object boxes, only the information of the most reliable object is stored, i.e., the highest $\alpha^{\chi,c}$.

Finally, as already introduced in the previous section, this work is mainly based on LiDAR data. Therefore, if V2X data is available, the corresponding modeled measurement OG has to be fused with the LiDAR-based measurement OG. This is performed leveraging on the confidence values:

$$P(O^c_{z_k}) = \frac{\alpha^{pO,c}(\boldsymbol{z}^\beta)P(O^c|\boldsymbol{z}^\beta) + \alpha^{pO,c}(\chi)P(O^c|\chi)}{\alpha^{pO,c}(\boldsymbol{z}^\beta) + \alpha^{pO,c}(\chi)} \tag{3.19}$$

It is considered that both sources are sufficiently reliable on their own. Thus, the confidence of the fused information is calculated as:

$$\alpha^{pO,c}_{z_k} = \min\left(1,\ \alpha^{pO,c}(\boldsymbol{z}^\beta) + \alpha^{pO,c}(\chi)\right) \tag{3.20}$$

3.6. Dynamic Occupancy Grid

In the previous Sections 3.4 and 3.5, the calculation of a single-frame OG with LiDAR data and V2X information has been introduced. This section introduces the recursive estimation procedure and the additional estimation of objects' dynamics, i.e., the development of a DOG.

The DOG implemented in this thesis is based on the approach presented in [90, 93, 96], where a real-time capable DOG is proposed using the DST and a Particle Filter (PF). Since the details of this method are crucial for the development and integration of other newly proposed methods, such as the COG (Section 4.5) or velocity feedback (Section 4.7), they are presented in the following sections. First, the DST and the PF are introduced in Sections 3.6.1 and 3.6.2 making special emphasis on their use in the DOG. Then the different steps concerning the recursive estimation of the DOG are provided in Section 3.6.3. An illustrative example of the main workflow of this DOG is provided in Figure 3.10.

3.6.1. Occupancy representation with Dempster-Shafer theory

The DST [28, 136], also known as evidence theory, is a mathematical framework for dealing with uncertainty and combining evidence obtained from different sources. It considers a finite set of hypotheses called the frame of discernment Ω. For each hypothesis $A \in 2^\Omega$, a mass function $m : 2^\Omega \rightarrow [0, 1]$ is defined. Mass functions are subject to:

$$m(\emptyset) = 0 \tag{3.21}$$

$$\sum_{A \in 2^\Omega} m(A) = 1 \tag{3.22}$$

where $m(A)$ denotes the degree of belief committed to A.

The frame of discernment that gathers the hypothesis of occupied O and free F is $\Omega = \{O, F\}$

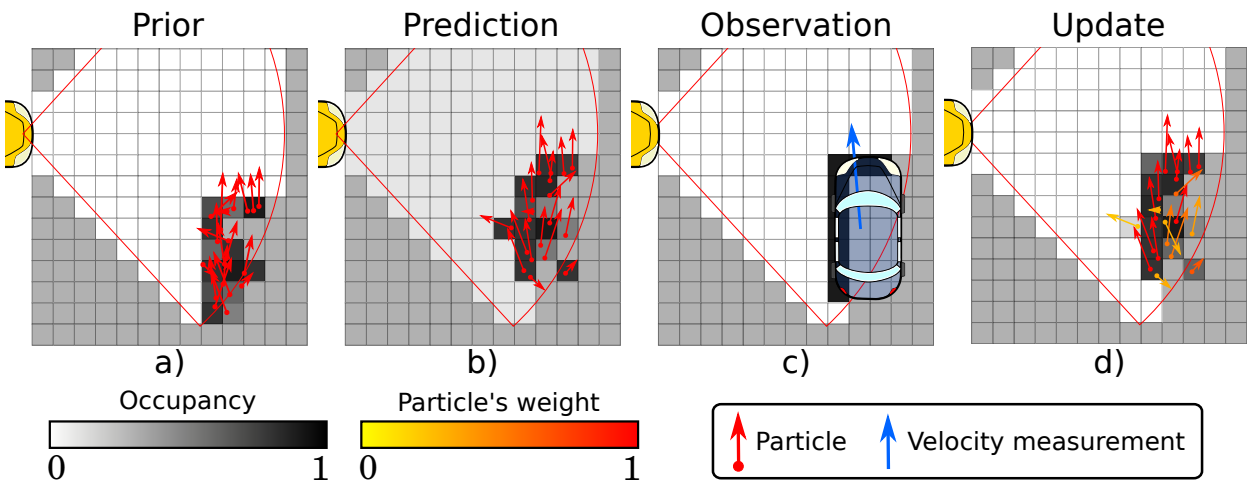


Figure 3.10: Illustrative example of the DOG main data flow. a) Prior grid map at time step k corresponding to the posterior grid map at time step $(k-1)$. b) Cells' occupancy state and particles' dynamic state are predicted taking into account process models and the elapsed time. c) Two types of measurements are expected for update: occupancy measurements and velocity measurements. d) Both types of measurements are used to update the occupancy state and particles' weights.

and is constituted by four subsets: $\{F\}$, $\{O\}$, $\{F,O\}$, and the empty set $\emptyset$. Complying with (3.21) and (3.22):

$$m(O) + m(F) + m(\Omega) = 1 \tag{3.23}$$

In contrast to a probability-based representation, the representation with evidence masses allows to distinguish between conflicting information and missing information. For example:

- **Conflicting information:** Consider two measurements that model opposing occupancy data for a cell's state: fully occupied and free. The resultant occupancy probability represents the cell's occupancy state as $P(O) = 0.5$ and the evidence masses as $m(O) = m(F) = 0.5$.
- **Missing information:** Consider a cell for which no data has been obtained. Since it is not known whether the cell is occupied or free, the occupancy probability is again set as $P(O) = 0.5$, while the evidence masses are set to $m(O) = m(F) = 0$ and $m(\Omega) = m(\{O,F\}) = 1$.

This distinction is used to obtain a more complete occupancy state representation, but also to reduce the computational load of the DOG. As is further explained in the next section, particles of the PF are linked to the occupied mass, thus free and unknown areas are not populated with them.

The measurement OG proposed in Sections 3.4 and 3.5 is represented in terms of occupancy probability, thus it has to be converted to the desired evidence masses. Therefore, for a cell c the measurement masses are calculated by:

$$m(O^c_{z_k}) = \min\left(\mathcal{T}^{max}_{mO},\ P(O^c_{z_k}) \cdot \alpha^{pO,c}_{z_k}\right) \tag{3.24}$$

$$m(F_{z_k}^c) = \min\left(\mathcal{T}_{mF}^{max},\ (1 - P(O_{z_k}^c)) \cdot \alpha_{z_k}^{pO,c}\right) \tag{3.25}$$

$$m(\Omega_{z_k}^c) = 1 - m(O_{z_k}^c) - m(F_{z_k}^c) \tag{3.26}$$

where $\mathcal{T}_{mO}^{max}$ and $\mathcal{T}_{mF}^{max}$ are design parameters defined to limit the maximum observed occupied and free masses. In this thesis, a conservative approach toward obstacles is selected, i.e., $\mathcal{T}_{mO}^{max} = 1$ and $\mathcal{T}_{mF}^{max} < 1$. Figure 3.11 shows the result of applying this transformation to an example of the inverse sensor model introduced in Section 3.4.2.

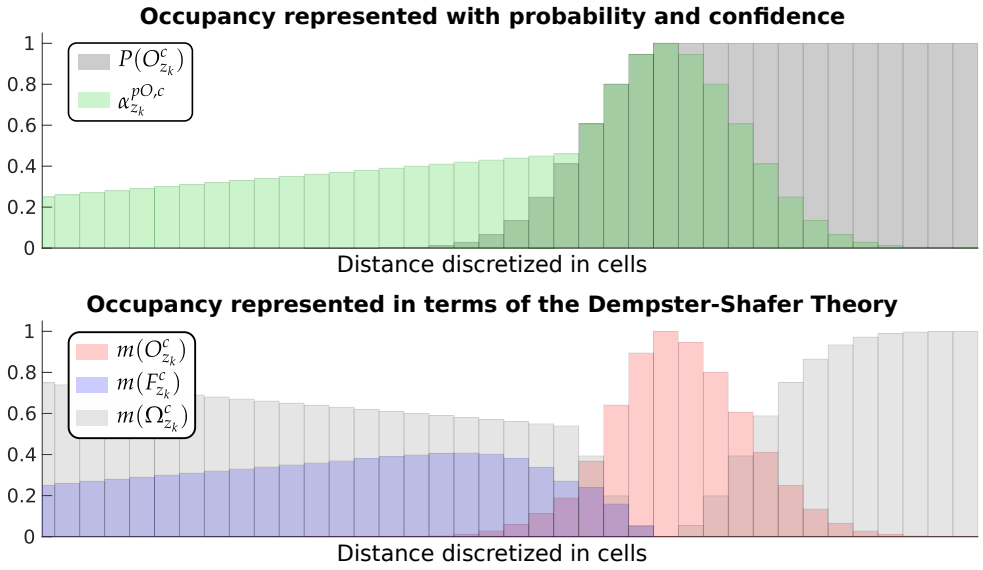


Figure 3.11: Example of the transformation from occupancy probability to evidence masses. a) Occupancy probability and confidence value obtained with the inverse sensor model introduced in Section 3.4.2. b) Obtained evidence masses applying equations (3.24), (3.25) and (3.26).

3.6.2. Particle Filter and relationship between particles and cells

The PF [150] is a non-parametric realization of the Bayes filter. Thus, it allows the modeling of non-Gaussian state probability distributions and non-linear systems. The PF aims to represent the posterior probability density function by a set of random state samples (particles) that are drawn from this posterior. Therefore, each particle is a hypothesis of what the true state may be.

