% of the sequence's duration. As the scene consists only of the test vehicle and the road, we analyze the speed of the particles that have a height higher than 50 cm.

First, for each frame we construct a speed magnitude histogram, which counts the number of particles for each speed value candidate, from 0 to 100 km/h.

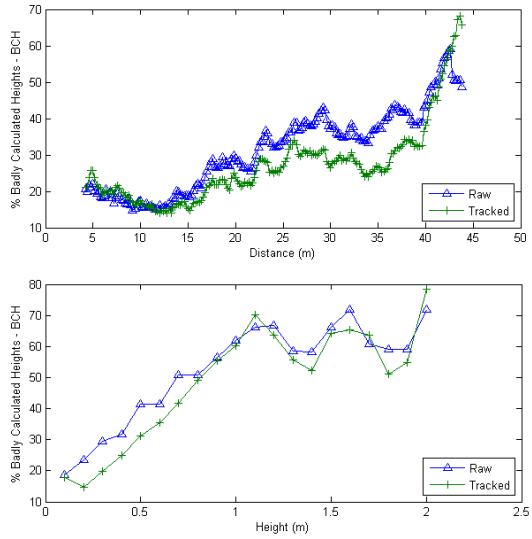

Fig. 7. Badly calculated heights (BCH) % comparison. Blue – raw map, green – tracked map. Top: BCH comparison per distance from the observing vehicle. Bottom: BCH comparison per height.

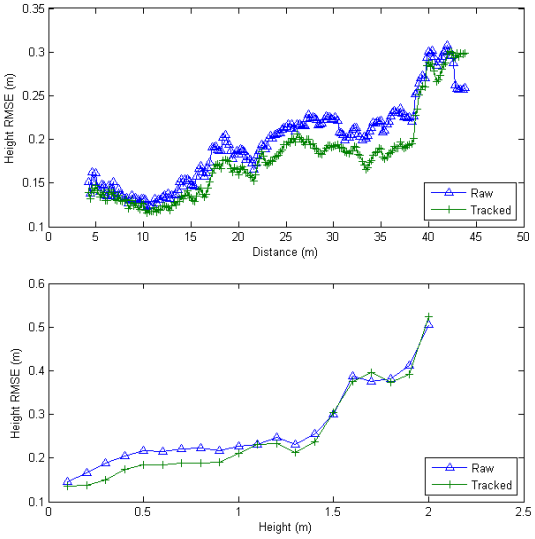

Fig. 8. Root Mean Square Error (RMSE) comparison. Blue – raw map, green – tracked map. Top: RMSE comparison per distance from the observing vehicle. Bottom: RMSE comparison per height.

Figures 11 and 13 show time sequences of these histograms, plotted as 3D surfaces, for the ground truth speeds of 30 km/h and 60 km/h (ground truth shown on the plots as the narrow vertical spike), on the two vehicle orientation scenarios. The speed value clustering behavior can be observed from these plots: as the vehicle enters the scene, the histograms become narrower around the ground truth value, and then they become diffuse again as the vehicle goes out of the field of view (but not completely out of the map).

From the speed histograms, we can make an estimation of the perceived speed of the moving obstacle. For the two orientation scenarios, we have plotted the estimated speeds against the ground truth, as shown in Fig. 12. and Fig. 14.

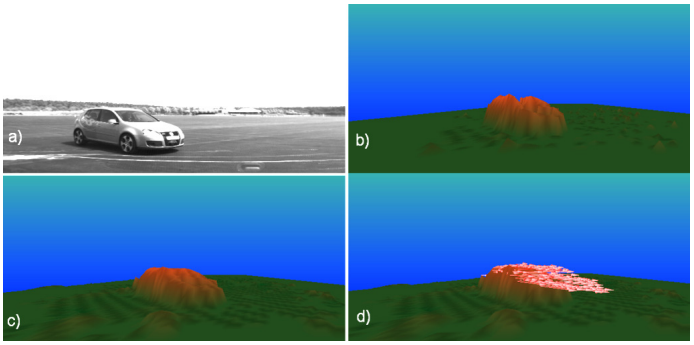

Fig. 9. Speed estimation test, with the target vehicle approaching at 45 degrees. a) left grayscale image; b) raw elevation map; c) tracked elevation map; d) tracked elevation map with speed vectors.

