From Particles to Self-Localizing Tracklets: a Multi-Layer Particle Filter based Estimation for Dynamic Grid Maps

Andrei Vatavu, *Member, IEEE*, Melissa Rahm, Suresh Govindachar, Gunther Krehl, Abhishek Mantha, Sagar Ravi Bhavsar, Manuel Schier, Janis Peukert, and Michael Maile

***Abstract*—One of the indispensable functions of a self-driving vehicle is to estimate its dynamic world, which includes various traffic participants within complex driving scenarios. The estimation mechanism has to be flexible, fast and robust. However, achieving these requirements is still challenging. Dynamic grid maps are one of the possible ways to combine and estimate the multi-sensory information at an intermediate level. In this paper we present a particle filter-based grid map estimation which addresses several challenges. First, we propose multi-layer particle filter-based tracking solution (MLPT) that uses two measurement grid channels as input: an occupancy grid and a semantic grid. Second, we introduce the concept of structuring the particle population into batches, where each batch represents an individual tracklet. Rather than using one particle filter for estimating the entire grid, we employ multiple individual particle filters (tracklets) that share the same world. Third, a concept of "self-localizing" tracklets is presented. Similar to simultaneous localization and mapping approaches, in our tracking solution every particle state is extended with a small set of landmarks. This allows a tracklet to "self-localize" itself with respect to tracked object boundary and leads to a more precise velocity estimation. Finally, we introduce an advanced tracklet management mechanism that allows executing some specific particle filter operations at the tracklet level. This optimization provides multiple advantages. Experimental results with ground-truth data show improvement in the estimation accuracy in comparison to similar techniques.**

***Index Terms*—Particle Filter, Grid Map, Tracklets, Autonomous Driving, Object Tracking, Self-Localizing Tracklets.**

I. Introduction

One of the key requirements of an autonomous vehicle is the ability to accurately perceive the environment along with its complex infrastructure. The process of acquiring knowledge about a vehicle's surrounding world is a basic prerequisite for advanced automated driving functions, including collision avoidance, path planning and motion control. To reliably model the environment, an estimation system must address the following challenges: support large amounts of varying measurements from different perception subsystems; decouple dependence on specific sensor configurations; synchronize and filter input measurement data; detect and track critical traffic participants within an autonomous vehicle's drivable field-of-view.

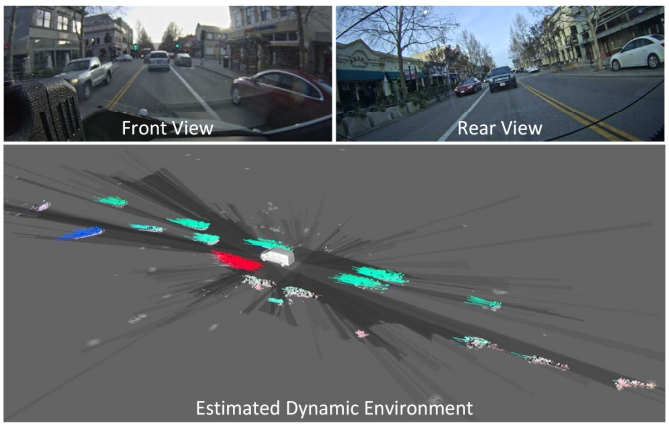


Fig. 1. Multi-Layer Particle Filter based Tracking (MLPT) for Dynamic Grid Maps. Top left and right images illustrate a snapshot from the traffic scene, provided by vehicle documentation cameras. The result of estimating dynamic objects is represented as a set of tracklets (visible as speed vectors in the image). The tracklet colors describe moving direction of the target objects. The color saturation encodes the velocity magnitude (white is for stationary objects).

In order to obtain a consistent representation in various complex scenes, obstacle detection and tracking algorithms rely on information acquired from different sensors such as radars (for directly estimating the object motion [3]), LiDAR (for a more precise obstacle position and shape [5], [6], [7]), and vision (for acquiring both range and semantic information [8], [9], [10]). Moreover, to generate a deeper understanding of the world, existing architectures rely on combinations of sensors to offset the shortcomings of individual sensor modalities [10], [11], [12].

Various techniques for modeling and tracking traffic participants have been proposed over the past years [1], [2]. Existing approaches can be divided into two main abstraction levels. Higher level abstraction methods focus on object-level processing. In most solutions, the objects are represented by simple and intuitive models such as boxes [13] or L-shapes [5]. In order to increase the robustness, various techniques try to include models with adaptive geometry such as boxes with

Manuscript received May 15, 2019; revised February 2, 2020.

The authors are with Mercedes-Benz Research and Development North America, Autonomous Driving Department. Address: 309 N Pastoria Ave., Sunnyvale, CA 94085, USA (e-mail: andrei.vatavu@daimler.com).

variable size [6], parametrized curves [14], deforming polygons [15], [16], rigid objects described by point sets [18], voxels [19]. However, object-level processing comes with its own limitations. As more sensors contribute to perceive the same world or when input measurements become noisy and cluttered, common functions, such as object-level data association, matching and grouping, require additional approaches to deal with issues like incorrectly grouped targets or inaccurate state parameter estimations. Therefore, in order to overcome these object-level challenges and reduce computational complexity, many solutions focus on modeling the dynamic world at a lower level of abstraction such as the ones described in [8],[18].

A well-known technique that has been used for lower level representation and tracking is grid-mapping. In a grid map, the surrounding space is tessellated into rectangular grid cells. Each cell is an independent building block of the space that stores properties like occupancy probability [20]. The traditional approaches incrementally update the cell occupancy values, assuming that the surrounding environment is static.

Additional approaches take advantage of the temporal inconsistencies between occupied and free space. For example, in [21], an object is labeled as "moving" if its location was previously sensed as free space. Other approaches explicitly consider the uncertainties to build the dynamic map of the environment. In [22] a Bayesian Occupancy Filter is proposed, where each grid cell state is represented by its occupancy and 2D velocity components.

More recent approaches propose different dynamic grid map estimation variants based on particle filters [4], [23], [25], [27], [29], [30], [31]. The grid cell state is approximated by a set of samples. Particles are not associated permanently to one grid cell state but are propagated according to their own motion model and, subsequently, are reassigned to new destination cells. Then, all particles receive a weight proportional to the occupancy belief of the new destination cell. When the new target cells are sensed to be occupied, higher weights are assigned to those particles. In general, a particle predicted in the middle of a larger object can be assigned to any of the occupied cells and still be rewarded with a high weight. In other words, the particles receive their weights without being aware of their own position with respect to the tracked object hypothesis. This results in higher uncertainties and slower algorithm convergence (see Fig. 5, top). In order to improve the estimation accuracy, especially for larger objects, we would need to employ extra features from various sensors or increase the number of particles. Therefore, a new challenge arises: what is the best way to incorporate more knowledge about the surrounding world in the particle state, increase the system flexibility, and keep a fixed memory space for each particle?

Starting from our previous work, presented in [17], we address the above described problems and present four concepts to improve the particle filter-based grid map estimation.

The first concept treats the dynamic grid-map estimation as a multi-channel measurement processing. We propose a **M**ulti-**L**ayer **P**article Filter based **T**racking solution (MLPT) that uses two evidence grids as input: an *occupancy grid* map channel, which contains the occupancy belief for each individual cell, and a *semantic grid* map channel, in which every grid cell is described by the most likely object class label.

The second concept adopts the idea of structuring the entire particle cloud into smaller independent batches of particles, where each batch represents an independent density. We refer to this batch of particles as a tracklet (see. Fig. 1). Instead of using one particle filter for estimating the entire grid, we employ multiple individual particle filters (tracklets) that share the same grid world and describe the state of individual object parts. Although particles from different tracklets can travel between different cells and share the same measurement information, they are linked to their own uniquely identified group (i.e. tracklet).

The third concept introduces tracklets that can "self-localize" themselves with respect to tracked object boundary, by extending the particle state with additional knowledge of the object shape. Like simultaneous localization and mapping approaches such as FastSLAM [32], in the proposed tracking solution, every particle model is extended with a small set of landmarks that are randomly selected from the object contour (see Fig. 4). Subsequently, the "anchored" particles contribute to a more precise tracklet state estimation (see an intuitive illustration in Fig. 5, bottom).

Finally, the last concept describes an advanced tracklet management mechanism. Operations such as resampling, initialization and removal are moved to the tracklet level. On the one hand, this allows iterating over a fewer number of tracklets. On the other hand, some processing steps are done in batches by manipulating individual subsets of particles belonging to one single tracklet. Specifically, we adopt a faster, two-level sampling mechanism, including both tracklet and particle levels. A priority-based mechanism is used to sample new tracklets from the measurement space, in order to keep the number of tracklets bounded.

The above proposed concepts provide the following advantages:

- Less memory consumption: Common grid cell properties (e.g. semantic label, object ID etc.) can be stored at the tracklet level instead being explicitly replicated at the particle level.
- Faster processing: Tracking operations (creation, deletion or update) work on groups of particles at the tracklet level instead of processing each single particle individually.
- Increased accuracy: Because each particle is "anchored" to a fixed sub-set of contour features, we obtain a decreased effect of "drifting" tracklets and, consequently, more precise velocity estimation, especially in the case of large objects described by uniform occupancy values.
- Improved robustness: Having individual particle filters can help in the cases when only some parts of the grid map receive new measurements. Only the tracklets that are associated to those specific measurement cells can be selectively updated, instead of processing the entire population of particles.
- Increased flexibility: Having data organized in batches can facilitate the development on parallel hardware architectures.

TABLE I
COMPARISON OF GRID-BASED PARTICLE FILTER SOLUTIONS

	Grid-based Particle Filter (PF) variants	Sensors	Occupancy cell estimation	Estimated result
[23]	Particles for estimating static and dynamic cells	Stereo-vision sensor	Number of particles in the cell	Particle-based occupancy grid
[4]	Particles for estimating static and dynamic cells	LiDAR and radar sensors	Dempster-Shafer theory of evidence	Dynamic occupancy grid
[24]	Particles for estimating dynamic cells only	LiDAR sensor	Extended Bayesian occupancy filter	Dynamic Occupancy grid
[25]	Particles for estimating static and dynamic cells	LiDAR and radar sensors	Dempster-Shafer theory of evidence	Dynamic occupancy grid
[26]	Particle population is divided into subsets of static and dynamic particles	Multi-layer laser scanners and short-range radar sensors	Dempster-Shafer theory of evidence	Evidential dynamic occupancy grids
[27]	Particles for estimating dynamic cells only	Laser scanners and short-range radar sensors	Dempster-Shafer theory of evidence	Evidential occupancy grid with radial velocity
[29]	Particles for estimating static and dynamic cells	Stereo-vision sensor	Ratio between the number of particles having a height > T and the total number of particles in the cell	Particle-based dynamic elevation map
[30]	Multiple Rao-Blackwellised particle filters for estimating static and dynamic cells	Stereo-vision sensor	Binary Bayes Filter for every particle	Dynamic occupancy grid with grayscale intensity information
Proposed Solution	Multiple individual Rao-Blackwellised particle filters (up to 128000) – self-localizing tracklets	LiDAR, stereo-vision, radar	Dempster-Shafer theory of evidence at the particle level	Dynamic occupancy grid and tracklets

The rest of the paper is structured as follows. The next section gives an overview of existing grid-based particle filter solutions. Section III describes the overall system processing flow, the concepts of self-localizing tracklets and multi-channel grid map estimation. The MLPT algorithm and its main steps including the tracklet estimation and tracklet management are described in Section IV. Section V presents the experimental results, followed by the conclusions in Section VI.

II. RELATED WORK

This section presents an overview of existing grid-based particle filter solutions, provides a comparison and points out the limitations.