In the implemented DOG approach, the grid is populated with a set of particles $\mathbf{P}$ which models the dynamics of the obstacles in the scene. The number of particles is constant and defined by a design parameter $\nu^{\mathcal{G}}$ for computational efficiency reasons. In each cycle, particles follow a workflow of (i) prediction, (ii) update, (iii) new particles generation, and (iv) resampling—the description of these steps is provided in Section 3.6.3.

A particle ρ is described by a dynamic state $\boldsymbol{s}^{\rho}$ and a weight ω^{ρ}. The state $\boldsymbol{s}^{\rho}$ is defined as:

$$\boldsymbol{s}^{\rho} = [x_{\mathrm{x}}, x_{\mathrm{y}}, v_{\mathrm{x}}, v_{\mathrm{y}}]^{T} \tag{3.27}$$

where x refers to the position, v to the velocity, subscripts x and y to the 2D components.

Particles can move along the cells of the grid map using a certain process model. Therefore, in each iteration, each cell gathers a different set of particles. The set of particles $\mathbf{P}^{c}$ within a grid cell c is calculated regarding the position of the particles $\boldsymbol{x}^{\rho}$ and the area covered by the cell $\boldsymbol{A}(c)$:

$$\mathbf{P}^{c} = \{\rho \in \mathbf{P} | \boldsymbol{x}^{\rho} \in \boldsymbol{A}(c)\} \tag{3.28}$$

being

$$\boldsymbol{A}(c) = \left[x_{\mathrm{x}}^{c} - 0.5l^{c},\ x_{\mathrm{x}}^{c} + 0.5l^{c}\right) \times \left[x_{\mathrm{y}}^{c} - 0.5l^{c},\ x_{\mathrm{y}}^{c} + 0.5l^{c}\right) \in \mathbb{R}^{2} \tag{3.29}$$

where x^{c} refers to the position of the center of the cell c and l^{c} denotes the cell's size.

The OG and PF algorithms are interconnected through the cells' occupied mass and dynamic state. The idea behind this is that the particles within an occupied cell are able to approximate the dynamic state of the object occupying it. For this purpose, the mass for occupied $m(O^{c})$ of a cell c is represented by the sum of the particle's weights gathered within it:

$$m(O^{c}) = \sum_{i=1}^{\nu^{c}} \omega^{i,\rho,c} \tag{3.30}$$

being $\omega^{i,\rho,c}$ the weight of the i-th particle within the cell c and ν^{c} the total number of particles within it. Given this relationship, particles with a dynamic state similar to the real object dynamic state will correctly model the occupancy transition along cycles and therefore have higher weights and survival chances.

3.6.3. Dynamic Occupancy Grid recursive realization

The employed DOG is a recursive process that estimates the posterior grid map considering the state of the grid map at the previous time step, the process and measurement models, and the observations at the current time step. In the following, the main steps to accomplish this are introduced.

3.6.3.1. Prediction

The prediction of the state is decoupled by the masses for occupied and free. The prediction of the mass for occupied is accomplished by predicting the particles' state, while the prediction of the mass for free is done as in static grid maps, with a decrease in its reliability.

Particles' state $\boldsymbol{s}^{\rho}$ is predicted from the previous to the current instant using a constant velocity

model. Their weight ω^{ρ} is also predicted using a persistence probability P_S:

$$\omega^{\rho}_{(k|k-1)} = P_S \cdot \omega^{\rho}_{(k-1|k-1)} \tag{3.31}$$

Based on the predicted particles, the predicted occupied mass $m^c(O^c_{(k|k-1)})$ of cell c is calculated using (3.30), i.e.:

$$m(O^c_{(k|k-1)}) = \sum_{i=1}^{\nu^c_{(k|k-1)}} \omega^{i,\rho,c}_{(k|k-1)} \tag{3.32}$$

being $\nu^c_{(k|k-1)}$ the total number of particles predicted into the cell c. Since multiple particles can be predicted into the same cell, the total sum of the predicted weights can possibly exceed the maximum allowed mass value, 1. In that case, the predicted occupied mass is limited to 1, and the particles' weights are normalized accordingly.

The predicted mass for free is simply predicted as a decrease in the belief depending on the elapsed time:

$$m(F^c_{(k|k-1)}) = \min\left(\alpha^F(\Delta t) \cdot m(F^c_{(k-1|k-1)}),\ 1 - m(O^c_{(k|k-1)})\right) \tag{3.33}$$

being $\alpha^F(\Delta t) \in [0, 1]$ a discount factor based on the time elapsed. Again, since the sum of masses cannot exceed 1, the predicted mass for free is limited accordingly to the predicted mass for occupied.

3.6.3.2. Occupancy state update

The update step is carried out in two phases. First, the cells' occupancy state is updated using predicted and measurement OGs. Then, particles' weights are updated with respect to velocity measurements, if available, and the resultant updated occupancy mass.

The occupancy update step is performed independently for each grid using the Dempster-Shafer combination rule [136] as follows:

$$m(O^c_{(k|k)}) = \frac{m(O^c_{(k|k-1)}) \cdot m(O^c_{z_k}) + m(\Omega^c_{(k|k-1)}) \cdot m(O^c_{z_k}) + m(O^c_{(k|k-1)}) \cdot m(\Omega^c_{z_k})}{1 - K} \tag{3.34}$$

$$m(F^c_{(k|k)}) = \frac{m(F^c_{(k|k-1)}) \cdot m(F^c_{z_k}) + m(\Omega^c_{(k|k-1)}) \cdot m(F^c_{z_k}) + m(F^c_{(k|k-1)}) \cdot m(\Omega^c_{z_k})}{1 - K} \tag{3.35}$$

with:

$$K = m(O^c_{(k|k-1)}) \cdot m(F^c_{z_k}) + m(F^c_{(k|k-1)}) \cdot m(O^c_{z_k}) \tag{3.36}$$

As explained in Section 3.6.3.5, in each cycle, new particles are generated. Therefore, a cell with occupied mass gathers a set of persistent particles—corresponding to the particles predicted into the cell—and a set of new generated particles. In order to fulfill with (3.30), the updated occupied

mass $m(O^c_{(k|k)})$ is divided into a mass for persistent objects ϱ^c_{per} and a mass for new-born objects ϱ^c_{new}, corresponding to the persistent and new particles, respectively. The division is calculated taking into account the predicted occupied mass $m(O^c_{(k|k-1)})$ and a birth probability P_B:

$$\varrho^c_{new} = \frac{m(O^c_{(k|k)}) \cdot P_B \cdot \left(1 - m(O^c_{(k|k-1)})\right)}{m(O^c_{(k|k-1)}) + P_B \cdot m(O^c_{(k|k-1)})} \tag{3.37}$$

$$\varrho^c_{per} = m(O^c_{(k|k)}) - \varrho^c_{new} \tag{3.38}$$

The mass for persistent objects will be used to normalize persistent particles weights with (3.43) and the mass for new-born objects to assign the weights of new particles with (3.53).

3.6.3.3. Particles update

Particles updated is only performed over the weight ω^ρ. Thus, the particles' states remain unchanged. Particles' weights are updated taking into account velocity measurements and the occupancy update introduced in Section 3.6.3.2. The velocity measurements are used to update particles' weights considering their dynamic state. The occupancy update is taken into account by normalizing the weights in order to sum up to the persistence mass.

The weight of a particle ρ in a cell c is updated with respect to a velocity measurement $\vartheta^c_{z_k}$ as follows:

$$\tilde{\omega}^{\rho,c}_{(k|k)} = \omega^{\rho,c}_{(k|k-1)} \cdot g^c_k(\vartheta^c_{z_k}|v^{\rho,c}_{(k|k)}) \tag{3.39}$$

where $\tilde{\omega}$ refers to the fact that the weight ω is not normalized yet and $g^c_k(\vartheta^c_{z_k}|v^{\rho,c}_{(k|k)})$ is a likelihood function of the velocity measurement $\vartheta^c_{z_k}$ and the velocity of the particle ρ in the cell c.

The likelihood function $g^c_k(\vartheta^c_{z_k}|v^{\rho,c}_{(k|k)})$ is modeled with a bivariate Gaussian distribution:

$$g^c_k(\vartheta^c_{z_k}|v^{\rho,c}_{(k|k)}) = \frac{1}{2\pi\sigma^{\vartheta,c}_{\mathrm{x}}\sigma^{\vartheta,c}_{\mathrm{y}}\sqrt{1-\lambda^2}} \cdot \exp\left(-\frac{1}{2\,(1-\lambda^2)}\left[C^2_{v_{\mathrm{x}}} + C^2_{v_{\mathrm{y}}} - 2\lambda\, C_{v_{\mathrm{x}}}\, C_{v_{\mathrm{y}}}\right]\right) \tag{3.40}$$

being:

$$\lambda = \frac{(\sigma^{\vartheta,c}_{\mathrm{xy}})^2}{\sigma^{\vartheta,c}_{\mathrm{x}}\sigma^{\vartheta,c}_{\mathrm{y}}} \tag{3.41}$$

$$C_{v_{\mathrm{x}}} = \frac{v^{\rho,c}_{\mathrm{x}} - \mu^{\vartheta,c}_{\mathrm{x}}}{\sigma^{\vartheta,c}_{\mathrm{x}}};\quad C_{v_{\mathrm{y}}} = \frac{v^{\rho,c}_{\mathrm{y}} - \mu^{\vartheta,c}_{\mathrm{y}}}{\sigma^{\vartheta,c}_{\mathrm{y}}} \tag{3.42}$$

where μ^ϑ and σ^ϑ correspond to the mean and the standard deviation of ϑ.