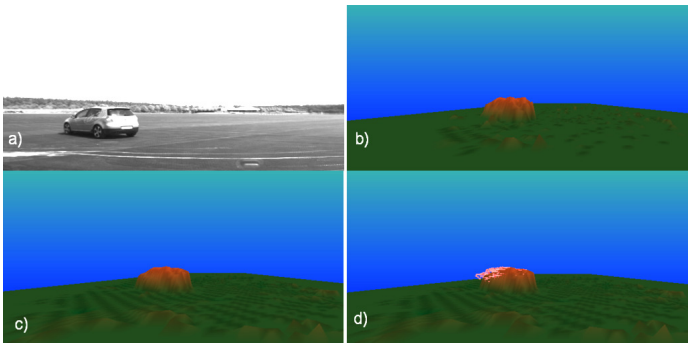

Fig. 10. Speed estimation test, with the target vehicle receding at 45 degrees. a) left grayscale image; b) raw elevation map; c) tracked elevation map; d) tracked elevation map with speed vectors.

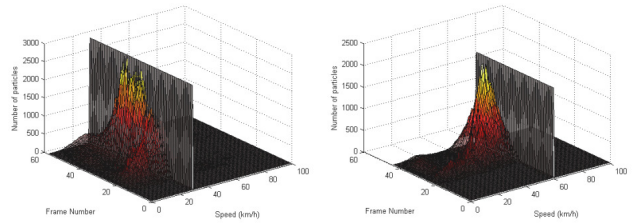

Fig. 11. Particle speed histograms, for the incoming vehicle scenarios. Top: time evolution of the histograms for the 30 km/h test speed; Bottom: time evolution for the 60 km/h test speed.

An error analysis of the speed estimation was performed per sequence, taking into consideration only those frames where the object was actually visible. The estimated mean speeds, and the root mean square errors against the ground truth, are presented in tables II and III.

The graphs in figures 11-14, and tables II and III, show that the elevation map tracking algorithm can reliably estimate the speeds of moving objects. The error of speed estimation increases with the speed of the target object, a behavior which is to be expected, as a higher speed of the object means a higher deviation from the initial average zero speed of the cell (when a cell is first populated with particles, they receive random values from a probability distribution of zero mean). The process of drift and resampling will gradually remove the wrong speeds, and multiply the correct ones, but this is not an instantaneous process. Another reason for higher errors at higher speeds is that the faster moving object is observed for a smaller period of time, meaning that the speed stable time in the graph is lower with respect to the transitional times.