A first grid-based particle filter solution is introduced by Danescu et. Al, in [23]. The entire dynamic grid is represented by a population of particles. Each particle represents an individual hypothesis which can move from cell to cell. The particle state is defined by position and speed and is predicted across the grid depending on its motion model and motion parameters. Particles are directly associated to grid cells based on their position and thus contribute to the grid cell's occupancy and velocity distribution. The occupancy probability is described by the number of particles in that cell. The measurement data used in [23] is a raw obstacle grid obtained by processing the stereovision-generated elevation map.

Another grid-based particle filter solution is described in [4], where the dynamic grid map estimation is formulated as a Random Finite Set problem. Techniques from the field of finite set statistics like the hypothesis density filter (PHD) and the Bernoulli filter (BF) are applied to estimate the dynamic state of grid cells. The Dempster-Shafer theory of evidence (DS) is then used to update the occupancy state of a dynamic cell. The used measurement data is obtained by laser and radar sensors. While solutions proposed in [4] and [23] uniformly estimate the full space including both static and dynamic components, other approaches focus on estimating static and dynamic parts separately [24], [25], [26], [27]. In [24] and [27] the particles are only associated to dynamic fields. The static infrastructure is derived by a traditional occupancy grid update mechanism. Therefore, in [24] a new representation of the Bayesian Occupancy filter (BOF) is used. Alternatively, in [25] and [26] the world model is represented by two distinct sets of particles, i.e., static (particles with zero velocity) and dynamic. In [27], the static and dynamic occupancies are directly updated by the map. The authors use dynamic particles only to estimate cell velocity distributions and to predict the dynamic occupancy evidence of the map. The input measurements are provided by laser scanners and short-range radar sensors.

In [29], [30] and [31] the particle state is extended to incorporate multiple observation cues and adopt various heuristics to improve data association and processing time. Depending on the architecture, world model and sensor setup, the most recent algorithms enrich the particle state with additional properties like object heights [29], grayscale intensities [29], [30], patches [30], and object IDs [28], [31]. For example, in [29], the particle state is extended with a new dimension, the height. The authors in [30] use two grid maps – an intensity grid and an occupancy grid. The two grids are used to extract a set of 3x3 rectangular patches. To estimate the state of these patches, a Rao-Blackwellised solution is used, where each particle incorporates the intensity and occupancy information and receives a weight based on how well the particle patch matches the extracted measurement. Although the patch-based approach can include the extra appearance

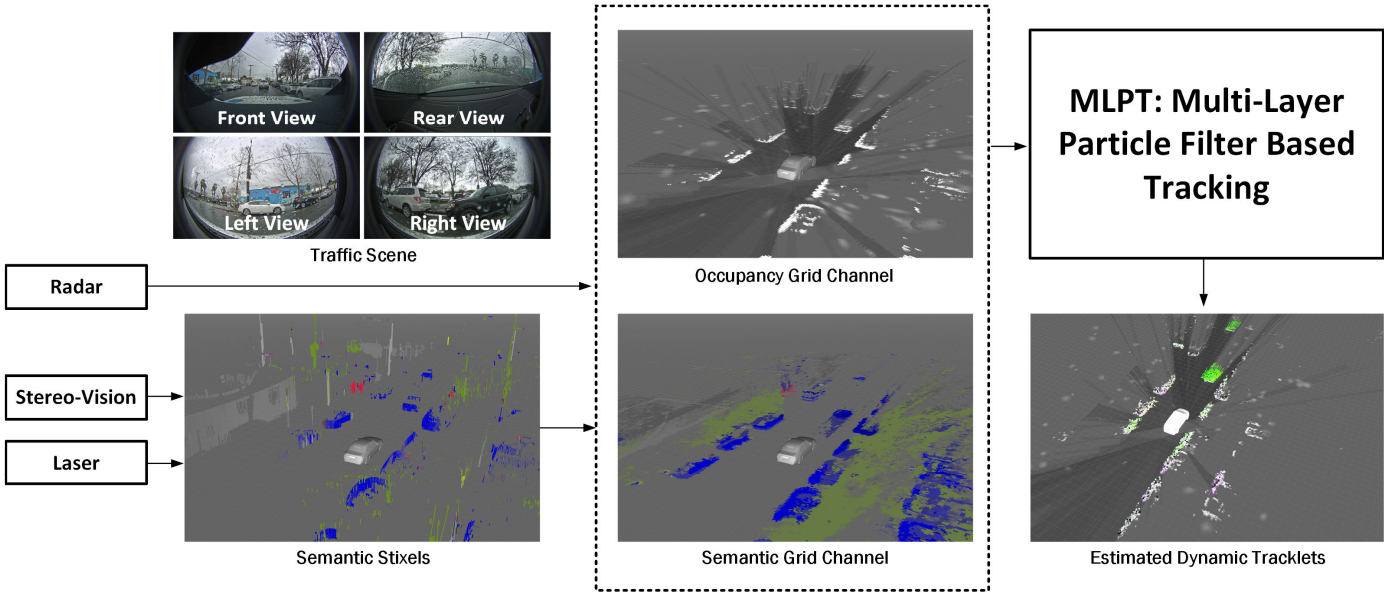


Fig. 2. System Overview.

information, it is limited only to the close neighborhood around a particle and its complexity increases for larger blocks.

Table I summarizes the characteristics of existing particle filter based methods for estimating dynamic grids, including the proposed solution.

In summary, existing grid-based particle filters are promising but come with two limitations. The first limitation is that most solutions use one population (one multi-modal probability density) of particles to estimate the state of multiple grid cells that are considered independent (one cell is described by its own probability density). A particle can migrate from one cell to another cell, becoming decoupled from its previous source cell density and reassigned to a new destination cell density. Because particles are regrouped into new independent cell estimators at each iteration, the identity of the tracked cell is lost. However, this information can be retained if additional heuristics are used. In other words, the particles can identify the existence of an object, in general, but not the existence of a "particular" object part or cell. In order to improve estimation and data association, the authors in [28], [31] adopt the idea of linking particles to objects by extending them with object IDs. It is important to note that the proposed methods still ignore the cell identities and must remap every particle to every cell.

The second limitation in grid-based particle filters is that the estimation of cells belonging to large and uniform grid areas are usually described by higher uncertainty and lower accuracy. For example, a particle that is predicted in the middle of a larger object can be attached to any of its occupied cells. This could lead to ambiguous data association and, in the end, higher uncertainties. In order to improve the estimation accuracy for uniform grid areas (i.e. in the middle of large objects with the same occupancy values), extra features from various sensors, larger patches, or more particles would be needed.

In the following we present our proposed MLPT solution, which is designed to overcome the above discussed limitations.

III. System Overview

This section presents an overview of the overall estimation system and highlights its primary subcomponents as shown in Fig. 2. The processing pipeline can be divided into three main steps. In the first step, raw sensor data is acquired by each sensor interface, preprocessed, and transformed into compact, medium-level data structures. In addition to measurement values, each sensor is described by its own measurement model. Lidar point clouds and stereo vision images are both transformed into a more compact Stixel-based representation. The Stixel is a convenient way to model the surrounding environment as vertically oriented rectangles that can incorporate properties like position, depth, size and semantic information of individual object parts [8], [9]. Unlike lidar and vision Stixels, the radar locations are compressed to a set of pre-filtered target points, which include the Cartesian position and velocity. More details about radar sensors and its data representation is presented in [33].

In the second step of the processing pipeline, compact sensor measurements are integrated into separate evidence grid channels. A channel can be defined as an independent 2D grid representation that accumulates a group of sensor observations, from one or multiple sensors. In this work, two measurement grid channels are computed – an *occupancy* grid channel and a *semantic* grid channel. However, it must be noted that the proposed method can be easily extended with additional grid channels using gradients or height properties.

The measurement occupancy grid channel is computed by accumulating the range measurements from all available sensors during a given fixed time interval. New measurements are then integrated into the occupancy grid by using the Dempster-Shafer theory of evidence. Thus, each cell is described by a belief mass of *occupied*, *free* or *unknown*, and can be converted into a conventional occupancy probability by using the pignistic transformation. More details about

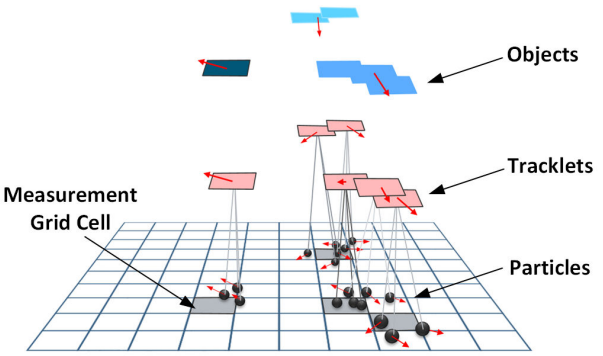


Fig. 3. Particles, tracklets and objects that share the same world. The particles are grouped into tracklets. Every tracklet state is updated with an individual particle filter, by employing its own particles. For persentation simplicity, the particle features are not included here, but are presented in the next image.Subsequent tasks like object clustering, can be performed by using the multiple estimated tracklets.

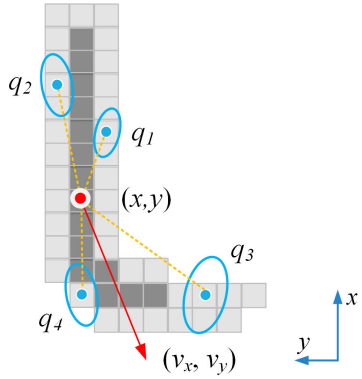


Fig. 4. Particle model. In the example the particle is created for an L-shape object (gray blob). The particle model is described by its position (*x.y*) and four reference object landmarks (blue). The landmarks are selected form the object contour (light gray). The ellipses represent the landmark covariances. Although the example presents only one particle assigned to agiven cell (red), similar particles are initialized in multiple locations, covering the entire object.

measurement occupancy grid calculation is presented in [4].

The measurement semantic grid channel combines the semantic information provided by both the Stixel world and the classified radar targets [33]. Every projected object label identifies a given object class (e.g. pedestrian, bicycle, vehicle etc.) and has an associated confidence score [9]. Due to memory constraints, after integrating labels into the semantic channel, we keep only the top K object labels per cell, based on their highest accumulated scores, instead of storing a full histogram of all the accumulated labels per cell.

It is important to note that both measurement grid channels are described by the same size and resolution. Also, both channels are aligned in time (the same time interval is used to integrate input measurements) and space (a given area in world coordinates is projected into the same cell indices in both grids).

The proposed hybrid particle filter-based estimator is the last step of the processing pipeline and is described in the following sections.

IV. Dynamic Grid Map Estimation Concepts

The primary objective of the tracking problem is to estimate the target's current state $\mathbf{s_t}$ from a set of noisy observations $\mathbf{z_{1:t}}$ received up to a given time *t*. The tracking problem can be formulated as a recursive Bayesian estimation process to estimate the posterior probability distribution $p(\mathbf{s_t}|\mathbf{z_{1:t}})$, by using the probabilistic motion model $p(\mathbf{s_t}|\mathbf{s_{t-1}})$ of the target and a defined measurement model $p(\mathbf{z_t}|\mathbf{s_t})$:

$$p(\mathbf{s_t}|\mathbf{z_{1:t}}) = \eta p(\mathbf{z_t}|\mathbf{s_t}) \int p(\mathbf{s_t}|\mathbf{s_{t-1}}) p(\mathbf{s_{t-1}}|\mathbf{z_{1:t-1}}) ds_{t-1} \quad (1)$$

where η denotes the normalization constant.