The updated normalized weight $\omega^{\rho,c}_{(k|k)}$ is calculated taking into account the confidence on the

velocity measurement $\alpha_{z_k}^{\vartheta,c}$ and the persistent mass ϱ_{per}^{c}:

$$\omega_{(k|k)}^{\rho,c} = \alpha_{z_k}^{\vartheta,c} \cdot \tilde{\omega}_{(k|k)}^{\rho,c} \cdot \eta + \left(1 - \alpha_{z_k}^{\vartheta,c}\right) \cdot \omega_{(k|k-1)}^{\rho,c} \cdot \bar{\eta} \tag{3.43}$$

The first term takes into account that the velocity measurement is reliable with a confidence $\alpha_{z_k}^{\vartheta,c}$. The second term models the fact that the velocity measurement is incorrect with $(1 - \alpha_{z_k}^{\vartheta,c})$, in which case, the velocity-based updated weight $\tilde{\omega}_{(k|k)}^{\rho,c}$ would be incorrect. η and $\bar{\eta}$ are normalization factors such that the final updated weight $\omega_{(k|k)}^{\rho,c}$ sums up to ϱ_{per}^{c}:

$$\eta = \left(\sum_{i=1}^{\nu_{(k|k-1)}^{c}} \tilde{\omega}_{(k|k)}^{i,\rho,c} \right)^{-1} \cdot \varrho_{per}^{c} \tag{3.44}$$

$$\bar{\eta} = \left(\sum_{i=1}^{\nu_{(k|k-1)}^{c}} \omega_{(k|k-1)}^{i,\rho,c} \right)^{-1} \cdot \varrho_{per}^{c} \tag{3.45}$$

being $\nu_{(k|k-1)}^{c}$ the number of particles predicted into the cell c (the number of persistent particles).

Notice that in cases where no velocity measurement is associated with the cell c $\alpha_{z_k}^{\vartheta,c}$ equals zero. Consequently, particle weights are only normalized with respect to the persistent mass.

3.6.3.4. Cells' dynamic state estimation

The dynamic state of a grid cell is calculated with respect to the updated persistent particles gathered within it. Moreover, given that new particles with random dynamic states are created in each cycle, only the particles that have been resampled at least n_R^{ρ} times are used for this calculation.

The mean velocity $\mu_{v_x}^{c}$ of the cell c in the x-component is calculated as:

$$\mu_{v_x}^{c} = \left(\sum_{i=1}^{\nu_R^{c}} \omega_{(k|k)}^{i,\rho,c} \right)^{-1} \cdot \sum_{i=1}^{\nu_R^{c}} \left(\omega_{(k|k)}^{i,\rho,c} \cdot v_{x,(k|k)}^{i,\rho,c} \right) \tag{3.46}$$

where ν_R^{c} denotes the number of particles that have been resampled at least n_R times in the cell c. The calculation for the y-component is analogous.

Similarly, the estimation of the standard deviation $\sigma_{v_x}^{c}$ and covariance $\sigma_{v_{xy}}^{c}$ are given by:

$$\left(\sigma_{v_x}^{c}\right)^2 = \left(\sum_{i=1}^{\nu_R^{c}} \omega_{(k|k)}^{i,\rho,c} \right)^{-1} \cdot \sum_{i=1}^{\nu_R^{c}} \left(\omega_{(k|k)}^{i,\rho,c} \cdot (v_{x,(k|k)}^{i,\rho,c} - \mu_{v_x}^{c})^2 \right) \tag{3.47}$$

$$\sigma_{v_{xy}}^{c} = \left(\sum_{i=1}^{\nu_R^{c}} \omega_{(k|k)}^{i,\rho,c} \right)^{-1} \cdot \sum_{i=1}^{\nu_R^{c}} \left(\omega_{(k|k)}^{i,\rho,c} \cdot (v_{x,(k|k)}^{i,\rho,c} - \mu_{v_x}^{c}) \cdot (v_{y,(k|k)}^{i,\rho,c} - \mu_{v_y}^{c}) \right) \tag{3.48}$$

In addition to the dynamic state, a dynamic-static classification of the cells is performed. This classification is accomplished using the Mahalanobis distance between the cell's dynamic state and the static velocity vector ($[0,0]^T$). Therefore:

$$d^{Mah} = \sqrt{(\boldsymbol{\mu}_v^c)^T (\boldsymbol{\Sigma}_v^c)^{-1} (\boldsymbol{\mu}_v^c)} \tag{3.49}$$

If d^{Mah} is lower than a threshold $\mathcal{T}_{v,Mah}$, the cell is classified as static; otherwise, it is classified as dynamic.

3.6.3.5. New particles and resampling

In each cycle, a set of new-born particles $\mathbf{P}_{new}$ is initialized along the occupied grid cells. For computational efficiency reasons, the total number of new particles initialized is fixed to ν_{new} and they are only initialized in cells in which an object has been sensed at the current time step, i.e., $m(O_{z_k}^c) > 0$.

The number of new-born particles ν_{new}^c for each cell c is selected based on its new-born object mass ϱ_{new}^c:

$$\nu_{new}^c = \left(\sum_{i=1}^{n_c^{\mathcal{G}}} \varrho_{new}^i \right)^{-1} \varrho_{new}^c \cdot \nu_{new} \tag{3.50}$$

where $n_c^{\mathcal{G}}$ denotes the total number of cells in the grid.

The set of new-born particles $\mathbf{P}_{new}^c$ inside the cell c is divided into two subsets: the set of new-born particles $\mathbf{P}_{new}^{\vartheta,c}$ associated with the velocity measurement $\vartheta_{z_k}^c$, and the set of not associated particles $\mathbf{P}_{new}^{\bar{\vartheta},c}$. The number of particles $\nu_{new}^{\vartheta,c}$ associated with $\vartheta_{z_k}^c$ is calculated by:

$$\nu_{new}^{\vartheta,c} = \lceil \nu_{new}^c \cdot \alpha_{z_k}^{\vartheta,c} \rceil \tag{3.51}$$

and the number of not associated particles $\nu_{new}^{\bar{\vartheta},c}$ is obtained using:

$$\nu_{new}^{\bar{\vartheta},c} = \nu_{new}^c - \nu_{new}^{\vartheta,c} \tag{3.52}$$

The position $\boldsymbol{x}^\rho$ of the particles in both sets, $\mathbf{P}_{new}^{\vartheta,c}$ and $\mathbf{P}_{new}^{\bar{\vartheta},c}$, is distributed uniformly within the area covered by the cell c (3.29). Particles associated with a velocity measurement initialize their velocity according to it while non-associated particles initialize it according to a Gaussian distribution with zero mean.

The new-born objects' mass ϱ_{new}^c is equally distributed along all new-born particles. Therefore, the weight of a new-born particle is defined as:

$$\omega_{new}^{\rho,c} = \frac{\varrho_{new}^c}{\nu_{new}^c} \tag{3.53}$$

Lastly, the set of particles is resampled based on the particles' weights. The higher the weight, the higher the chances of a particle being sampled. Particles can be sampled more than once. Given the new set of particles in each cell, their weights are set all equal and normalized to sum up to the updated occupied mass, i.e., complying with (3.30).

3.6.4. Dynamic Occupancy Grid output visualization

The DOG is composed of the occupancy and dynamic states. The occupancy state is represented by evidence masses for occupied, free and unknown, while the dynamic state describes the velocity and orientation of the obstacles. To visualize this environment estimation comprehensively, this thesis represents the occupancy state with the classical grayscale, and the dynamic state of dynamic cells using a color code.

The occupancy is represented using the occupancy probability $P(O)$, that can be derived from the evidence masses of occupancy $m(O)$ and free space $m(F)$ using the pignistic transformation [138]:

$$P(O) = m(O) + 0.5 \cdot \Big(1 - m(O) - m(F)\Big) \tag{3.54}$$

The dynamic state is only represented in those cells that present sufficient occupied evidence $m(O) > \mathcal{T}^o$ and have been classified as dynamic by the classification method introduced in Section 3.6.3.4. The cells that fulfill these two conditions are colored according to the mean orientation of the cell and its brightness is proportional to the velocity magnitude.

Figure 3.12 displays an example of a DOG following this representation approach.

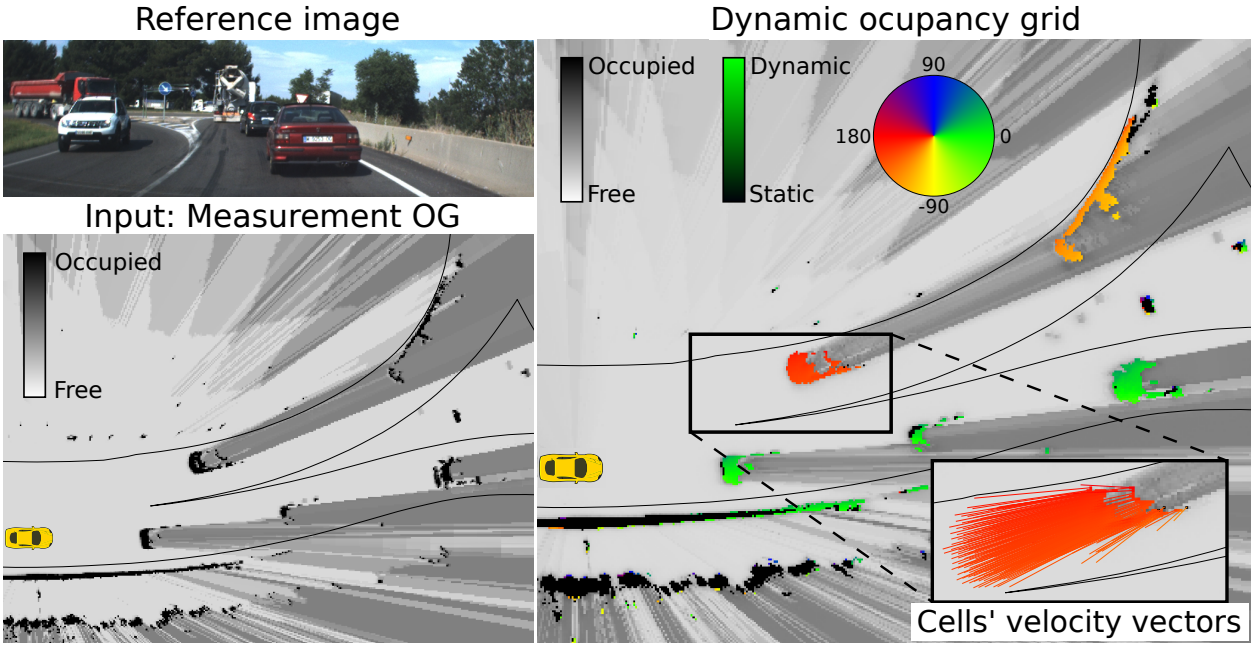


Figure 3.12: Dynamic Occupancy Grid example.