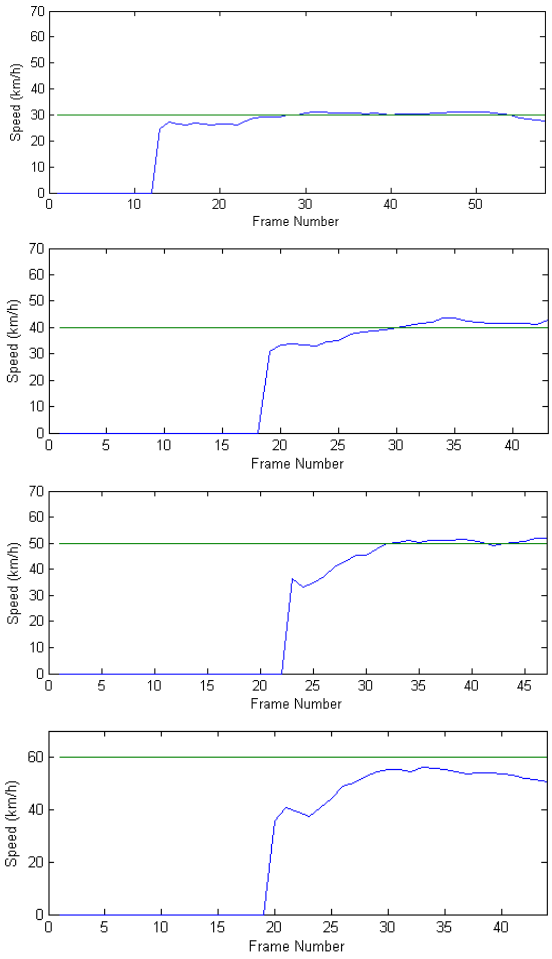

Fig. 12. Estimated object speed, for 30, 40, 50 and 60 km/h ground truth speeds, for the incoming vehicle scenario.

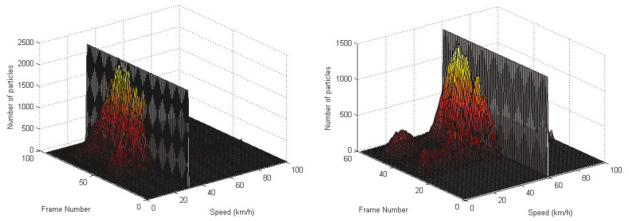

Fig. 13. Particle speed histograms, for the receding vehicle scenarios. Left: time evolution of the histograms for the 30 km/h test speed; Right: time evolution for the 60 km/h test speed.

C. Qualitative Evaluation

The elevation map tracking system has been tested on multiple sequences of real traffic scenes, acquired in Cluj-Napoca, Romania. The resulted 3D models of the perceived environment were compared to the acquired images, as we looked for: likeness of the resulted model to the real scene, speed vectors orientation and rough magnitude, describing the general motion characteristics of the objects in the scene, density of estimation, obvious errors. The system proved to be able to handle the complex traffic scenes in a robust manner, increasing the quality of the height perception of the raw stereo-derived elevation map. In figures 15, 16 and 17 some of the observed scenes are illustrated.

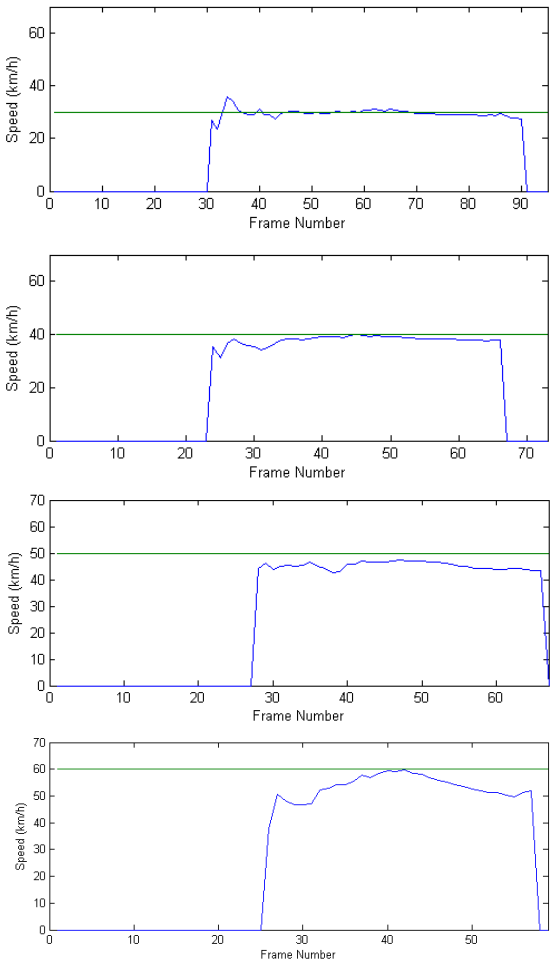

Fig. 14. Estimated object speed, for 30, 40, 50 and 60 km/h ground truth speeds, for the receding vehicle scenario.