A possible implementation of the Bayes filter update rule described in (1) can be realized by using a particle filter-based estimator. In a particle filter algorithm, at each moment in time *t*, the posterior probability distribution $p(\mathbf{s_t}|\mathbf{z_{1:t}})$ is approximated by a set of *N* individual particles $\{\langle \mathbf{s_t^{[i]}}, w_t^{[i]} \rangle\}_{i=1..N}$, where each particle $\mathbf{s_t^{[i]}}$ represents a hypothesis of the tracked state $\mathbf{s_t}$ and has an assigned weight $w_t^{[i]}$ according to how well the particle matches the observations.

In general, standard particle filters can be used for solving various estimation problems described by non-linear motion models or non-linear measurement processes. Although particle filters are relatively simple to implement, they are not suitable for estimating larger states. This limitation comes from the fact that the number of particles required to approximate the object state tends to scale exponentially once more parameters are incorporated into the state $\mathbf{s_t}$.

As described in applications like object tracking [6], [16], [30] or simultaneous localization and mapping [32], a Rao-Blackwellised particle filter (RBPF) is applied as a common approach to estimate larger states [34]. The main idea of a RBPF algorithm is to draw samples only from a part of the state (e.g. object position and velocity), while the other state parameters are just carried with particles and are estimated analytically by using, for example, Binary Bayes filters, Dempster Shaffer-based updates, or Kalman Filters (one per particle). In this work we use the same idea of Rao-Blackwellisation to be able to handle "richer" particles. The tracking problem is therefore modeled as two decoupled recursive Bayesian estimations which, in the end, are combined into one single estimator. The two estimation parts are *self-localizing tracklets* and *multi-channel grid estimation*.

A. Self-localizing tracklets

The main idea of the following proposed concept is to improve the particle weighting by extending its state with additional knowledge about object shape. We assume that a given dynamic grid cell *c* is part of a larger object hypothesis. Therefore, apart from its position (x_c, y_c) and velocity $(v_{x,c}, v_{y,c})$, a dynamic grid cell *c* is also described by its relative position to *K* object landmarks $\mathbf{Q} = \{\mathbf{q_1}, \dots, \mathbf{q_K}\}$. These landmarks are initialized by selecting a random set of *K* points from the same object contour (see Fig. 4). It must be noted that, in order to take into account, the change in the object geometry, the relative distances from cells to their selected landmarks must also be recursively updated, at each measurement

iteration.

For every newly measured grid cell we create a fixed set of *N* particles. This group of particles will represent an individual particle filter-based estimator – a tracklet $\mathbf{h_t}$ (see Fig. 3). Therefore, instead of maintaining and updating one larger set of particles for the whole grid, we use multiple, smaller, independent populations of particles organized into tracklets, (see Fig. 3). A tracklet position and velocity at time *t* is denoted by $\mathbf{x_t} = [x_t, y_t, v_{x,t}, v_{y,t}]^T$. Additionally, the tracklet state is extended by a unique set $\mathbf{Q_t}$ of *K* random landmarks selected from the same object contour. The tracking problem can be formulated probabilistically as estimating the following joint posterior distribution:

$$p(\mathbf{x_t}, \mathbf{Q_t}|\mathbf{z_{1:t}}) \tag{2}$$

As in FastSLAM methods [32], tracklet state estimation can be implemented with a Rao-Blackwellised particle filter [34]. Thus, the joint posterior (2) is factored into independent estimators as:

$$\begin{aligned} p(\mathbf{x_t}, \mathbf{Q_t}|\mathbf{z_{1:t}}) &= p(\mathbf{x_t}|\mathbf{z_{1:t}})p(\mathbf{Q_t}|\mathbf{x_t}, \mathbf{z_{1:t}}) \\ &= p(\mathbf{x_t}|\mathbf{z_{1:t}}) \prod_{k=1}^{K} p\left(\mathbf{q_{t,k}}|\mathbf{x_t}, \mathbf{z_{1:t}}\right) \end{aligned} \tag{3}$$

The first distribution $p(\mathbf{x_t}|\mathbf{z_{1:t}})$ in (3) describes the tracklet position and velocity and is represented by a set of particles. The remaining factors $p\left(\mathbf{q_{t,k}}|\mathbf{x_t}, \mathbf{z_{1:t}}\right)$ are the landmark posterior distributions, which can be marginalized out analytically, i.e. every particle carries with it *K* Gaussian distributions (one per landmark) which are updated with Kalman Filters [35].

The main motivation of extending the tracklet state with a fixed set of randomly selected features is the need to improve the particle-to-measurement matching (see Fig. 4), through a set of "anchor" points. Intuitively, this leads to an improved stability in positioning tracklets with respect to a rigid object shape, therefore in fewer "drifting" tracklets (see Fig. 5). By making the analogy with the existing SLAM approaches, it also can be said that the presented tracklets are able to self-localize themselves with respect to the object boundary. By incorporating object feature information as part of particle state, a link between low-level particle world and high-level object world is created.

B. Multi-channel grid estimation

Multi-channel grid estimation involves integrating the measurement information structured into two grid channels: an occupancy grid and a semantic grid.

Similar to the previous section, for every newly measured grid cell, at time *t*, we define a tracklet as an independent particle filter-based estimator. Besides its dynamic properties $\mathbf{x_t}$, every tracklet is represented by an appearance vector $\mathbf{A}_t$, which combines the raw sensor data of the occupancy and semantic channels. The appearance vector $\mathbf{A}_t$ is thus completely described by a belief mass for occupied m_t^o, a belief mass for free m_t^f, and a semantic label l_t:

$$\mathbf{A_t} = [m_t^o, m_t^f, l_t]^T \tag{4}$$

In the context of Dempster-Shafer theory of evidence, both masses of occupied m_t^o and free m_t^f can be converted into a conventional occupancy probability $p(o_t)$ by using the pignistic transformation [37]. Thus, similarly to (3), the tracklet state estimation is described by a joint estimation problem of its motion $\mathbf{x_t}$ and appearance $\mathbf{A_t}$ and can be written in a factored form as:

$$\begin{aligned} p(\mathbf{x_t}, \mathbf{A_t}|\mathbf{z_{1:t}}) &= p(\mathbf{x_t}|\mathbf{z_{1:t}})p(\mathbf{A_t}|\mathbf{x_t}, \mathbf{z_{1:t}}) \\ &= p(\mathbf{x_t}|\mathbf{z_{1:t}})p(o_t|\mathbf{x_t}, \mathbf{z_{1:t}})p(l_t|\mathbf{x_t}, \mathbf{z_{1:t}}), \end{aligned} \tag{5}$$

where $p(\mathbf{x_t}|\mathbf{z_{1:t}})$ represents the posterior over tracklet position and velocity $\mathbf{x_t}$, $p(o_t|\mathbf{x_t}, \mathbf{z_{1:t}})$, and $p(l_t|\mathbf{x_t}, \mathbf{z_{1:t}})$ are tracklet occupancy and semantic label posteriors that are conditioned on its motion $\mathbf{x_t}$.

C. Combining the two problems into one estimator

The two decoupled problems described in the previous section can be combined into one estimator. Therefore, we can say that a dynamic tracklet $\mathbf{h_t}$ is fully described by its dynamic properties $\mathbf{x_t}$, appearance $\mathbf{A_t}$ and a set of object landmarks $\mathbf{Q_t}$, i.e., $\mathbf{h_t} = \{\mathbf{x_t}, \mathbf{A_t}, \mathbf{Q_t}\}$. Consequently, the two update equations (3) and (5) can be written as a single estimator as:

$$\begin{aligned} p(\mathbf{h_t}|\mathbf{z_{1:t}}) &= p(\mathbf{x_t}, \mathbf{A_t}, \mathbf{Q_t}|\mathbf{z_{1:t}}) \\ &= p(\mathbf{x_t}|\mathbf{z_{1:t}})\, p(o_t|\mathbf{x_t}, \mathbf{z_{1:t}}) \\ &\quad p(l_t|\mathbf{x_t}, \mathbf{z_{1:t}}) \prod_{k=1}^{K} p\left(\mathbf{q_{t,k}}|\mathbf{x_t}, \mathbf{z_{1:t}}\right) \end{aligned} \tag{6}$$

To approximate this posterior, we initialize for each occupied cell a tracklet $\mathbf{h_t}$ that is described by its position, speed, landmarks and appearance components. The full posterior density $p(\mathbf{h_t}|\mathbf{z_{1:t}})$ of the tracklet $\mathbf{h_t}$ at time *t* is represented by a set of *N* weighted particles:

$$\{\mathbf{x_t^{[i]}}, \mathbf{m_t^{[i]}}, l_t^{[i]}, (\mathbf{q_{t,1}^{[i]}}, \Sigma_{t,1}^{[i]}) \dots, (\mathbf{q_{t,K}^{[i]}}, \Sigma_{t,K}^{[i]}), w_t^{[i]}\}_{i=1}^{N} \tag{7}$$

where $\mathbf{x_t^{[i]}}$ represents the position and velocity of a particle, $\mathbf{m_t^{[i]}} = [m_t^{o,[i]}, m_t^{f,[i]}]^{\mathrm{T}}$ is the particle mass vector including both masses for occupied and free, $l_t^{[i]}$ is the particle semantic label, $w_t^{[i]}$ denotes the particle weight, $\mathbf{q_{t,k}^{[i]}}$ and $\Sigma_{t,k}^{[i]}$ define the mean and 2x2 covariance of the *k*-th landmark (see Fig. 4), assigned to the *i*-th sample, for $k = 1, \dots, K$, and $i = 1, \dots, N$.

For the purpose of implementation, we adopt the concept of Rao-Blackwellisation, i.e. marginalizing out a part of the state. That is, each tracklet is specified by a set of particles that are sampled from the position and velocity. The appearance and landmark properties are associated with each sample, thus contributing to a more precise weighting. Fig. 5 presents an intuitive example about how particle landmarks are used as

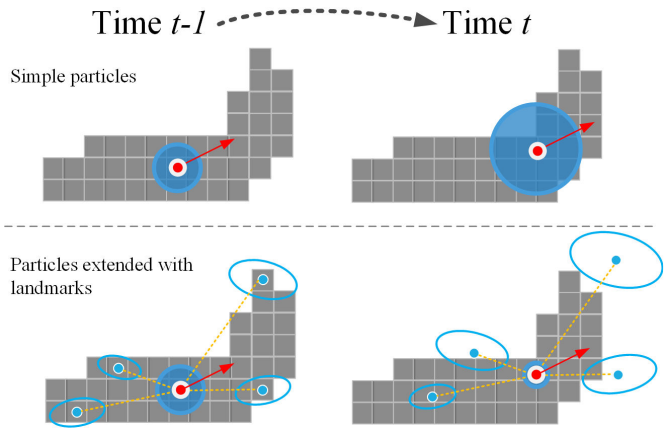


Fig. 5. An intuitive example of predicting and weighting one particle (red). Top: using a particle represented as a point described by a position and velocity. Bottom: using a particle extended with 4 landmarks (blue ellipses). In both cases the particle is predicted into the same destination cell. However it can be seen that, in the top scenario a particle is receiving a higher weight, because it is confirmed by an occupied cell. In the bottom case, even if the particle is confirmed by the same measurement, it receives a lower weight, due to its landmark-to-object misalignment

anchor points to improve the particle weighting. Subsequently they are updated in closed form at particle level.

V. Multi-Layer Particle Filter-Based Tracking

This section describes in more details how the Multi-Layer Particle Filter Tracking (MLPT) is implemented by taking into consideration the probabilistic model that was described previously. At each point in time t, the dynamic world is represented by a fixed list of N_h weighted tracklets $\mathbf{h}_t^{[i]}$:

$$\mathbf{H_t} = \{\mathbf{h}_t^{[i]}, w_h^{[i]}\}_{i=1}^{N_h} \tag{8}$$

Every tracklet $\mathbf{h}_t^{[i]}$ maintains an individual population of particles (7) and is regarded as a decoupled particle filter instance with its own prediction and update steps. Although a tracklet is estimated independently, it is also considered as a part of a higher level estimation mechanism, at the grid level. In other words, a tracklet $\mathbf{h}_t^{[i]}$ is seen as a meta-particle having its weight $w_h^{[i]}$ (based on the sum of particle un-normalized weights) and being created and destroyed by the same particle-filter specific resampling mechanism. In the following we will first present in detail how a particle filter is applied locally at the tracklet level and then, how the population of tracklets, treated as meta-particles, are managed at the grid level.