3.7. Height and Road Grids

As briefly introduced in Section 3.3, the height and road maps consist of a grid-based representation of the input data at the current iteration. This information is not intended to be a filtered representation of the surrounding environment but an input data for the subsequent tasks. Figure 3.13 shows an example of these two grid maps.

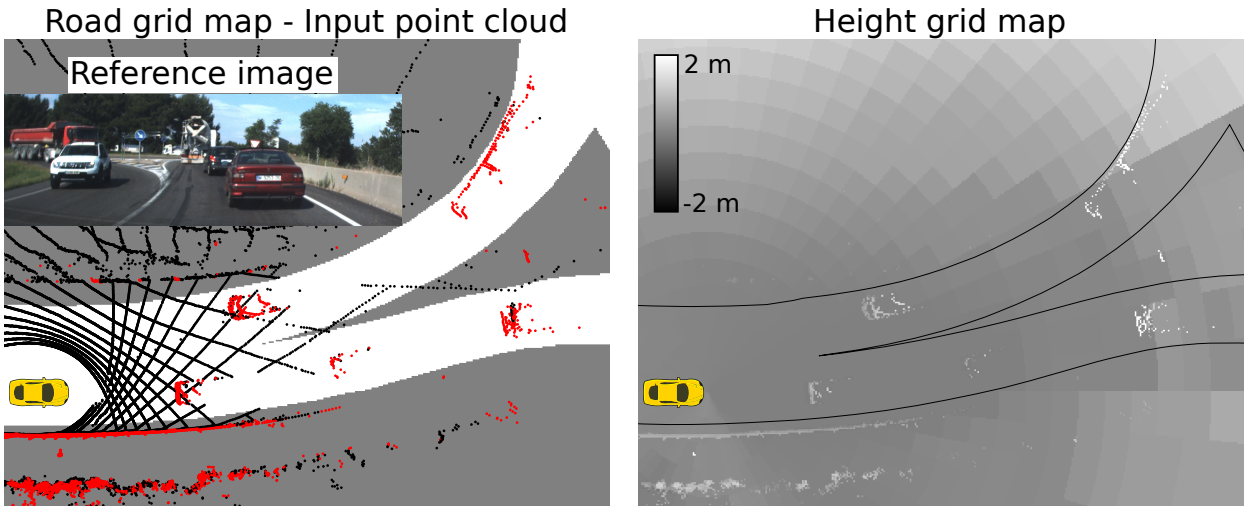


Figure 3.13: Example of the Road Grid (left) and Height Grid (right).

3.7.1. Height Grid

The HG $\mathcal{G}^H$ stores the height values corresponding to the obstacles and the ground surrounding the ego-vehicle. This information is used as input data for modeling the free space (see Section 3.4.2) and object classification tasks (see Sections 4.5.2 and 4.6.3).

This grid is calculated using the data computed during the first stage: point cloud obstacle–ground classification and ground height estimation. Therefore, for a cell c the height corresponding to the obstacles $h^c(obs)$ is selected as the height of the highest *obstacle* point within it. The ground height $h^c(gr)$ is obtained by computing the correspondence between the cartesian cells of $\mathcal{G}$ and the nodes of G (polar cells)—see Section 2.5.

3.7.2. Road Grid

The RG $\mathcal{G}^R$ is in charge of indicating whether a cell corresponds to the road or not. It will be used as input for classification tasks in Section 4.5.2.

As stated in Appendix A.1.2, this thesis counts with a high-definition digital map. In an offline step, this digital map is rasterized into a binary grid $\mathcal{G}^{R,global}$ with a known reference system in global coordinates. Each cell is labeled as $\ell^R = road$ if its location is inside the road, or $\ell^R = offroad$ if it is outside. The local RG $\mathcal{G}^R$ is then computed at each iteration of online processing by determining

the correspondence between each cell $c \in \mathcal{G}^R$ and a cell $c^{global} \in \mathcal{G}^{R,global}$.

3.8. Object-based evaluation framework for Dynamic Occupancy Grids

As already introduced in Section 3.2.3, there is no standardized quantitative evaluation for OGs in the field of autonomous driving. On the contrary, perception of the environment from an object-level representation perspective has well-established performance metrics. This section presents a novel evaluation framework based on application-specific analysis regarding the task of object detection and state estimation in OGs.

The proposed strategy extracts objects from the grid and evaluates the results using object-level metrics. For this purpose, (i) objects are clustered and described at an object-level using common and generalizable strategies and (ii) the well-established object metrics are adapted to properly address common issues of object detection in grid maps.

In the following sections, the proposed evaluation framework is presented. First, the evaluation problem is described along with the required data. Then, the proposed evaluation is presented. The proposal includes three evaluation perspectives: (i) scores concerning object clustering capability, (ii) scores concerning object features estimation, and (iii) overall score.

3.8.1. Evaluation framework scope

The intended evaluation requires an object-level representation of the detected obstacles. However, OGs represent the detected objects as groups of independent cells, without any object-based assumption. It is common to find objects described by incomplete shapes, represented by one or more groups of cells, or conformed by cells with non-homogeneous velocity vectors. Moreover, the transformation from cell-level to object-level representation is a complex task without a common approach.

The proposed evaluation framework seeks to provide a method that clusters objects from the grid and describes them at object-level in a way that the obtained conclusions can be transferable to other object segmentation and estimation strategies. The objects calculated following this approach are not intended to be the best possible estimations, but a repeatable process that allows to measure how easily and accurately these tasks—object segmentation and object-level representation—can be performed.

Commonly used object-level metrics are intended to evaluate the best results concerning the accuracy of object-level representation. On the contrary, this framework is intended to use object-level representation to assess the quality of an OG. Therefore, the commonly used metrics have to be adapted to the evaluation purpose. Taking this into account, the proposed framework seeks to provide an adequate and useful evaluation method by (i) evaluating multiple aspects concerning frequent clustering and object-level features estimation issues, (ii) selecting and implementing the

algorithms based on widely accepted criteria, and (iii) employing several tools to evaluate the same feature from different perspectives.

3.8.2. Ground truth data

The proposed evaluation requires a dataset containing sensor data to compute an OG and a ground truth including the road users in the scene. The state of the Ground Truth Objects (GTO) must be described by (i) a 2D oriented bounding box—denoting position and shape— and (ii) a velocity vector.

3.8.2.1. Footprint Quality Reference

As already introduced in Section 3.8.1, in OGs it is common to find objects described by incomplete shapes. This is because OGs do not generally make object-level shape assumptions, but rather directly represent the sensors' information. In the case of LiDAR sensors, this results in a footprint representation of the object's side where the laser beams are reflected. Depending on the detected footprint, the object will be better or worse described in terms of location and shape. This issue was already introduced in Section 2.6.3 during the evaluation of the proposed obstacle–ground classification strategy's ability to detect vehicles.

Given the importance of the footprint, an OG evaluation from an object detection perspective should take it into account. Therefore, if a dataset containing LiDAR data with semantically labeled points is available, e.g. [14, 145], a score of the expected object estimation quality is proposed: the Footprint Quality Reference. This score estimates the quality with which a sensed footprint models an object's shape and location, as the IoU between the bounding box of the ground truth and the convex hull of the impacts on the object. Henceforth, this reference metric is denoted as $IoU(PC)$. The higher the $IoU(PC)$ value is, the better the object is described and the better results from object estimation are expected. The example shown in Figure 3.14 clearly illustrates the functioning and usefulness of this score: a better estimation regarding position and shape is expected for the right object since its $IoU(PC)$ score is higher.

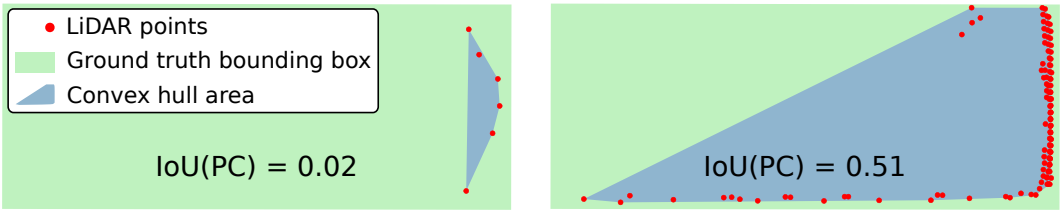


Figure 3.14: Footprint Quality Reference example. Depending on the detected footprint, the object is worse (left) or better (right) described.

3.8.2.2. Detectable objects

Typically, ground truth data is derived using multiple types of sensors and forward/backward temporal filtering. Therefore, some ground truth objects may not be detectable with the sensor data used to compute the evaluated OG.