TABLE II
SPEED MEASUREMENT EVALUATION - INCOMING VEHICLE SCENARIO

Ground truth speed (km/h)	Mean estimated speed (km/h)	RMSE (km/h)
30	29.2650	1.9720
40	38.9354	3.9316
50	46.9964	6.5184
60	50.0729	11.7318

TABLE III
SPEED MEASUREMENT EVALUATION – RECEDING VEHICLE SCENARIO

Ground truth speed (km/h)	Mean estimated speed (km/h)	RMSE (km/h)
30	29.6231	1.6149
40	37.8779	2.6842
50	45.2310	4.9515
60	53.1327	8.2875

A high resolution video file, showing the system's behavior for a two minutes driving sequence can be accessed at [30].

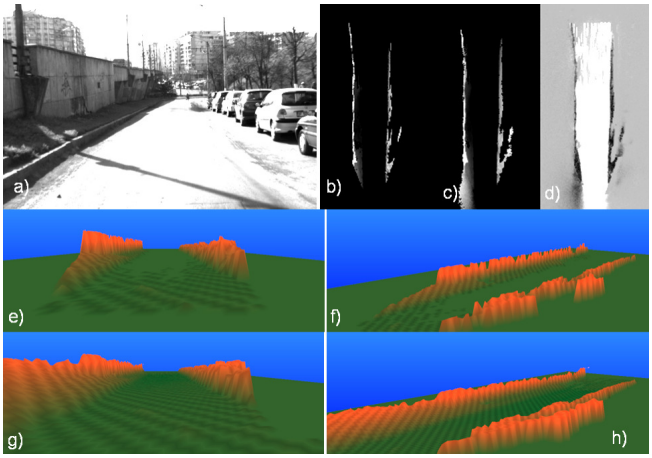

Fig. 15. Large, continuous static structures. a) left grayscale image; b) raw map, intensity encoding for height; c) tracked map, intensity encoded for height; d) occupancy grid estimation; e) raw map, 3D visualization, perspective view; f) raw map, 3D lateral view; g) tracked map, 3D perspective view; h) tracked map, 3D lateral view.

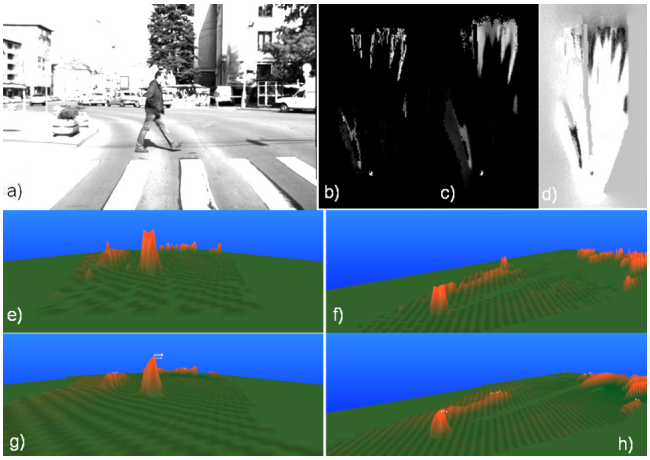

Fig. 16. Pedestrian tracking, with estimated speed vectors. a) left grayscale image; b) raw map, intensity encoding for height; c) tracked map, intensity encoded for height; d) occupancy grid estimation; e) raw map, 3D visualization, perspective view; f) raw map, 3D lateral view; g) tracked map, 3D perspective view; h) tracked map, 3D lateral view.