A. Particle Filter Based Tracklet Estimation

Once new measurements are received, the following steps recursively estimate the dynamic state for each tracklet as well as each grid cell:

1) Initialization: In the initialization, a new tracklet is created for every new measurement cell. Every tracklet is described by a set of N particles with random positions around the measurement cell and random velocities sampled from an initial distribution. At this step, all particles are initialized with the same occupancy masses and semantic values that are received from the corresponding input channels. Additionally, the new tracklet state is extended with a unique set $\mathbf{Q_t}$ of K random landmarks that are selected from an object contour. For assigning new landmarks, the following pre-processing steps are performed. First, every new cell must know which object it belongs to. We perform a pre-clustering step, similar to the classical connected component algorithm [39], in order to identify connected components in the measurement grid space. Every connected component approximates an object candidate (a blob) identified with a unique ID. Subsequently, every measurement grid cell stores the ID of its corresponding connected component. Second, in order to enable the selection of landmarks from a blob boundary, its contour is extracted by collecting all the blob cells that have at least one non-occupied neighbor cell (free cell or unknown). This operation requires one pass over the measurement occupancies and proves to be sufficiently fast, especially when the input data is organized in a compact form (excluding empty cells and including only the list of measurements pointing to the corresponding 2-D positions in the grid). Finally, after determining the correspondences between one possible object, its contour and its new tracklets are initialized with a fixed set of landmarks. For every tracklet we first identify a random index i on the object contour. Starting from that index, the initialization step selects K landmarks that are uniformly distributed along the contour. Specifically, a landmark is initialized by every $i + (k-1) \cdot M/K$ contour point, where M is the contour length, K is the number of landmarks, i is a random index of the starting contour point used to select the first landmark, with $1 \leq i \leq M$ and k is the index used to refer to a selected landmark in $\mathbf{Q_t}$, for $k = 1, \dots, K$.

In the end, different tracklets that are part of the same grid blob (object hypothesis) will be described by different combinations of random landmarks that are selected from the same blob contour. However, all the particles included into one tracklet will be initialized with the same fixed constellation of landmarks, pre-selected to describe the tracklet state.

It has to be noted that this procedure does not guarantee that landmarks are always selected from the same real object. However, considering that multiple tracklets are spawned for the same object, these tracklets (including their particles) will "compete" against each other and, as a consequence, weaker tracklets and particles will be discarded in a natural way by the resampling steps (presented further, in the Tracklet Management sub-section). Additionally, new tracklet hypotheses, along with their newly selected anchor points, are always being created and initialized from the latest measurements, ensuring in this way different hypotheses adapted to the newest measurements.

2) Prediction: In this step, particles are predicted at a new position in the grid, by considering the elapsed time, and their estimated state at the previous particle filter cycle. A constant velocity motion model is assumed, where the modeling error is accounted for by perturbing each propagated sample with a random noise component.

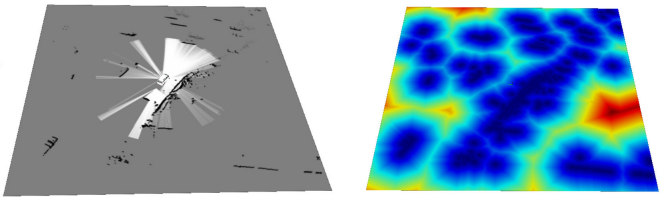


Fig. 6. Example of a distance map (right), precomputed for the occupancy grid (left).

Note that landmark prediction is done implicitly by the particle prediction, as the landmarks are conditioned on the particle state and follow the motion of the particle.

3) Weighting: Every predicted sample is being assigned a new importance weight. This step incorporates the information from the measurement into the particle distribution by giving weights to particles, which are proportional to the likelihood of matching the observation.

For a measurement $\boldsymbol{z_t}$ at time *t*, and the predicted state from above $\boldsymbol{s_t}$, the measurement model consists of three components: a measurement cell likelihood $p\left(\boldsymbol{z_t^d}\middle|\boldsymbol{s_t^{[i]}}\right)$, a landmark based likelihood $p\left(\boldsymbol{z_t^l}\middle|\boldsymbol{s_t^{[i]}}\right)$, and a semantic likelihood $p(\boldsymbol{z_t^s}|\boldsymbol{s_t^{[i]}})$. The measurement cell likelihood is based on a position error, which is the distance between the measurement and the closest particle, while the landmark likelihood depends on the distance between the particle landmarks and the measurement contours representing a measure for shape alignment. Furthermore, the semantic likelihood is defined by a dissimilarity metric given the particle's semantics.

In the following, we will refer to these terms as *weight factors*, which contribute to the overall particle importance weight.

If all three likelihood components are independent, the weight $w^{[i]}$ of the *i*-th particle $\mathbf{s_t^{[i]}}$ can be defined as:

$$w^{[i]} = p\left(\mathbf{z_t}\middle|\mathbf{s_t^{[i]}}\right) = p\left(\mathbf{z_t^d}, \mathbf{z_t^l}, \mathbf{z_t^s}\middle|\mathbf{s_t^{[i]}}\right)$$

$$= p\left(\mathbf{z_t^d}\middle|\mathbf{s_t^{[i]}}\right)p\left(\mathbf{z_t^l}\middle|\mathbf{s_t^{[i]}}\right)p\left(\mathbf{z_t^s}\middle|\mathbf{s_t^{[i]}}\right) \quad (9)$$

We use two grids to cache computed distances for speeding up particle-to-measurement and landmark-to-measurement-contour association. For computing distance grids, we apply an algorithm similar to [38]. One grid stores all distances to the closest occupied points (see Fig. 6) and the other stores all distances to the closest measurement contours. In addition, each grid cell stores the position to the closest observation. Based on these grids, we find d_p as the distance between the *i*-th sample and the closest occupied point. Assuming a Gaussian error model, with a given standard deviation σ_d, the particle-to-measurement weight component follows as:

$$p\left(\mathbf{z_t^d}\middle|\mathbf{s_t^{[i]}}\right) = \frac{1}{\sqrt{2\pi}\sigma_d}\exp\{-\frac{d_p^2}{2\sigma_d^2}\} \quad (10)$$

In order to improve the stability in the presence of missed detections, every particle with a distance to the closest measurement larger than a given threshold is updated with a minimum default particle-to-measurement weight – a weight corresponding to the likelihood that a measurement was not observed.

We model an object contour point likelihood given the *k*-th particle landmark as:

$$w_l^{[k]} = \frac{1}{\sqrt{2\pi}\sigma_l}\exp\{-\frac{(d_l^{[k]})^2}{2\sigma_l^2}\}, \quad (11)$$

where $d_l^{[k]}$ represents the distance between the *k*-th landmark and its closest contour point defined by the distance map (see Fig. 6), and σ_l describes the standard deviation assigned to the landmark-to-contour distance noise. Assuming error independence between the total *K* landmarks, the joint factor $p\left(\boldsymbol{z_t^l}\middle|\boldsymbol{s_t^{[i]}}\right)$ assigned to the *i*-th particle can be computed according to:

$$p\left(\mathbf{z_t^l}\middle|\mathbf{s_t^{[i]}}\right) = \prod_{k=1}^{K} w^{[k]} \quad (12)$$

As in the case of particle-to-measurement weight component, every landmark with a distance to the closest contour larger than a given threshold will be assigned a default minimum landmark-to-contour weight $w^{[k]}$. This helps in handling missed detections, avoids sharp changes in the particle weight (zero weight in the case of wrong or no associations) and improves the overall algorithm stability.

For the semantic weight, we first calculate a semantic dissimilarity metric between the predicted label l_p (particle label) and the associated measurement label l_m received from the closest occupied object cell:

$$d_s = 1 - \eta_l \cdot h(l_p, l_m), \quad (13)$$

where $h(l_p, l_m)$ is a score function that is defined as:

$$h(l_p, l_m) = \begin{cases} c_1, & l_p = l_m \\ c_2, & either\ l_p\ or\ l_m\ is\ unknown \\ c_3, & l_p \neq l_m \end{cases} \quad (14)$$

and η_l is a normalization constant: $\eta_l = 1/(c_1 + c_2 + c_3)$ having c_1, c_2 and c_3. The three score values are selected such that $c_3 < c_2 < c_1$. Considering that the resulting dissimilarity distance d_s is distorted by a Gaussian noise with a given standard deviation σ_s, the semantic weight factor can be calculated according to:

$$p\left(\mathbf{z_t^s}\middle|\mathbf{s_t^{[i]}}\right) = \frac{1}{\sqrt{2\pi}\sigma_s}\exp\{-\frac{d_s^2}{2\sigma_s^2}\} \quad (15)$$

4) Appearance and Landmark Updates: In order to update the particle landmarks, that are defined previously according to the Rao-Blackwellisation process, we use 2x2 Kalman filters (one per landmark). The state that is estimated by each Kalman filter is a 2D position. Moreover, each particle's mass for

occupied $m_t^{o,[i]}$ and mass for free $m_t^{f,[i]}$ is updated with the associated measurement masses by using the Dempster-Shafer rule of combination, as also proposed in [4]. For simplicity, the semantic labels are not updated.

5) Estimation: A weighted average of the particle states is applied to estimate both the tracklet states and the grid cell states. The tracklet state is estimated based on its corresponding particles, even if its particles are projected into multiple cells. However, for computing the grid cell state, all particles projected into the same cell are used, regardless of to which tracklet they belong. In practice, for a better processing time we directly use tracklets to get the average cell state.

6) Resampling: We use a Stochastic Universal Resampling algorithm with linear complexity to resample the particles after normalizing particle weights for each tracklet. This algorithm selects a new set of particles from the previous set by taking their importance weights into account and thus replacing particles with lower weights. We also adopt a selective resampling strategy where the particle resampling is triggered only when the particle diversity is lower than a predefined threshold [36].

B. Tracklet Management

Theoretically, the number of tracklets can grow above the available memory limits (more tracklets could be created in order to describe the tracked objects). While tracklets should be replicated and removed according to their existence probability, the maximum number of tracklets should be fixed within a predefined memory bound. Consequently, measurements need to be prioritized. This creates a need for a sophisticated management mechanism. Intuitively, a tracking mechanism should do the following: initialize new tracklets in new measurement cells; keep and replicate old tracklets by resampling as soon as new observations are associated; remove tracklets when no measurements are associated for a longer time interval.

The main idea of the proposed tracklet management is that, at the end of every grid map particle filter iteration t the final tracklet list $\mathbf{H_t}$ is composed of a subset of persistent tracklets $\mathbf{H_{t,p}} = \{\mathbf{h}_{t,p}^{[i]}, w_{h,p}^{[i]}\}_{i=1}^{N_p}$ and another subset of newly initialized tracklets $H_{t,b} = \{\mathbf{h}_{t,b}^{[i]}, w_{h,b}^{[i]}\}_{i=1}^{N_b}$:

$$\mathbf{H_t} = \mathbf{H_{t,p}} \cup \mathbf{H_{t,b}} = \{\mathbf{h}_{t,p}^{[i]}, w_{h,p}^{[i]}\}_{i=1}^{N_p} \cup \{\mathbf{h}_{t,b}^{[i]}, w_{h,b}^{[i]}\}_{i=1}^{N_b} \quad (16)$$

The set of persistent tracklets $\mathbf{H_{t,p}}$ is obtained by resampling from all the surviving tracklets propagated from time $t-1$ to time t. The list of newborn tracklets $\mathbf{H_{t,b}}$ is obtained by sampling according to a set of initialization weights $w_{init}^{[c]}$ precomputed for each measurement cell c (see Fig. 7).