Given that the proposed perception framework is intended for LiDAR data, only objects that have at least three impacts are considered detectable at the current iteration. This criterion is defined based on the minimum number of points required to compute the convex hull of the $IoU(PC)$ score.

3.8.3. Object segmentation-based evaluation

Inferring object hypotheses from OGs is not a straightforward process. All the cells corresponding to each obstacle have to be identified and grouped without merging with other objects. Moreover, the obtained clusters should provide enough evidence of the objects' existence in order to be considered valid. Therefore, the object segmentation-based evaluation seeks to evaluate the results of the OG by assessing how easily this task can be performed.

3.8.3.1. Proximity-based object segmentation method

As fully described in Chapter 4, object segmentation methods usually combine several criteria and information resources. However, since OGs are a discretization of the space, most object segmentation methods take into account the proximity between occupied cells as one of the main criteria. As an example, in the following, three examples of different techniques employing this criterion during their process are given. [39] clusters cells using a Density-based Spatial Clustering of Applications with Noise (DBSCAN) [34]. In [141] objects are extracted using tracked objects' information, but the obtained clusters are additionally expanded by evaluating adjacent cells. [50] uses deep learning to identify objects; however, during training data generation, a Connected Component algorithm [117] is employed.

Given the above, a simple but generalizable distance-based clustering is employed to segment objects from the OG. Henceforth, the objects segmented in this way are referred to as Distance-based Clustered Objects (DCO).

A detailed description of the clustering procedure is provided in Section 4.3, but it can be briefly explained as follows: cells with an occupied mass higher than a certain threshold are clustered using the Connected Components algorithm. Given that the only criterion applied is the proximity between cells, an 8-connected neighborhood kernel is employed assuming that: (i) compact clusters better model the real object's shape and (ii) object merging is minimized as much as possible.

Since this evaluation addresses scenarios with pedestrians, which can be represented by few cells, clusters are accepted despite their size, i.e., a DCO can be constituted by a single cell. However, taking into account that OGs are prone to noise in occluded areas, only the clusters that have been observed during the current iteration are considered, i.e., $m(O_{z_k}^{cluster}) > 0$.

3.8.3.2. Distance-based clustered objects description

For each DCO, a simple description is calculated. To consistently represent the model-free estimation of objects' location and shape in OG approaches, for every accepted DCO, the convex hull of the cells is computed. In addition to that, if the evaluated grid is a DOG, a label concerning the dynamic state of the DCO—*dynamic* or *static*—is calculated with respect to the weighted average velocity of all cells and the threshold value $\mathcal{T}_v^{static}$.

In this way, DCOs are defined by three features: (i) the number of clustered cells, (ii) the convex hull of the clustered cells, and (iii) a label concerning its dynamic classification.

3.8.3.3. Detection function

The detection function determines which GTOs are considered as detected and by which DCO. This is an important process that must be defined for the evaluated framework.

As already explained, in OGs it is common to find objects split into groups or represented by small footprints. Therefore, detection functions based on the distance between centers or IoU, such as the ones used in [14] and [38], may not be suitable. Conversely, LiDAR data is precise in terms of points' position and can be used to model the space into occupied and free areas. As a result, an accurate description of the environment regarding occupancy is expected, i.e., occupied space in the location of LiDAR-sensed objects and free space where beams have passed through.

Based on this reasoning, a GTO is therefore considered as detected if it overlaps with the cells of at least one DCO, i.e., if occupied space has been modeled in its position. If more than one DCO overlaps the GTO, the one with the highest IoU is associated. The IoU is computed between the bounding box of the GTO and the convex hull of the DCO's cells:

$$IoU(DCO) = \frac{\boldsymbol{A}(DCO \cap GTO)}{\boldsymbol{A}(DCO \cup GTO)} \tag{3.55}$$

where $\boldsymbol{A}(DCO \cap GTO)$ denotes the area of the intersection and $\boldsymbol{A}(DCO \cup GTO)$ the area of the union.

Figure 3.15 shows an example of this detection function. Three vehicles are detected and represented in the grid as several groups of cells. Only the DCOs that best describe the shape and position of the vehicles, based on the IoU, are associated with the GTOs.

3.8.3.4. Segmentation scores

The segmentation scores are divided into three elements: (i) Occupancy Detection Capability Score, (ii) Quality Clustering Score, and (iii) Dynamic Segmentation Score.

The Occupancy Detection Capability Score ($ODCS$) measures the probability of estimating occupied space in the objects' location. It is the most basic indicator and it is based on the idea that

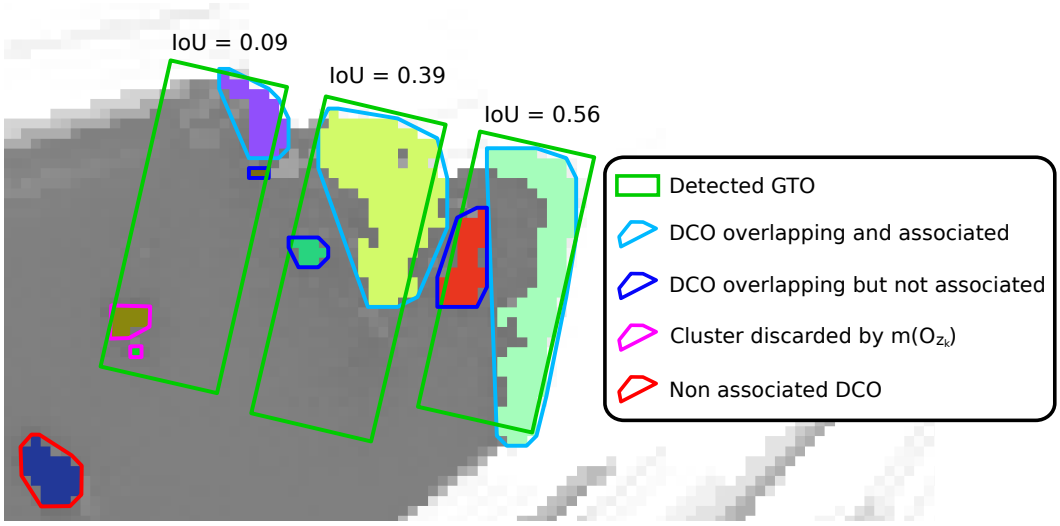


Figure 3.15: Example of detection function in object detection-based evaluation.

if occupied space has been estimated, then the object's information can be inferred.

The *ODCS* relates the number of detected GTOs, $n^{GTO}_{detected}$, with the total number of GTOs, n^{GTO}:

$$ODCS = \frac{n^{GTO}_{detected}}{n^{GTO}} \tag{3.56}$$

where $n^{GTO}_{detected}$ is obtained applying the detection function presented in Section 3.8.3.3.

The *ODCS* measures the detection capability. However, it is a permissive score that does not take into account relevant segmentation issues, such as noise, split, or merge. These issues can affect the accuracy of the estimation and subsequent steps, e.g. filtering or data association. Therefore, to measure the quality of the clustered objects three Quality Clustering Scores (QCS) are proposed:

- Quality Clustering Noise Score (QCS_{noise}): it is a common practice in clustering tasks to define a minimum size to consider a reliable cluster. Therefore, a score measuring the probability of wrongly considering a DCO as noise is defined: every detected GTO is labeled as *noise* if the number of cells of its associated DCO is less than the threshold $n^{min,c}_{noise}$. The noise score is computed as:

  $$QCS_{noise} = 1 - \frac{n^{GTO}_{noise}}{n^{GTO}_{detected}} \tag{3.57}$$

  where n^{GTO}_{noise} is the number of objects labeled as *noise*.

- Quality Clustering Merge Score (QCS_{merge}): objects close to each other can be merged into a single one, thus obtaining a bad object segmentation of both. A detected GTO is labeled as *merged* if (i) its associated DCO overlaps with other GTOs or (ii) its area $\boldsymbol{A}(GTO)$ is much smaller than the area of its associated DCO, $\boldsymbol{A}(DCO)$ (this latter criterion seeks to control the

merge with objects unlabeled by the ground truth, e.g. vegetation):

$$\frac{\boldsymbol{A}(GTO)}{\boldsymbol{A}(DCO)} < \varphi_{merge} \tag{3.58}$$

being $\varphi_{merge} \in [0, 1]$.

Therefore, the merge score is computed as:

$$QCS_{merge} = 1 - \frac{n_{merge}^{GTO}}{n_{detected}^{GTO}} \tag{3.59}$$

where n_{merge}^{GTO} is the number of objects labeled as *merged*.

- Quality Clustering Split Score (QCS_{split}): as explained, when working with LiDAR data it is common to find objects divided into several parts—for example, the rear and the roof of a vehicle might be detected individually. For this reason, OGs are prone to object splitting. This issue should be taken into consideration since it may lead to wrong object estimations and can provide a good indication of the objects' compactness. A GTO is labeled as *split* if it is overlapped by more than one DCO. The split score is computed as:

$$QCS_{split} = 1 - \frac{n_{split}^{GTO}}{n_{detected}^{GTO}} \tag{3.60}$$

where n_{split}^{GTO} is the number of objects labeled as *split*.

These three Quality Clustering Scores are jointly represented as the Joint Quality Clustering Score ($JQCS$), which expresses the probability that a reliable and representative object segmentation has been obtained:

$$JQCS = \frac{1}{3} \sum_{i=1}^{3} QCS_i \tag{3.61}$$

$JQCS$ does not provide information about how representative the resulting clusters are of the size and location of the actual object; instead, it assesses the quality of the obtained clusters based on how successfully the clustering stage was carried out.