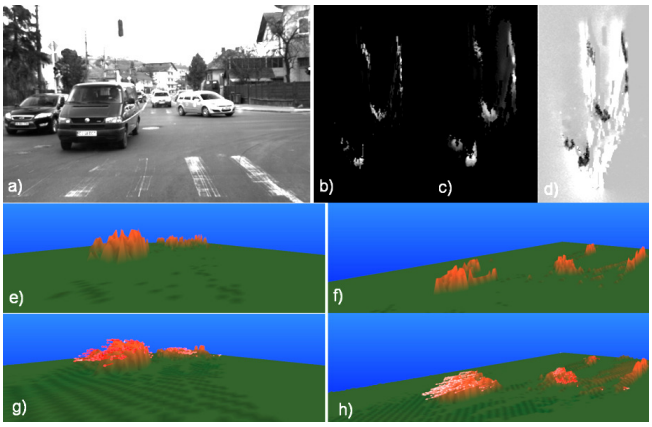

Fig. 17. Tracking vehicles at an intersection. a) left grayscale image; b) raw map, intensity encoding for height; c) tracked map, intensity encoded for height; d) occupancy grid estimation; e) raw map, 3D visualization, perspective view; f) raw map, 3D lateral view; g) tracked map, 3D perspective view; h) tracked map, 3D lateral view.

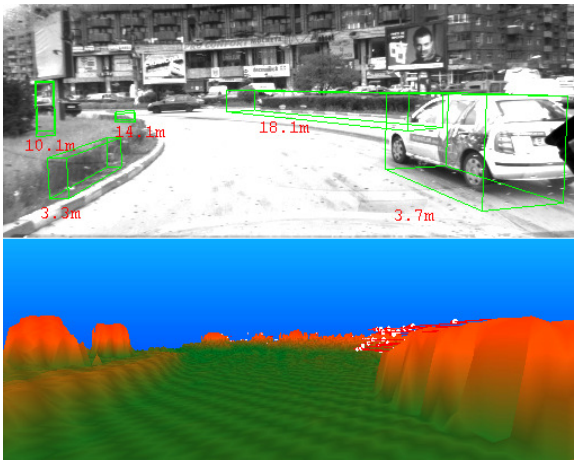

Fig. 18. Cuboid-based world perception (top) versus dynamic elevation map based perception at a roundabout (bottom).

TABLE IV
COMPARISON OF THE DYNAMIC ELEVATION MAP WITH OTHER ENVIRONMENT MODELING METHODS

Method	Flexible shape description	Speed estimation	Density	Height estimation
Oriented boxes [17]	No	Yes	Low	Yes
Elevation map[11] [12]	Yes	No	High	Yes
Dynamic occupancy grid[2] [21]	Yes	Yes	High	No
6D vision [4]	Yes	Yes	Low	Yes
Dynamic stixel [5]	Yes	Yes	Low	Yes
Dynamic elevation map	**Yes**	**Yes**	**High**	**Yes**

Fig. 18 shows a comparison between the classical cuboid-based representation of the environment and the perception based on the proposed particle-based dynamic elevation map, in a situation extremely relevant for driving assistance – navigating a roundabout. One can see that the curved nature and the low height of the roundabout are not faithfully described by the oriented boxes, while the dynamic elevation map describes the environment geometry accurately.

VI. COMPARISON OF WORLD MODELING TECHNIQUES

While the classic representation of the obstacles as 3D oriented objects remains the most popular choice for the environment perception systems, especially in the field of driving assistance, various alternatives have been proposed in the recent years, aiming to increase the flexibility of the representation of the environment and the level of perceived detail.

The method proposed in this paper is a completely dynamic probabilistic elevation map, having multi-modal probability density of the cell states. Table IV presents a non-exhaustive comparison with existing environment description methods, in terms of *flexibility* (the capability of describing free-form obstacle areas as seen from the bird-eye view), *speed estimation capability*, *density* (in the bird-eye view space) and *height estimation capability*. The table shows that the dynamic elevation map combines the advantages of the elevation map with those of the dynamic occupancy grid, being the most faithful reproduction of the real world.