As long as the number of tracklets does not reach maximum capacity, newly initialized tracklets are appended to the existing list. However, in cases when the maximum memory capacity is reached, the management mechanism ensures that, through sampling, the new list of tracklets will have a balanced ratio between newborn and persistent tracklets. In these extreme cases, the maximum allowed number of new tracklets is a parameter of the system and is setup to be less than 20% of the total available space in the list.

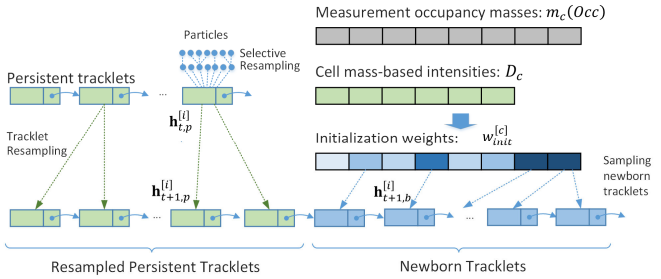


Fig. 7 Tracklet Management.

The advanced tracklet management solution can be summarized into several steps.

1) Compute Tracklet Occupancy Mass.

The tracklet occupancy mass $m_h(Occ)$ can be calculated as the weighted average of occupancy masses of its particles $m_t^{o,[i]}$:

$$m_h(Occ) = \sum_{i=1}^{N_h} \hat{w}^{[i]} \cdot m_t^{o,[i]} \quad (17)$$

where $\hat{w}^{[i]}$, is the normalized particle weight. The tracklet occupancy mass (17) is used in the next steps to determine how well a measured cell c is covered by its underlying tracklets.

2) Compute the cell mass-based intensity

The cell mass-based intensity D_c of a cell c is defined as a sum of all its tracklet occupancy masses $m_h^{[c]}(Occ)$.

$$D_c = \sum_{i=1}^{N_{h,c}} m_h^{[c]}(Occ) \quad (18)$$

where $N_{h,c}$ denotes the number of tracklets that fall into the cell c. Intuitively, the mass-based intensity can be interpreted as the expected number of targets in the cell c. It also provides a quantitative value about how well a cell is covered by tracklets.

3) Compute the initialization weights

The initialization weights $w_{init}^{[c]}$ describe how likely it is that we have to initialize a new tracklet $\mathbf{h}_t^{[i]}$ into a given cell c. As these weights have to be proportional to the need of initializing new tracklets, we determine them for each grid cell c as:

$$w_{init}^{[c]} = \begin{cases} m_c(Occ) - D_c, & m_c(Occ) > D_c \\ 0 & m_c(Occ) < D_c \end{cases} \quad (19)$$

where, $m_c(Occ)$ is the measurement occupancy mass of the cell c, and D_c is the mass-based intensity defined by (18).

4) Estimating the number of new and persistent tracklets

Suppose the system memory is limited so that the maximum number of tracklets that can be created is N_h^{max}. Additionally, the maximum number of newly accepted tracklets is N_b^{max}. Our goal is to calculate the new numbers for persistent $N_{t+1,p}$ and newborn tracklets $N_{t+1,b}$, for the next particle filter iteration at $t+1$, given the established memory limitations.

The number of potential newborn tracklets N_b^{pot} is determined by counting all the cells that meet the condition $m_c(Occ) > D_c$ in (19). Without memory limitation, new tracklets would be initialized in all N_b^{pot} grid cells. However, having the above defined bounds, the number of newborn tracklets $N_{t+1,b}$ to be used in the next particle filter cycle at time $t+1$ is estimated as:

$$N_{t+1,b} = \min(N_b^{pot}, \max(N_b^{max}, N_{available})) \quad (20)$$

In the equation above $N_{available}$ is the number of empty slots available for adding new tracklets, defined as:

$$N_{available} = N_h^{max} - N_{t,h} \quad (21)$$

where $N_{t,h}$ is the total number of tracklets used in the current particle filter iteration. Finally, the updated number $N_{t+1,p}$ of persistent tracklets is calculated as:

$$N_{t+1,p} = \min(N_{t,h}, N_h^{max} - N_{t+1,b}) \quad (22)$$

The total number of all the tracklets at time $t+1$, therefore, is defined as:

$$N_{t+1,h} = N_{t+1,p} + N_{t+1,b}, \text{ where } 0 < N_{t+1,h} < N_h^{max} \quad (23)$$

5) Tracklet sampling

Once the number of persistent $N_{t+1,p}$ and newborn tracklets $N_{t+1,b}$ is estimated, the updated set of persistent tracklets $\mathbf{H_{t+1,p}}$ is created by selecting $N_{t+1,p}$ tracklets from $\{\mathbf{h}_{t,p}^{[i]}, w_{h,p}^{[i]}\}_{i=1}^{N_p}$ according to their weights $w_{h,p}^{[i]}$. However, for initializing new tracklets, the method consists of sampling a set of grid cell indices with a sampling probability proportional to their assigned initialization weights $w_{init}^{[c]}$, where $w_{init}^{[c]}$ is previously calculated according to (19). Finally, the sampled grid cell indices serve as the location to initialize new tracklets $\mathbf{h}_{t+1,b}^{[i]}$.

As every tracklet represent an independent particle filter, it requires a particle-level resampling to avoid degeneracy. The particle-level resampling, explained in the previous section, is applied only after the tracklet-level management step. In addition, we use a selective resampling implementation to trigger the particle-level resampling only for a subset of tracklets, when their particle diversity is lower than a given threshold [36]. It is important to note that the resampling method is applied once per tracklet ID. If the same tracklet was replicated multiple times by the tracklet-level resampling, its particles will be resampled once and then copied multiple times in the replicated tracklets.

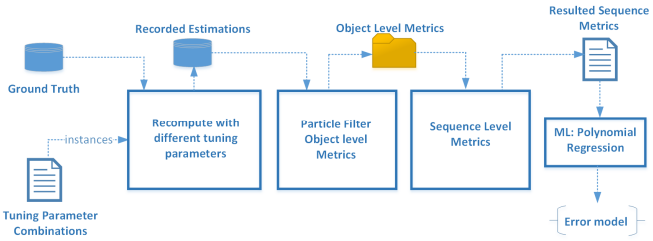


Fig. 8. Evaluation and parameter tuning pipeline.

TABLE II
COMPARISON BETWEEN DS-PHD [4] AND MLPT (PROPOSED)

	DS-PHD	MLPT (proposed)
Number of estimators	1 particle filter with *N* particles (*N*=8 Millions)	*K* filters with *M* particles each, where M ≪ N (e.g. *K*=128000, M=50)
Algorithm Structure	1 layer of particles	2 layers: particles and tracklets
Grid cell estimation	Average of all the particles in the cell	Average of all the tracklets in the cell
Estimated state	Position, velocity and occupancy	Position, velocity, landmarks, occupancy, semantics
Particle-to-cell regrouping	Yes	Can be ignored. Using tracklet-to-cell mapping
Resampling	On particle level	First – resampling on tracklet level, then resampling particles for selected tracklets only

V. EXPERIMENTAL RESULTS

In order to evaluate our approach, we use multiple challenging sequences from real scenarios with provided ground truth data. The ground truth is manually annotated and contains different objects (e.g. dynamic and static pedestrians, vehicles) with various orientation, speed and position.

For comparative results we analyze the tracking errors (see Fig. 8) obtained with the proposed MLPT approach and with a similar grid-based tracking solution – the Dempster-Shafer Probability Hypothesis Density tracking for Dynamic Occupancy Grid Maps (we will refer to it as DS-PHD) [4]. There are two main motivations for choosing DS-PHD as a reference algorithm for the comparison. First, it represents a well-established state-of-the art technique that addresses the same research problem, i.e., particle filter-based dynamic grid map estimation. Second, it represents an appropriate candidate with a similar setup (same sensors, same measurement grid resolution, same output interface etc.). This enables us to objectively assess both the proposed MLPT implementation and the reference DS-PHD in the same fair conditions and on the same ground-truth data set.

As discussed in previous sections, there are several conceptual differences between the existing point-based particle filter estimators and the proposed MLPT approach. The main differences between DS-PHD and MLPT are centralized in table II. These conceptual differences are consistent in comparison with other solutions proposed in the literature [23],

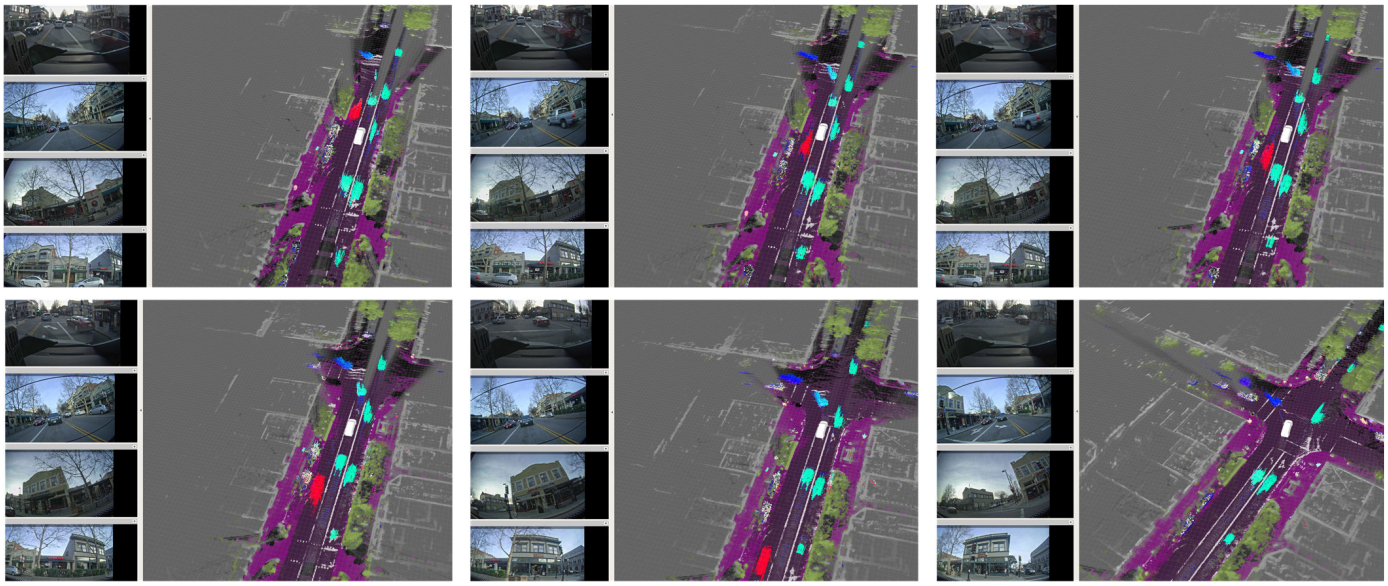


Fig. 9. MLPT Results for a traffic scenario, including consecutive snapshots sampled over a short period of time. The ego-vehicle is moving straight, and then is making a left-turning maneuver. Every snapshot includes images showing the vehicle's surroundings in the following order: front, rear, left and right. Additionally, every snapshot includes the perceived world and the estimated tracklet. The occupancy and semantic grids are overlayed into one plane. Every tracklet is placed on top of the grid plane and colored based on its moving direction. The color value indicates the tracklet velocity vector orientation, while the saturation value is proportional to the velocity magnitude (white for static objects).