Therefore, in addition to the above three metrics, the IoU between the bounding box of every detected GTO and the convex hull of its associated DCO, calculated as DCO's descriptor, is also taken into account:

$$MIoU(DCO) = \frac{1}{n_{detected}^{GTO}} \sum_{i=1}^{n_{detected}^{GTO}} IoU(DCO)_i \tag{3.62}$$

The last segmentation score addresses the ability to correctly classify DCOs into *static* or *dynamic*. Therefore, every detected GTO is considered as:

- Dynamic True Positive ($DynTP$) if the GTO and its associated DCO are both labeled as

dynamic.

- Dynamic False Positive ($DynFP$) if the GTO is *static* and the DCO is *dynamic*.
- Dynamic True Negative ($DynTN$) if both are *static*.
- Dynamic False Negative ($DynFN$) if the GTO is *dynamic*, but the DCO is *static*.

This classification is used to obtain the Dynamic Segmentation Score which is computed as the F1 score (which measures the classification accuracy taking into account Precision and Recall):

$$F_1^{Dyn} = \frac{2DynTP}{2DynTP + DynFP + DynFN} \tag{3.63}$$

Figure 3.16 shows an example of the proposed scores. This example is particularly useful to demonstrate how the combination of multiple metrics provides a more comprehensive evaluation of the results. Although every road user in this scene is counted as detected given that occupied space has been modeled inside its bounding boxes, the quality of the detections varies significantly. Some GTOs are correctly represented by a single DCO, while others are represented by several DCOs (split). Moreover, detecting a GTO with a single DCO is not a guarantee of obtaining a good representation of the object. Examples of this are the GTOs highlighted with a magenta dashed circle. The GTO (1) is accurately represented in terms of shape and position, but not regarding velocity; as a result, it is considered a dynamic false positive. On the contrary, the GTO (2) is correctly classified as *static*, but the side of the DCO barely provides information about the object. This fact is correctly modeled by its IoU score: $IoU = 0.02$.

3.8.4. Object features-based evaluation

After the object segmentation task, each group of cells is represented at an object-level by a set of features, usually: 2D position, heading, bounding box-based shape, and 2D velocity vector. The object features-based evaluation aims at measuring how easy is to obtain good object features from the evaluated OG given an ideal clustering.

3.8.4.1. Ideal object segmentation method

In the evaluation presented in Section 3.8.3 the difficulties related to object segmentation in OGs (split and merge) have been exposed. Therefore, seeking to avoid the inherent clustering problems, an ideal clustering for the object features-based evaluation is applied by taking advantage of the ground truth's bounding boxes.

Similarly to Section 3.8.3.1, only the cells that have an occupied mass above the threshold $\mathcal{T}_o^{cl}$ are clustered. Nevertheless, instead of using the Connected Components clustering algorithm, the occupied cells inside the GTOs' bounding boxes are clustered and associated with them (in this way, the detection function is solved directly). Then, the obtained clusters are expanded to neighboring

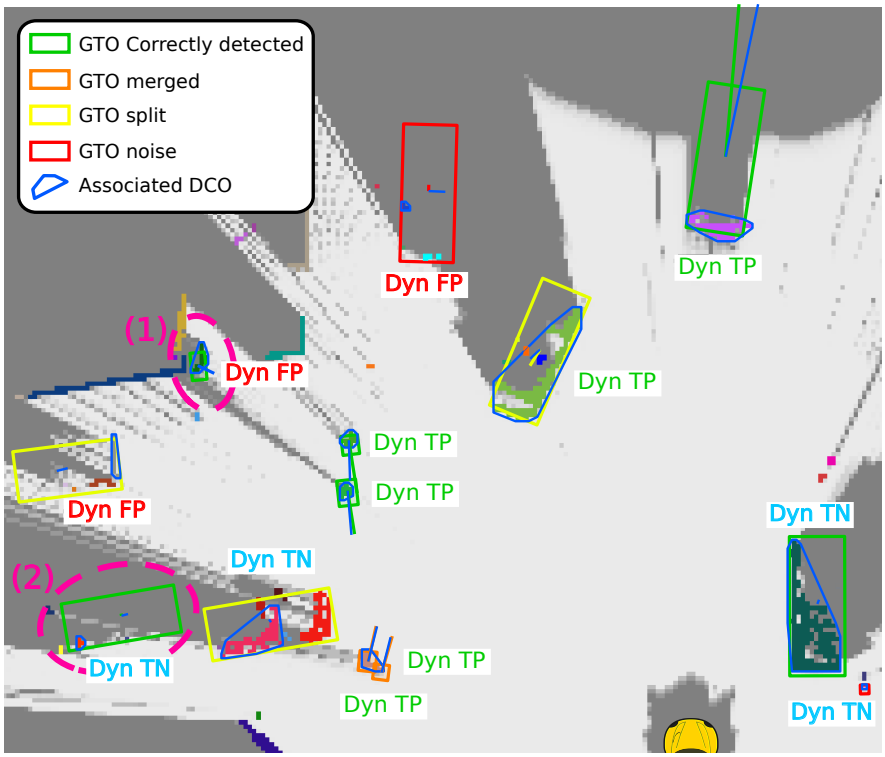


Figure 3.16: Example of object segmentation-based evaluation scores.

unclustered cells with an 8-connected neighborhood kernel. This expansion process is performed n^{expand} times. Henceforth, the objects segmented in this way are referred to as Ideally Clustered Objects (ICO).

Again, clusters are accepted despite their size, but only objects that have been observed in the current iteration are accepted.

3.8.4.2. Ideal clustered objects description

For each ICO, an object-level representation is generated concerning position, shape, and velocity. Therefore, an ICO is described by (i) a dynamic state $\boldsymbol{s}^{ICO}$ and (ii) the $IoU(ICO)$ computed between the bounding box of the GTO and the convex hull of the ICO's cells, which is calculated equivalently to (3.55).

The dynamic state represents the position and location in terms of an oriented bounding box, and the velocity as a 2D velocity vector. Thus:

$$\boldsymbol{s}^{ICO} = \left[x_{\mathrm{x}}, y_{\mathrm{y}}, \theta, l_{\mathrm{x}}, l_{\mathrm{y}}, v_{\mathrm{x}}, v_{\mathrm{y}}\right] \tag{3.64}$$

where x refers to the position, v to the velocity, θ and l to the box's orientation and sides length.

The velocity vector is estimated as the average of all cell's velocity, weighted with respect to their

occupied mass:

$$v_{\mathrm{x}}^{ICO} = \left(\sum_{i=1}^{n_c^{ICO}} m(O^{i,c}) \right)^{-1} \sum_{i=1}^{n_c^{ICO}} m(O^{i,c}) \cdot \mu_{v_{\mathrm{x}}}^{i,c} \tag{3.65}$$

being n_c^{ICO} the number of cells conforming the ICO, $\mu_{v_{\mathrm{x}}}^{i,c}$ the mean velocity of the cell c and $m(O^c)$ its occupied mass. v_y^{ICO} is calculated equivalently.

The oriented bounding box, which corresponds to the variables x_{x}, y_{y}, θ, l_{x} and l_{y}, is calculated using a search-based algorithm that selects the box that minimizes a variance criterion [173]. Briefly explained, a minimum rectangle gathering all ICO's cells is computed for every angle. The orientation that minimizes the variance in the distance between the cells and the rectangle's borders is selected. This method is selected because it only depends on the similarity between the estimated footprint and the assumed rectangular shape of vehicles. Thus, the utmost importance is placed on the correct estimation of the object's shape by the OG. Furthermore, this method can be conveniently modified to assess additional criteria, such as free space inside the box.

3.8.4.3. Object features scores

To evaluate the quality of the object-level features estimated for each ICO, the following metrics are computed:

- Mean Absolute and Square Translation Errors (MATE and MSTE): the location error is defined by the Euclidean distance between the bounding boxes' centers.
- Mean Absolute and Square Scale Errors (MASE and MSSE): the error in the estimated scale is computed as $(1 - IoU(box))$, being $IoU(box)$ the IoU between the GTO's bounding box and the ICO's bounding box, after location and orientation alignment.
- Mean Absolute and Square Box Orientation Errors (MABOE and MSBOE): the error in the orientation of the estimated bounding box is defined as the minimum absolute angle difference between the four possible headings of the bounding boxes.
- Mean Absolute and Square Velocity Errors (MAVE and MSVE): the error in the velocity is defined as the absolute velocity module difference.
- Mean Absolute and Square Velocity Orientation Errors (MAVOE and MSVOE): the error in the velocity orientation is calculated as the absolute angle difference between the velocity vectors. It is only computed for dynamic objects ($||v^{GTO}|| > \mathcal{T}_v^{static}$).

Figure 3.17 shows an illustrative example of the calculation of these five scores.

The five mean absolute errors (MAE) are jointly represented as the Joint Feature Mean Score (JFMS):

$$JFMS = \frac{1}{5} \sum_{i=1}^{5} \left(1 - \min \left(1, \frac{MAE_i}{E_i^{max}} \right) \right) \tag{3.66}$$

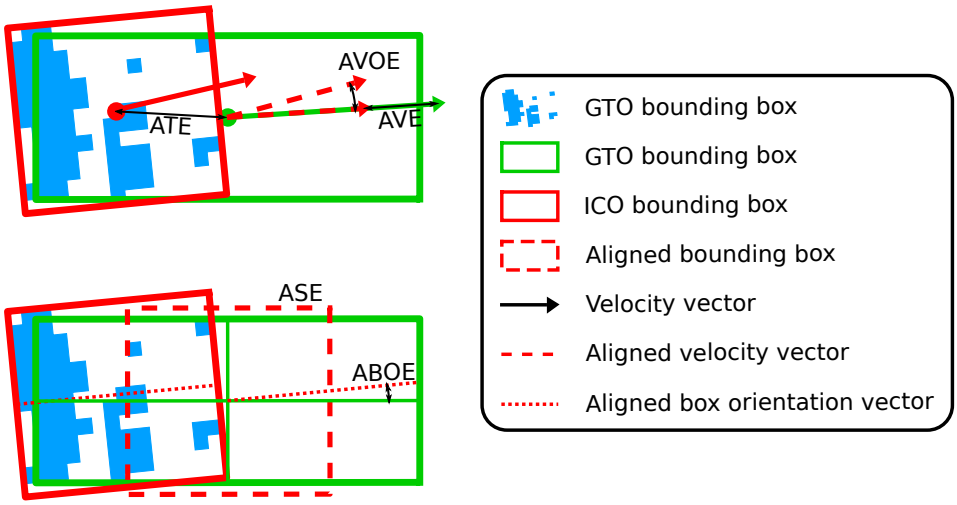


Figure 3.17: Example of object features-based evaluation scores.

where E_i^{max} refers to the maximum acceptable error for each object-level metric. This bound is used to normalize the five metrics and minimize the effect of noisy estimations.