VII. CONCLUSION AND OUTLOOK

This paper describes an elegant environment modeling and tracking technique, the particle-based dynamic elevation map. The building block of the proposed model is the dynamic particle, having position, speed, and height, which can populate the grid cells, can migrate between cells, and can be created, multiplied, and destroyed based on the measurement data. Using the particle set, an elegant, intuitive and easily adaptable system was designed to solve the non-parametric PDF Bayesian tracking problem for a dynamic digital elevation map. While the basic moving particle-based idea was described previously in [21], the solution presented in this paper brings significant changes, required by the challenge of the additional dimension, the height. The most important change is the type of measurement used: a raw elevation map instead of a raw occupancy grid, which requires an efficient measurement model. Also, the particularities of the vehicle environment required that the discontinuous event of vehicle pitching was taken into consideration. Coping with moving elements in an elevation map required also original alterations to the particle resampling mechanism, to reduce the particle population in a cell when the particles do not agree with the measurement, thus making room for the moving obstacle.

The resulted system unifies the dynamic cell-based world tracking specific to occupancy grids, with the power of 3D representation of elevation maps. While other contributions described in the literature combine these two types of representation, the system presented in this paper is able to track dynamic elevation maps directly, in a uniform manner, without discriminating between empty cell, occupied cell, static cell or dynamic cell. The dynamic elevation map is a general dynamic 3D world representation, which can easily be transformed into less general ones, such as the classic elevation map, or the dynamic occupancy grid, using the simple steps described in the paper.

The particle-based dynamic elevation map can easily be adapted to other sensorial data sources, such as the laserscanner. We have, in fact, tried to apply the algorithm directly to the Velodyne data that we used as ground truth, and the results look very promising. For this reason, we expect that we may use this representation as a convenient sensor fusion platform, which will combine stereo and laser.

The particle-based dynamic elevation map and the dynamic environment tracking algorithm based on this model form a generic intermediate level perception system, capable of enhancing the performance of the dense stereovision through improving its accuracy and density, and providing speed information for the elements of the scene. A better generic 3D description of the 3D environment can be applied to better extract the road surface for the purpose of lane detection, to better identify the static and the dynamic

obstacles for detection, tracking and classification, or to provide a detailed 3D map of the environment.

There is still future work to be done. The current system is, so far, not a real time algorithm, the processing time per frame being around 500 ms, on a single thread of a classical CPU processor. However, multi-threaded implementation, or even a massive parallel port to a CUDA-compatible GPU can achieve huge speedups.

REFERENCES

[1] B. Gassmann, L. Frommberger, R. Dillmann, and K. Berns, "Real-time 3D map building for local navigation of a walking robot in unstructured terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003, pp. 2185-2190 vol.3.

[2] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessière, "Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application," *INT J ROBOT RES,* vol. 25, pp. 19-30, January 1, 2006.

[3] C. Brenneke, O. Wulf, and B. Wagner, "Using 3D laser range data for SLAM in outdoor environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003, pp. 188-193 vol.1.

[4] U. Franke, C. Rabe, H. Badino, and S. K. Gehrig, "6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception," in *DAGM-Symposium* vol. 3663, ed: Springer, 2005, pp. 216-223.

[5] D. Pfeiffer and U. Franke, "Efficient representation of traffic scenes by means of dynamic stixels," in proc. *IEEE Intelligent Vehicles Symposium (IV 2010)*, 2010, pp. 217-224.

[6] D. F. Huber and M. Hebert, "A new approach to 3-D terrain mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99)* 1999, pp. 1121-1127 vol.2.

[7] M. Vergauwen, M. Pollefeys, and L. Van Gool, "A stereo-vision system for support of planetary surface exploration," *MACH VISION APPL,* vol. 14, pp. 5-14, 2003.

[8] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila, "Autonomous Rover Navigation on Unknown Terrains: Functions and Integration," *INT J ROBOT RES,* vol. 21, pp. 917-942, October 1, 2002.

[9] F. Oniga and S. Nedevschi, "Processing Dense Stereo Data Using Elevation Maps: Road Surface, Traffic Isle, and Obstacle Detection," *IEEE T VEH TECHNOL,* vol. 59, pp. 1172-1182, 2010.

[10] M. Harville, "Stereo person tracking with adaptive plan-view templates of height and occupancy statistics," *IMAGE VISION COMPUT,* vol. 22, pp. 127-142, 2004.