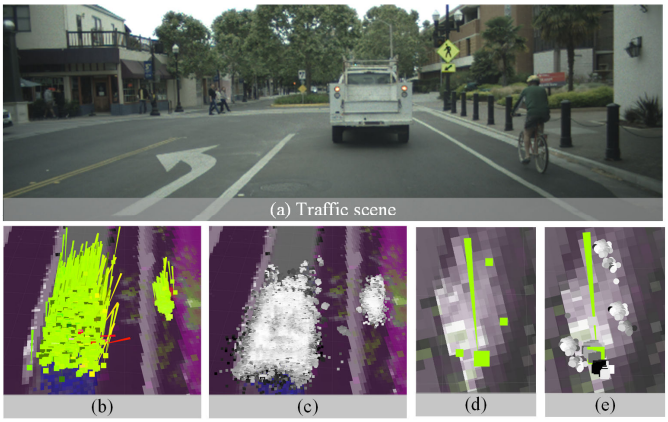
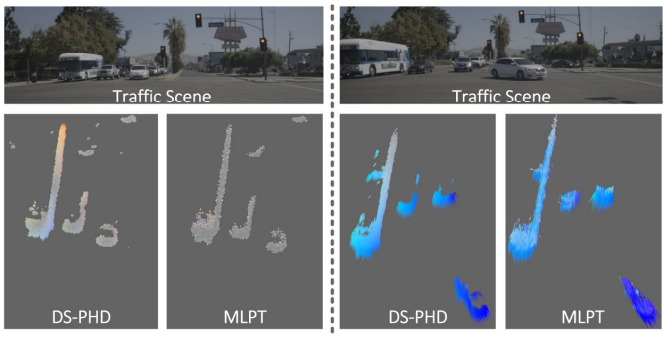


Fig. 10. A traffic scene showing a white vehicle stopping at the crosswalk and a bicyclist moving in the same direction along with the host vehicle. (a) Camera image. (b) Estimated dynamic tracklets for both front car and bicyclist. (c) The weighted particles provided by all the tracklets. Larger weights are illustrated by higher grayscale intensity. (d) One tracklet and its features, selected from the tracked bicyclist. For more clarity, the other tracklets are deactivated. (e) The same tracklet is illustrated by its weighted particles (squares) and features (spheres). It can be seen that the particles' features tend to be clustered around the estimated tracklet features in (d).

[24], [26], [27], [29], [31].

Fig. 9 shows an example with the results of the proposed MLPT approach for a traffic scenario. The shown example includes consecutive snapshots sampled over a short period of time. The estimated dynamic world is represented as dynamic tracklets. The ego-vehicle is moving straight, and then is making a left-turning maneuver. Every snapshot presents the perceived world and the estimated tracklets. Every tracklet is represented by a speed vector and colored based on its moving direction.

Fig. 10 presents a scene that includes a white vehicle stopping at the crosswalk and a bicyclist moving in the same direction along with the host vehicle. The subsequent images



(a) Traffic light intersection, stationary vehicles.

(b) Traffic light intersection, accelerating vehicles.

Fig. 11. A comparison between DS-PHD and MLPT (proposed) for a traffic light intersection use-case. Top: camera images from the traffic scene. Bottom: a region of interest extracted from the estimated dynamic grid (top view). (a). All vehicles are being stopped at the red light. (b) All the vehicles are accelerating (approximately 9.5 seconds later). The estimations are represented as a velocity vector field. Each velocity vector is attached to a dynamic grid cell. A vector color encodes the moving object orientation, while its saturation and length is direct proportional to the velocity magnitude (white is used for stationary vehicles).

show comparative examples with the estimated tracklets (Fig. 10.b), all weighted particles sharing the same world (Fig. 10.c), an example with one selected tracklet and its features (Fig. 10.d) and an example with the same tracklet represented by its weighted particles and features (Fig. 10.e).

Fig. 11 shows a traffic light scenario, with stationary objects stopping at the red traffic light (see Fig. 11.a) and then accelerating, which is recorded approximately 9.5 seconds later (see Fig. 11.b). The example compares the results estimated by DS-PHD (see Fig. 11.a left and Fig. 11.b left) and the proposed MLPT (see Fig. 11.a right and Fig 11.b right). In the stationary case, DS-PHD shows residual velocities, especially in the larger object (bus), due to a higher ambiguity generated by particles migrating from one cell to another inside a larger and uniform object area (see Fig. 11.a left). MLPT reduces the residual

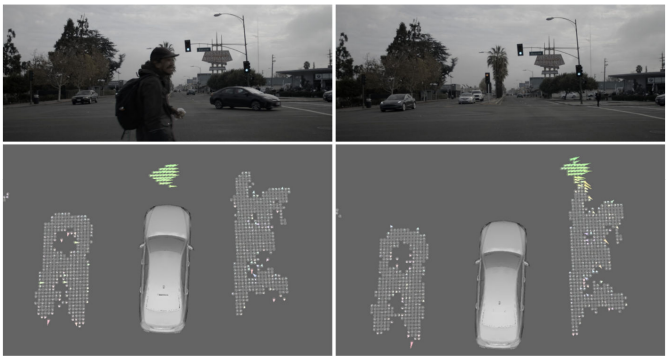


Fig. 12. A traffic scene showing a pedestrian crossing in front of the ego vehicle from left to right. Top: image snapshots from the corresponding traffic scene. Bottom: A part of estimated dynamic grid in the proximity of the ego-vehicle (top-view). The dynamic grid result shows a crossing pedestrian (green arrows). Additionally, the estimated grid includes the projection of two adjacent cars represented with white dots (not visible in the top camera images). The ego vehicle, as well as the other two left and right cars are stopped at the traffic light. It can be seen that as soon as the crossing pedestrian passes closer to the static obstacle (in front of the right vehicle), some of the pedestrian velocity vectors are temporarily biased by the wrong associated measurements belonging to stationary vehicle.

velocities by using landmarks to stabilize the particle-to-measurement matching (see Fig. 11.a right). As illustrated in the Fig. 11.b right, particle "anchor" points also improve the estimation of dynamic cells, especially in the middle of larger objects, as opposed to simple particles (without landmarks) that tend to provide reliable velocity vectors only in front of the accelerating bus (see Fig.11b left). Subsequently this leads to an improved algorithm convergence, as illustrated in Fig. 15, right (the comparison between the velocity graphs estimated for an accelerating object).

However, one has to note that estimated grid velocities can still be temporarily biased, especially when the tracked objects are close to each other. For example, Fig. 12 shows a traffic scene situation with a pedestrian crossing in front of the ego vehicle from left to right. The dynamic grid result illustrates the crossing pedestrian (green arrows). Additionally, the estimated grid includes the projection of two adjacent cars represented by white dots (white arrows which appear as dots due to the zero velocity of the two stationary vehicles). As soon as the crossing pedestrian passes in front of the right vehicle, some of pedestrian's velocity vectors are biased towards the stationary car. This is explained by the fact that some of the dynamic particles, including their anchor points are incorrectly associated to the measurements corresponding to the stationary vehicle. Although these particles are described by a lower weight and, subsequently, are more likely to be removed by the resampling mechanism, they still might lead to temporarily inaccurate estimated tracklets.

A. Parameter Tuning and Error Model Estimation

To perform a quantitative evaluation of both approaches, we introduce a parameter tuning framework (see Fig. 8) that determines the optimal parameter values for the given tracking algorithms. The proposed parameter tuning framework executes the given algorithm multiple times on the same reference sequence with a different combination of tuning parameters. The reference sequence describes a real traffic scenario and includes manually annotated ground truth objects. Each ground truth object is represented by a box and includes object semantics (vehicle, pedestrian, etc.), position, velocity and size.

There are two goals we hope to achieve through evaluation. One goal is to see how the accuracy metrics are evolving based on different tuning parameters. Another goal is to find the best combination of the tuning parameters that provide the highest estimation accuracy. The tuning parameters used were: standard deviation for matching features; occupancy mass decay factor (persistence probability) when no measurements are associated to particles; velocity process noise (standard deviation of the velocity noise); process noise for initializing new particles (standard deviation used to sample the initialization velocity). The permutations of tuning parameters are predetermined by either uniformly sampling from a domain of values or manually selecting from desired values. At each iteration, the evaluation process runs on the same ground truth sequence but with different instances of tuning parameters. Every permutation of tuning parameters is then stored as a separate instance.

The framework computes respective metrics for each iteration such as the sequence mean absolute error (MAE), root mean squared error (RMSE) and the standard deviation (StdDev) for speed and distance estimation. Our goal is to find a model which describes this error in the continuous space as a linear function of tuning parameters with the subset of discrete error values. To find this error model, we use a linear regression, where tuning parameters (features) estimate velocity mean absolute error $\varepsilon_v \in \mathbb{R}$ (target values). The error model is approximated by a quadratic polynomial function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ defined as:

$$\varepsilon_v = f(p_0, p_1, \dots, p_n)$$

$$= w_0 + w_1 p_0^2 + \cdots + w_{n+1} p_n^2 + w_{n+2} p_0 + \cdots + w_{2n+2} p_n, \quad (24)$$

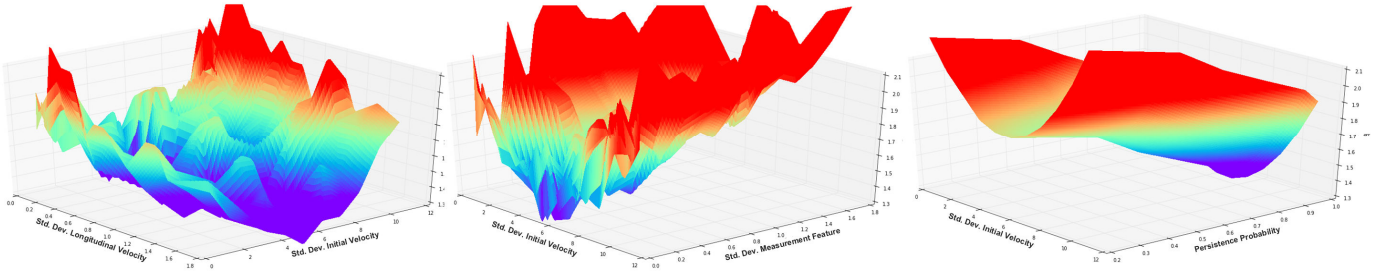


Fig. 13. Estimated Error Model. The examples show how the Velocity MAE is varying depending on different tuning parameters such as: StdDev of longitudinal velocity vs. StdDev of initial particle velocity (left), StdDev of initial particle velocity vs. StdDev of measurement featrues (middle) vs Persistence Probability.

TABLE III
QUANTITATIVE RESULTS DS-PHD [4] AND MLPT (PROPOSED)

	DS-PHD		MLPT (proposed)			
Number of particles	8M particles		10K tracklets, 50 particles/tracklet		10K tracklets, 128 particles/tracklet	
Metric	MAE	StdDev	MAE	StdDev	MAE	StdDev
With the tuning parameters used before optimal parameter tuning						
Velocity	0.82	0.94	0.67	0.53	0.56	0.35
Distance	0.38	0.02	0.46	0.02	0.47	0.016
With the best tuning parameters						
Velocity	0.65	0.51	0.546	0.42	0.474	0.788
Distance	0.38	0.02	0.24	0.02	0.46	0.02

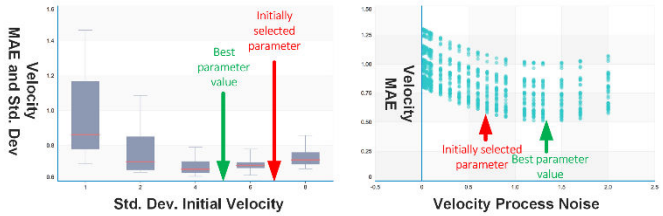


Fig. 14. DS-PHD [4] Tuning Parameters vs. Estimated Velocity Error. Left: the picture shows how the Mean Absolute Velocity (MAE) error is varying when selecting different distributions to initialize new particles. Right: the plot shows how the same velocity error is changing with respect to selected process noise (in this case velocity standard deviation) that is used in the particle prediction. In both figures we show the initially selected tuning parameters and the best tuning parameters, estimated after analyzing the evaluation results over different combinations of parameters.

where $w_0, \dots, w_{2n+2} \in \mathbb{R}$ are the coefficients of the polynomial determined by the linear regression, $p_0, \dots, p_n \in \mathbb{R}$ are the tuning parameters, and n is the number of tuning parameters.