The five mean square errors (MSE) are jointly represented as the Joint Feature Mean Square Score (JFMSS):

$$JFMSS = \frac{1}{5}\sum_{i=1}^{5}\left(1 - \min\left(1, \frac{MSE_i}{(E_i^{max})^2}\right)\right) \tag{3.67}$$

Additionally, as in Section 3.8.3.4, the IoU between the GTOs bounding box and the ICOs is computed and summarized as the average $MIoU(ICO)$:

$$MIoU(ICO) = \frac{1}{n_{detected}^{GTO}} \sum_{i=1}^{n_{detected}^{GTO}} IoU(ICO)_i \tag{3.68}$$

3.8.5. Overall score

Five different scores have been proposed in order to measure the quality of the OG from an object estimation perspective: (i) the number of detections ($ODCS$), (ii) the clustering quality ($JQCS$), (iii) the accuracy of the dynamic segmentation (F_1^{Dyn}), (iv) the object features estimation quality ($JFMS$) and (v) the accuracy of the modeled footprint ($MIoU$). Depending on the developer, each task may have higher or lower relevance. Nevertheless, in order to express the overall result with a single understandable score, the following Object Estimation Score (OES) is proposed:

$$OES = ODCS\frac{1}{4}\left(JQCS + F_1^{Dyn} + JFMS + MIoU\right) \tag{3.69}$$

being:

$$MIoU = \frac{1}{2}\left(MIoU(DCO) + MIoU(ICO)\right) \tag{3.70}$$

3.9. Evaluation results

3.9.1. Dynamic Occupancy Grid

This section introduces a set of experiments seeking to validate the results of the implemented DOG and the evaluation framework.

Figure 3.18 displays a measurement OG obtained using the AUTOPIA dataset. The occupancy probability $P(O)$, its confidence value α^{pO}, and the evidence masses $m(O)$ and $m(F)$ are shown. In order to illustrate the process of computing the measurement OG from different LiDARs, the occupancy probability and the confidence value calculated for four layers of the IBEO-Lux sensor and four layers of one of the VLP-16 sensors are shown. It can be seen that the proposed strategy provides a conservative estimation where the occupied space is favored over the free space. This can be clearly noticed in the plots corresponding to the confidence value or comparing the maximum values achieved for the masses for occupied and free. As explained in Section 3.4.2.1, the free space is weighted depending on the height of the beam. In the case of the IBEO-Lux sensor, the confidence of layer 1 is lower since its beams travel close to the ground surface and, hence, tend to pass beneath the vehicles. Similarly, some beams of the VLP-16 sensor are tilted upwards and, therefore, the free space modeled is not reliable, e.g. red dashed circle.

Figure 3.19 shows a similar experiment, but in this case, the objective is to illustrate the results of fusing LIDAR data with information received through V2X communications. As can be seen, if only LiDAR data is used, the shape of the bus is detected partially, only the rear. On the contrary, when including the information received by V2X communications, the complete shape is modeled, obtaining an improvement particularly useful for tasks such as motion planning or maneuver planning, e.g. for an overtaking maneuver. The fusion of the occupancy measurements obtained from both sources is also displayed at the right of the figure. Note that in this case, both sources complement each other. The LiDAR sensor cannot detect the complete shape of the bus, but it is able to model the free space surrounding it. The information received by communications models the complete bus shape but does not provide any information about its surrounding space.

Finally, Figure 3.20 shows the DOG corresponding to the measurement OG shown in Figure 3.18. It can be seen that the cells corresponding to dynamic obstacles are estimated as dynamic (colored cells), while the rest of the obstacles are mostly correctly represented by static cells (black). The vehicle highlighted by a red circle is changing from static to dynamic, thus its cells are also changing from static to dynamic. The velocity vectors of the individual cells are shown for the two dynamic vehicles at the right of the ego-vehicle, demonstrating the correct convergence of the cells to the real vehicle's velocity. Note that this estimation is obtained only using laser measurements. This is possible since particles with a dynamic state similar to the objects' real velocity are more likely

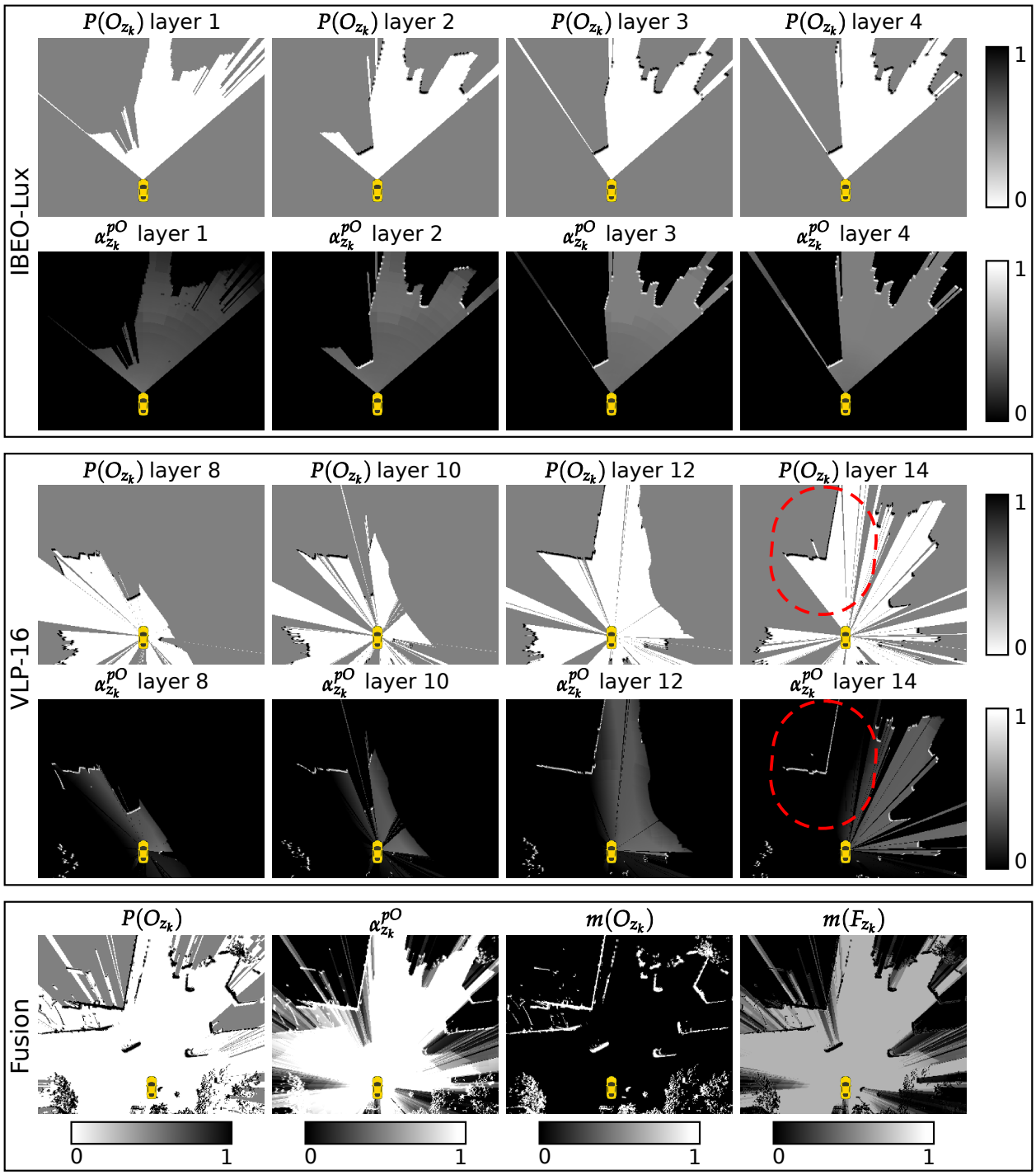


Figure 3.18: Example of the calculation of the measurement OG calculated with AUTOPIA's dataset.

to model correctly the objects' displacement and, therefore, remain with a high weight after the occupancy update. For example, a particle with a correct velocity vector is probably predicted into the space occupied by the object. Conversely, a particle with an incorrect velocity vector may be predicted into free space and, hence, obtain a weight penalization by the occupancy measurements.