[11] P. Pfaff, R. Triebel, and W. Burgard, "An Efficient Extension to Elevation Maps for Outdoor Terrain Mapping and Loop Closing," *INT J ROBOT RES,* vol. 26, pp. 217-230, February 1, 2007.

[12] R. Triebel, P. Pfaff, and W. Burgard, "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)* 2006, pp. 2276-2282.

[13] I. Dryanovski, W. Morris, and X. Jizhong, "Multi-volume occupancy grids: An efficient probabilistic 3D mapping model for micro aerial vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, 2010, pp. 1553-1559.

[14] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer,* vol. 22, pp. 46-57, 1989.

[15] C. Chen, C. Tay, C. Laugier, and K. Mekhnacha, "Dynamic Environment Modeling with Gridmap: A Multiple-Object Tracking Application," in *9th International Conference on Control, Automation, Robotics and Vision (ICARCV '06)*, 2006, pp. 1-6.

[16] J. Moras, V. Cherfaoui, and P. Bonnifait, "Moving Objects Detection by Conflict Analysis in Evidential Grids," *2011 IEEE Intelligent Vehicles Symposium,* pp. 1122-1127, 2011.

[17] T. N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M. M. Meinecke, "Stereo-Camera-Based Urban Environment Perception Using Occupancy Grid and Object Tracking," *IEEE T INTELL TRANSP,* vol. 13, pp. 154-165, Mar 2012.

[18] M. Perrollaz, J. D. Yoder, A. Negre, A. Spalanzani, and C. Laugier, "A Visibility-Based Approach for Occupancy Grid Computation in Disparity Space," *IEEE T INTELL TRANSP,* vol. 13, pp. 1383-1393, Sep 2012.

[19] M. Kumar and D. P. Garg, "Three-Dimensional Occupancy Grids With the Use of Vision and Proximity Sensors in a Robotic Workcell," *ASME Conference Proceedings,* vol. 2004, pp. 1029-1036, 2004.

[20] H. Lategahn, W. Derendarz, T. Graf, B. Kitt, and J. Effertz, "Occupancy grid computation from dense stereo and sparse structure and motion points for automotive applications," in *IEEE Intelligent Vehicles Symposium (IV 2010)* 2010, pp. 819-824.

[21] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and Tracking the Driving Environment With a Particle-Based Occupancy Grid," *IEEE T INTELL TRANSP,* vol. 12, pp. 1331-1342, 2011.

[22] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE T SIGNAL PROCES,* vol. 50, pp. 174-188, 2002.

[23] M. Isard and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking," *INT J COMPUT VISION,* vol. 29, pp. 5-28, 1998.

[24] C. D. Pantilie and S. Nedevschi, "SORT-SGM: Subpixel Optimized Real-Time Semiglobal Matching for Intelligent Vehicles," *IEEE T VEH TECHNOL,* vol. 61, pp. 1032-1042, 2012.

[25] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*: Mit Press, 1993.

[26] WWWConsortium. (January 3). *VRML Virtual Reality Modeling Language*. Available: http://www.w3.org/MarkUp/VRML/

[27] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012)*, 2012, pp. 3354-3361.

[28] A. Geiger. (2013). *The KITTI Vision Benchmark Suite - Raw Data*. Available: http://www.cvlibs.net/datasets/kitti/raw_data.php

[29] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *INT J COMPUT VISION,* vol. 47, pp. 7-42, 2002.

[30] R. Danescu. (2013). *Dynamic Elevation Map Tracking Demo Video*. Available: http://users.utcluj.ro/~rdanescu/dynamap.avi

Radu Danescu (M'11) 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 an Associate Professor 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.

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 from 2004 to 2012 the Dean of the Faculty of Automation and Computer Science. Prof. Nedevschi is now Vice-President in charge with scientific research of TUCN. He has published hundreds of scientific papers and has edited multiple volumes, including books and conference proceedings. His research interests include Image Processing, Pattern Recognition, Computer Vision, Intelligent Vehicles, Signal Processing, and Computer Architecture.