From (24), the optimal tuning parameters correspond to those arguments $p_0, \dots, p_n$ for which the error ε_v function is minimum. The same parameter tuning procedure was applied for both DS-PHD and the proposed MLPT solution.

Fig. 13 shows the estimated error model in the case of the MLPT algorithm. It can be observed how the velocity error surface is changing depending on different parameter combinations and parameter values.

Fig. 14 illustrates similar example with the error model in case of DS-PHD. Specifically, the picture shows how the Mean Absolute Velocity (MAE) error is varying with respect to different distributions to initialize new particles (left), or with respect to selected process noise (in this case velocity standard deviation) that is used in the particle prediction. In both charts we show the initially selected tuning parameters and the best tuning parameters, estimated after analyzing the evaluation results over different combinations of parameters.

B. Evaluation with provided ground truth data

The main objective of this phase was to evaluate both particle-filter grid map tracking solutions by using the ground truth sequences.

Fig. 15 presents a comparison between the estimated velocity for MLPT and DS-PHD in a scenario with manually annotated targets. The evaluation was done only on specific selected objects, in order to see how the estimated velocity of a given target is evolving in time. The left image shows the bounding boxes provided by the ground truth and the estimation result of MLPT. The other two images present the result of the velocity estimation of both algorithms (MLPT: blue, DS-PHD: orange) compared to the ground truth velocity (green). The example graphs correspond to an almost stationary pedestrian (central image) and an accelerating car (right image). As previously described in the Fig. 11, it can be seen that MLPT converges faster and closer to the ground truth value. However, it also

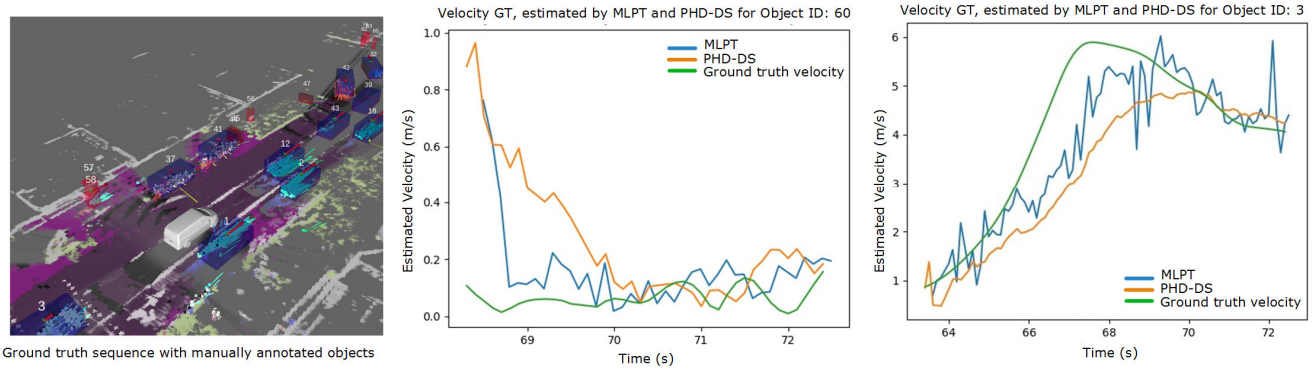


Fig. 15. Evaluating the MLPT and DS-PHD on individual objects from the ground truth sequence (left image). Each individual diagram represents the estated velocity for one single ground truth object (green line). The MLPT results is shown with blue. The DS-PHD estimation is illustrated with yellow line.

TABLE IV
COMPARISON BETWEEN DS-PHD [4] AND MLPT (PROPOSED) BY USING MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)

Velocity Range	1-3 m/s	3-7 m/s	> 7 m/s
DS-PHD	21.05 %	14.9 %	10.3 %
MLPT (proposed)	20.1 %	14.6 %	11.6 %

shows that MLPT is described by a higher estimation variance than DS-PHD.

Table III centralizes the mean absolute error (MAE) and standard deviation (StdDev) for the velocity and distance estimation using the two methods. The error metrics are calculated for the entire sequence by considering all the estimated objects. For MLPT we performed two different evaluations – with 50 particles per tracklet, and 128 particles per tracklet. Moreover, for both algorithms we include an evaluation based on initially tuning parameters, and another evaluation based on the best tuning parameters. Consequently, MLPT provides more accurate estimation with slightly higher variances in some cases.

Table IV compares both algorithms' accuracy by calculating the Mean Absolute Percentage Error (MAPE) for the estimated object velocity. The estimation accuracy was calculated for three different velocity ranges: 1 – 3 m/s, 3 – 7 m/s and above 7 m/s. It can be observed that the obtained percentage errors are higher for targets moving at lower velocities (1-3 m/s). This is due to the normalization of the estimated velocity by a lower ground truth velocity. In case of the velocity ranges 1 – 3 m/s and 3 – 7 m/s MLPT provides a higher accuracy than DS-PHD. At higher object velocities (> 7 m/s) DS-PHD achieves better results. The better performance of MLPT at low speeds (like traffic light intersection scenarios) where accelerating and decelerating objects are more frequent, can be explained by the faster convergence towards the ground truth value.

C. Processing time on a GPU implementation

Key steps in the inner-loop of the proposed algorithm were partially optimized in CUDA; other steps such as the use of distance transform to generate intermediate data structures that speed up associating features to measurement and the management of tracklets were left on the CPU. These steps are still un-optimized and are still an area of active research from the point of view of functionality in relation to the proposed algorithm and of implementation techniques for optimal performance. Fig. 16 shows the performance for a parametrization of 128K tracklets, 128 particles per tracklet and 3 features per particle. The size of the measurement grids is 1232x1232 cells, where each cell covers a 0.13x0.13 m region. The algorithm is executed on a Nvidia 1080Ti GPU platform. The graphs presented in Fig.16 are summed up durations in milliseconds of the operations on GPU for predicting particles using linear motion (4.2ms), associating features to measurements (7.8ms), weighting particles (3.9ms) and estimating the state of the tracklets (3.9ms). The total average duration of all steps is 19.9ms.

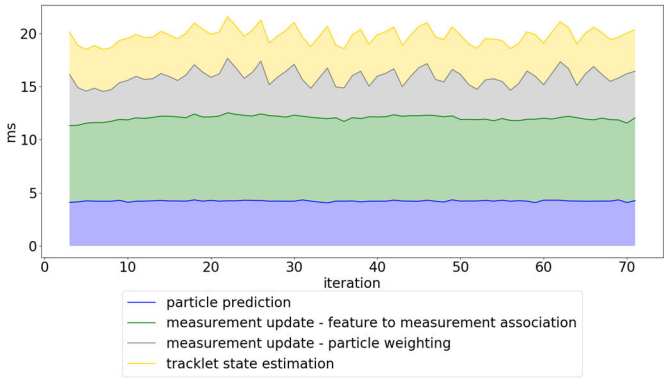


Fig. 16. Performance of the main steps of the MLPT algorithm.

VI. CONCLUSIONS

In this paper we presented a novel particle filter solution and introduced important concepts to track dynamic grid maps. The main motivation of the work was to address existing challenges in the field of grid map estimation and object tracking, in general. The problem of combining various sensor measurements was addressed by a multi-layer particle filter-based tracking approach (MLPT). Information like occupancy, object semantics, or shape was used to estimate the object state via Bayesian update mechanisms. Instead of using one particle filter for estimating the entire grid, we employ multiple individual particle filters, organized into tracklets that share the same world. By having tracklets in place, various particle filter-related steps like initialization or sampling were performed at the tracklet step. This offered the advantage of working in batches of particles instead of iterating over every single particle. Another important concept was "self-localizing" tracklets. Like simultaneous localization and mapping approaches, in our tracking solution every particle state is extended with a small set of landmarks that are randomly selected from the object contour. Intuitively, this idea was adopted in order to decrease the effect of "drifting" tracklets and, consequently, to obtain a more precise velocity estimation.

Although this work aimed to provide a more stable environment estimation by using "anchored" tracklets and particles via landmarks, further work can address better method variants to detect, select and manage these landmarks as well as to reduce the ghost velocities by using, for example, static map information, similarly to [27].

The experimental results with the ground-truth data illustrate that MLPT can estimate the dynamic world with a higher accuracy than previous methods. Although having richer particles contributes to more accurate results and a more comprehensive understanding of the world, this comes at a cost of having more complex structures and intermediate data representations. Incorporating additional features as part of the particle is a new topic that requires more focus in future research.

REFERENCES

[1] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 28, no. 5, pp. 694–711, May 2006, doi:10.1109/TPAMI.2006.104, [Online].

[2] S. Sivaraman, M. M. Trivedi, "Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis," *IEEE Trans. Intel. Transp. Sys.*, vol.14, no.4, pp.1773-1795, Dec. 2013, doi: 10.1109/TITS.2013.2266661, [Online].

[3] D. Nuss, T. Yuan, G. Krehl, M. Stuebler, S. Reuter and K. Dietmayer, "Fusion of laser and radar sensor data with a sequential Monte Carlo Bayesian occupancy filter," *2015 IEEE Intel. Veh. Symp. (IV),* Seoul, 2015, pp. 1074-1081, doi: 10.1109/IVS.2015.7225827, [Online].

[4] D. Nuss, et al., "A random finite set approach for dynamic occupancy grid maps with real-time application," *Intl. Jrnl. Robo. Res.*, vol. 37, no. 8, pp. 841-866, 2017.

[5] X. Zhang, W. Xu , C. Dong, and J.M. Dolan, "Efficient L-shape fitting for vehicle detection using laser scanners," *2017 IEEE Intell. Veh. Symp. (IV),* Los Angeles, pp. 54-59, 2017.

[6] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Auton. Robots*, vol. 26, no. 2-3, pp. 123-139, 2009, doi:10.1007/s10514-009-9115-1, [Online].

[7] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," *2012 IEEE Conf. Comp. Vis. and Pat. Recog.,* Providence, 2012, pp. 3354-3361, doi:10.1109/CVPR.2012.62448074, [Online].

[8] D. Pfeiffer and U. Franke, "Efficient Representation of Traffic Scenes by Means of Dynamic Stixels," *2010 IEEE Intel. Veh. Symp.*, San Diego, 2010, pp. 217-224, doi:10.1109/IVS.2010.5548114, [Online].

[9] L. Schneider, M. Cordts, T. Rehfeld, D. Pfeiffer, M. Enzweiler, U. Franke, M. Pollefeys, and S. Roth, "Semantic Stixels: Depth is not enough," *2016 IEEE Intell. Veh. Symp. (IV)*, Gothenburg, 2016, pp. 110-117, doi: 1109/IVS.2016.7535373, [Online].

[10] R. Varga, A. Costea, H. Florea, I. Giosan and S. Nedevschi, "Super-sensor for 360-degree environment perception: Point cloud segmentation using image features," *2017 IEEE 20th Intl. Conf. Intel. Transp. Sys. (ITSC)*, Yokohama, 2017, pp. 1-8, doi: 10.1109/ITSC.2017.8317846, [Online].

[11] M. H. Daraei, A. Vu and R. Manduchi, "Velocity and shape from tightly-coupled LiDAR and camera," *2017 IEEE Intel. Veh. Symp. (IV)*, Los Angeles, pp. 60-67, 2017, doi: 10.1109/IVS.2017.7995699, [Online].

[12] F. Kunz, D. Nuss, J. Wiest, H. Deusch, S. Reuter, F. Gritschneder, A. Scheel, M. Stubler, M. Bach, P. Hatzelmann, C. Wild, and K. Dietmayer, "Autonomous driving at Ulm University: A modular, robust, and sensor-independent fusion approach," *2015 IEEE Intel. Veh. Symp. (IV)*, Seoul, 2015, pp. 666-673, doi: 10.1109/IVS.2015.7225761, [Online].

[13] Y.-L. Chen, C.-T. Lin, C.-J. Fan, C.-M. Hsieh, and B.-F. Wu, "Vision-based nighttime vehicle detection and range estimation for driver assistance," *IEEE Intl. Conf. SMC*, Singapore, Oct. 2008, pp. 2988–2993, doi: 10.1109/ICSMC.2008.4811753, [Online].

[14] M. Isard and A. Blake "Condensation – conditional density propagation for visual tracking," *Intl. Jrnl. Comp. Vis.,* vol. 29, no. 5, pp. 5-28, Aug. 1998, doi: 10.1023/A:1008078328650, [Online].

[15] Jackson, J.D.; Yezzi, A.J.; Soatto, S., "Tracking deformable moving objects under severe occlusions," *43rd IEEE Conference Deci. Cont. (CDC)*, vol.3, no., pp. 2990-2995, 2004, doi: 10.1109/CDC.2004.1428922, [Online].

[16] A. Vatavu, R. Danescu, and S. Nedevschi, "Stereovision-Based Multiple Object Tracking in Traffic Scenarios using Free-Form Obstacle Delimiters and Particle Filters," *IEEE Trans. Intel. Transp. Sys.*, Vol. 16, No. 1, pp. 498-511, Feb. 2015, doi: 10.1109/TITS.2014.2366248, [Online].

[17] A. Vatavu, N. Rexin, S. Appel, T. Berling, S. Govindachar, G. Krehl, J. Peukert, M. Schier, O. Schwindt, J. Siegel, C. Zalidis, T. Rehfeld, D. Nuss, M.M. Maile, S, Zimmermann, K. Dietmayer, A. Gern, "Environment Estimation with Dynamic Grid Maps and Self-Localizing Tracklets," in *Proc. 2018 IEEE 21st Intl. Conf. Intel. Transp. Sys. (ITSC),* Maui, Hawaii, USA, Nov., 2018, pp. 3370-3377.

[18] S. Kraemer, M. E. Bouzouraa and C. Stiller, "Simultaneous tracking and shape estimation using a multi-layer laserscanner," in *Proc. 2017 IEEE 20th Intl. Conf. Intel. Transp. Sys. (ITSC)*, Yokohama, 2017, pp. 1-7, doi: 10.1109/ITSC.2017.8317667, [Online].

[19] A. Broggi, S. Cattani, M. Patander, M. Sabbatelli and P. Zani, "A full 3D voxel-based dynamic obstacle detection for urban scenario using stereo vision," in *Proc. 2013 IEEE Intl. Conf. Intel. Transp. Sys. (ITSC)*, The Hague, 2013, pp.71-76, doi: 10.1109/ITSC.2013.6728213, [Online].

[20] A. Elfes. "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp.46-57, June 1989, doi: 10.1109/2.30720, [Online].

[21] T.-D. Vu, O. Aycard, and N. Appenrodt, "Online localization and mapping with moving object tracking in dynamic outdoor environments," *2007 IEEE Intel. Veh. Symp. (IV)*, Istanbul, 2007, pp. 190–195, doi: 10.1109/IVS.2007.4290113, [Online].

[22] C. Coue, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessiere. "Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application," *Intl. Jrnl. Robo. Res.*, vol. 25, no. 1, pp. 19-30, Jan. 2006, doi: 10.1177/0278364906061158, [Online].

[23] R. Danescu, F. Oniga, S. Nedevschi, "Modeling and Tracking the Driving Environment with a Particle Based Occupancy Grid" *IEEE Trans Intel. Transp. Sys.*, vol. 12, no. 4, pp. 1331-1342, Dec. 2011, doi: 10.1109/TITS.2011.2158097, [Online].

[24] A. Negre, L. Rummelhard, and C. Laugier, "Hybrid Sampling Bayesian Occupancy Filter," *2014 IEEE Intel. Veh. Symp. (IV), Dearborn*, 2014, pp. 1307–1312, doi: 10.1109/IVS.2014.6856554, [Online].

[25] G. Tanzmeister and D. Wollherr, "Evidential grid-based tracking and mapping," *IEEE Trans. Intel. Transp. Sys.*, vol. 18, no. 6, pp. 1454-1467, June 2017, doi:10.1109/TITS.2016.2608919.

[26] S. Steyer, G. Tanzmeister and D. Wollherr, "Object tracking based on evidential dynamic occupancy grids in urban environments," *2017 IEEE Intel. Veh. Symp. (IV)*, Los Angeles, 2017, pp. 1064-1070, doi: 10.1109/IVS.2017.7995855, [Online].

[27] S. Steyer, G. Tanzmeister and D. Wollherr, "Grid-Based Environment Estimation Using Evidential Mapping and Particle Tracking," *IEEE Trans. Intel. Veh.*, vol. 3, no. 3, pp. 384-396, Sept. 2018. doi: 10.1109/TIV.2018.2843130, [Online].

[28] Sascha Steyer, Georg Tanzmeister, Christian Lenk, Vinzenz Dallabetta, Dirk Wollherr, "Data Association for Grid-Based Object Tracking Using Particle Labeling," in *Proc. 2018 IEEE 21st Intl. Conf. Intel. Transp. Sys. (ITSC)*, pp. 3036-3043, 2018, doi: 10.1109/ITSC.2018.8569511, [Online].

[29] R. Danescu, S. Nedevschi, "A Particle-Based Solution for Modeling and Tracking Dynamic Digital Elevation Maps," *IEEE Trans. Intel. Transp. Sys.*, vol. 15, No. 3, pp. 1002-1015, June 2014, doi: 10.1109/TITS.2013.2291447, [Online].

[30] A. Vatavu, R. Danescu, and S. Nedevschi, "Modeling and Tracking of Crowded Traffic Scenes by using Policy Trees, Occupancy Grid Blocks and Bayesian Filters," in *Proc. IEEE 17th Intl. Conf. Intel. Transp. Sys. (ITSC)*, Chingdao, Shandong, October 9-11, 2014, doi:10.1109/ITSC.2014.6957991, [Online].

[31] L. Rummelhard, A. N`egre, and C. Laugier, "Conditional Monte Carlo Dense Occupancy Tracker," in *Proc. 2015 IEEE 18th Intl. Conf. Intel. Transp. Sys. (ITSC)*, Las Palmas, Spain, 2015, pp. 2485–2490, doi: 10.1109/ITSC.2015.400, [Online].

[32] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *18th Nat. Conf. Art. Intel.*, pp. 593-598, 2002.

[33] C. Wöhler, O. Schumann, M. Hahn and J. Dickmann, "Comparison of random forest and long short-term memory network performances in classification tasks using radar," *2017 Sens. Data Fus.: Trends, Sols., Apps. (SDF)*, Bonn, 2017, pp. 1-6, doi: 10.1109/SDF.2017.8126350, [Online].

[34] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. "Rao-Blackwellised Particle Filters for Dynamic Bayesian Networks," in *Proc. 16th Conf. Uncert. Art. Intel. (UAI)*, pp. 176-183, 2000.

[35] R. Kalman, "A new approach to linear filtering and predict ion problems", in *Trans. ASME. Jrnl. Bas. Eng.,* vol. 82, pp. 35-45, 1960.

[36] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2006.

[37] P. Smets, "The combination of evidence in the transferable belief model," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 12, no. 5, pp. 447–458, 1990.

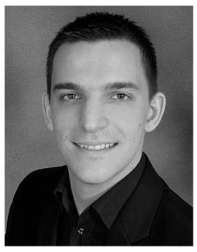
[38] G. Borgefors, "Distance transformations in arbitrary dimensions," *Comput. Vis. Graph. Image Proc. 27*, pp. 321–345, 1984.

[39] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Computer Vision and Image Understanding*, Vol. 89(1), pp. 1-23, 2003.

Andrei Vatavu (M'12) received the M.S. degree in computer science, from Technical University of Cluj-Napoca, Cluj-Napoca, Romania, in 2008, followed by the PhD degree (Computer Science) in 2014 from the same university. From 2007 to 2011 he was research assistant with the Computer Science Department, TUCN, and from 2011 to 2016 he was Lecturer with the same university, teaching Image Processing, Data Structures and Algorithms and Object Oriented Programing. He is currently working with Mercedes-Benz Research and Development North America, Inc., Autonomous Driving Department. His main research interests include sensor fusion, grid map based estimation and object tracking with the application in Autonomous Driving.

Melissa Rahm received the B.S. degree in mathematics from University of Applied Sciences, Stuttgart, Germany, in 2017. She is currently pursuing the M.S. degree in applied computer science with focus on autonomous systems at University of Applied Sciences, Esslingen, Germany and is writing her master's thesis under the supervision of Mercedes Benz R&D North America, Sunnyvale, CA, US.

Suresh Govindachar received the B. Tech. degree in mechanical engineering in 1983 from the Indian Institute of Technology, Powai, M.S. degree in theoretical and applied mechanics in 1985, M.S. degree in mathematics in 1989, and Ph.D. degree in algebra in 1992 from Cornell University. He has since been working on optimizing applications for deployment with real-time performance in embedded systems.



Gunther Krehl received the M.S degree in computer science and engineering from the Technical University of Ilmenau, Germany, in 2014. Since then, he has been working in the Autonomous Driving department of Mercedes-Benz Research & Development North America, Inc. where his main focus is on sensor fusion. Specifically algorithm design for multimodal belief fusion of radar, lidar, stereo and mono vision into an overall environment representation. Further interests expand to machine learning techniques, multi-layer grid representations, object tracking and general optimization.

Abhishek Mantha received a B.S. degree in Computer Science with an emphasis in artificial intelligence from the University of Southern California in 2018. He is currently working on evaluating sensor fusion systems at Mercedes-Benz Research and Development North America.

Sagar Ravi Bhavsar received the B.E. degree in Electronics and Communications, in 2013 from Gujarat Technological University, India followed by M.S. degree in Electrical Engineering from The University of Texas at Arlington, in 2015. He works as a Senior Software Engineer with Mercedes-Benz Research and Development North America's Autonomous Driving team. His research interests include Object detection and tracking, and Sensor Fusion.

Manuel Schier finished a dual study program in 2014. This included a B.E. degree in industrial electronics from the University of Applied Sciences Ulm, Germany and an apprenticeship as an automotive mechatronics technician in collaboration with EvoBus GmbH in Neu-Ulm, Germany. In 2017 he received the M.S. degree in electrical engineering from the University Ulm, Germany. Since 2017 he's working as a Software Engineer in the Sensor Fusion team of the Autonomous Driving department of Mercedes-Benz in Sunnyvale, California.

Janis Peukert received the M.S. degree in electrical engineering and information technology from Karlsruhe Institute of Technology in 2016. He is currently working on perception systems for autonomous vehicles at Mercedes-Benz focusing on consistent object tracking and particle filters.

Michael Maile received the Dipl. Phys. degree in biophysics from the University of Ulm. He was with the Optoelectronics Department, where he was involved in the fields of optoelectronic components, telematics, crash avoidance, environment perception, and sensor fusion for autonomous vehicles. He manages the Sensor Fusion and Localization Team, Autonomous Driving Department, Mercedes-Benz Research and Development North America, Inc.