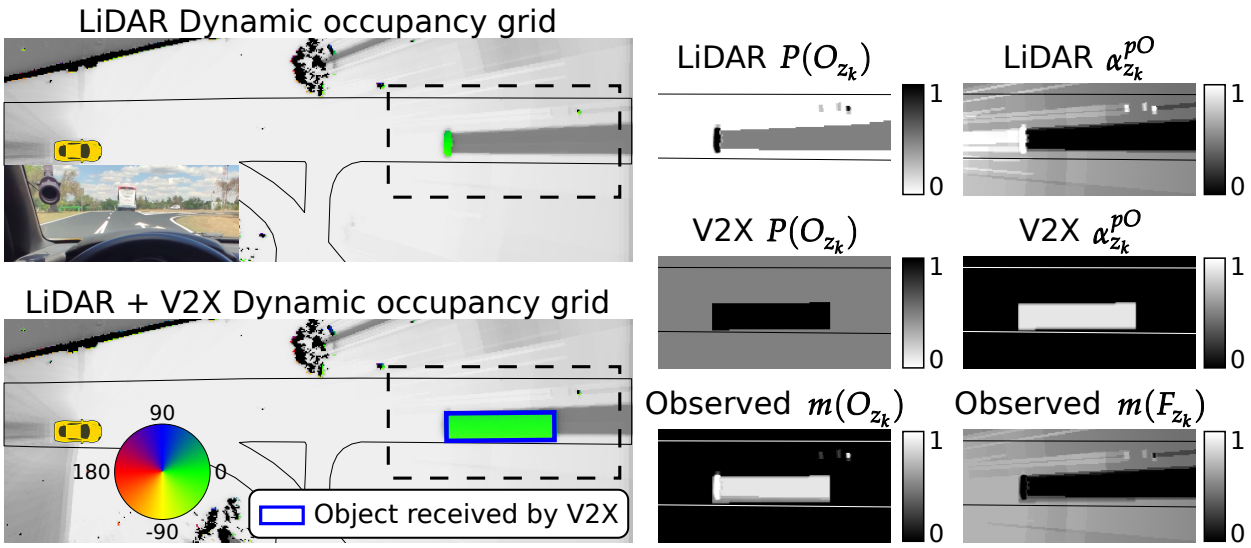


Figure 3.19: Example of a DOG computed with LiDAR data and V2X information. The ego-vehicle is following a bus that reports its own position through V2X communications.

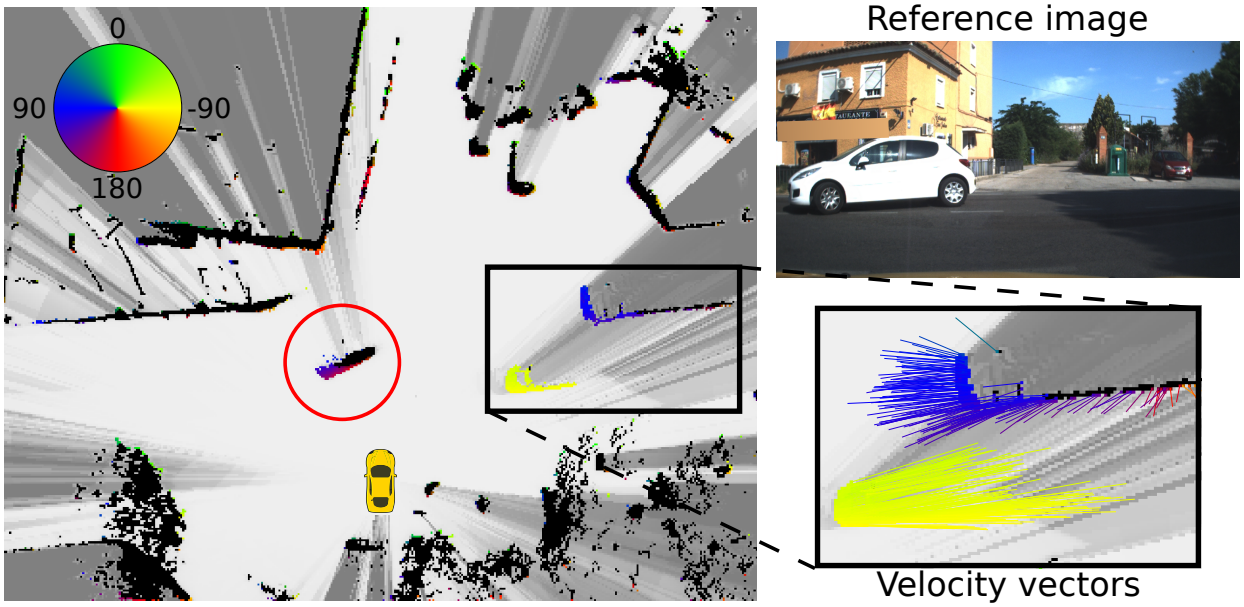


Figure 3.20: Example of DOG in an urban scenario with dynamic and static obstacles.

3.9.2. Evaluation framework – Measurement Occupancy Grid comparison

Taking advantage of the evaluation framework introduced in Section 3.8, this section provides quantitative results of the implemented DOG together with an evaluation of different strategies for computing the measurement OG.

3.9.2.1. Measurement Occupancy Grid comparison motivation

As introduced in Section 3.2, the measurement OG can be computed in multiple ways. Visual inspection evaluation is the most common OG evaluation approach. Nevertheless, as will be shown below, evaluation by visual inspection is complex and highly subjective. In fact, it motivated the development of this quantitative evaluation. In the following, the two most popular but opposing strategies for calculating a measurement OG are qualitatively compared: (i) the line drawing scan rendering method with an occupancy model based on the Dirac function and (ii) the polar grid scan rendering method with a Gaussian model. Figure 3.21 shows the result of both methods with nuScenes dataset.

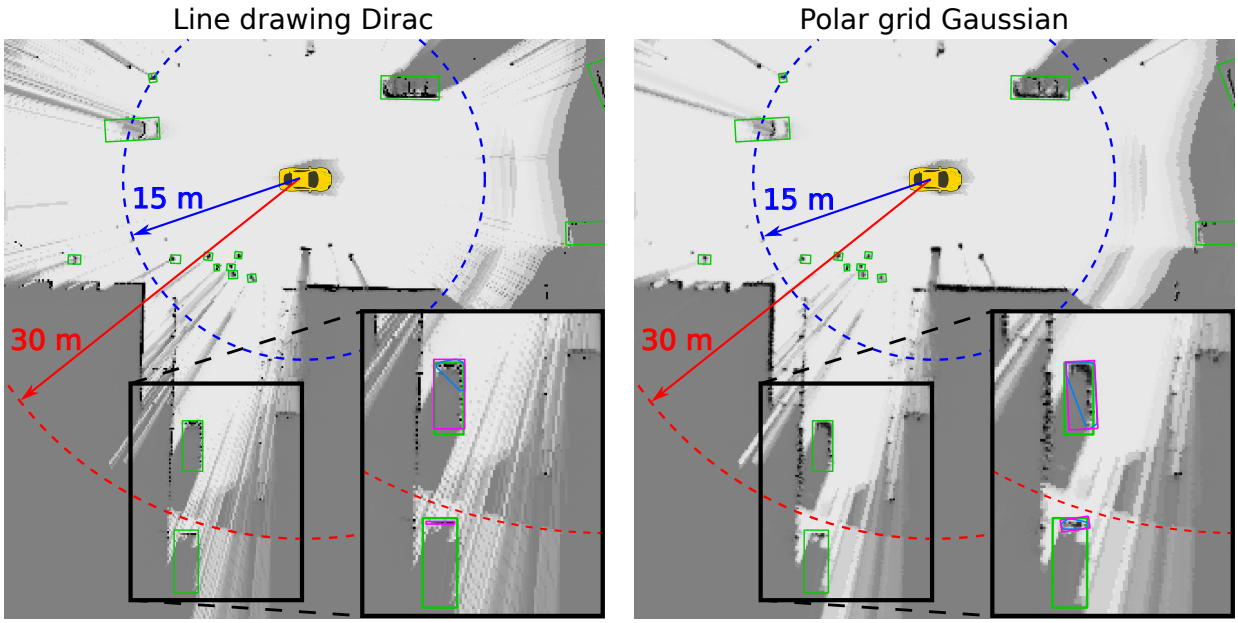


Figure 3.21: Comparison between line drawing Dirac-based and polar grid Gaussian-based OGs.

It can be seen with blue dashed circle that at short distances, both methods provide similar results. In this range, multiple beams traverse the same cell, thus the effect of the shape assumption is not very significant. The main differences are caused by the Dirac or Gaussian models. In the line drawing Dirac OG, object's footprints are thinner and cells have higher occupied values, while, for the polar Gaussian OG, objects are thicker but blurred.

At farther distances, red dashed circle, the space between the beams increases, and more notable differences appear between both methods. When applying line-based scan rendering algorithms, the space between beams is modeled as unknown. Thus, an irregular free space estimation and non-compact objects are generated. On the contrary, when applying angular-based algorithms, the space between beams is approximated by the angularly close estimated occupancy values. Therefore, a smoother occupancy distribution is obtained, but the object's shapes are slightly enlarged and distorted.

As shown in the vehicles highlighted in Figure 3.21, these differences influence object detection tasks. The polar Gaussian OG models the upper car with a compact footprint, leading to better object segmentation by distance-based clustering (blue line). On the contrary, when estimating the oriented bounding boxes, the line drawing Dirac OG obtains better results, since it does not distort the L-shape and I-shape captured by the LiDAR.

3.9.2.2. Measurement Occupancy Grids evaluated

Based on the results obtained in the previous section, the best strategy for calculating measurement OGs is assessed by addressing three factors: (i) the use of strategies addressing fast computation, model accuracy or balance between both, (ii) the shape assumed for the laser beam, line-based or angular-based, and (iii) the model for occupied space, Dirac or Gaussian model. Therefore, six scan rendering methods, three line-based and three angular-based, are selected, leading to a total of twelve measurement OGs.

In the following, the six scan rendering methods are briefly introduced and Figure 3.22 provides illustrative examples of the measurement OG achieved using each one of them. The three line-based methods evaluated are:

- **Line drawing-based method:** Bresenham's algorithm [12] is selected as representative of computationally efficient line-based methods. An example of a fast OG implementation using this algorithm can be found in [115].
- **Traversal-based method:** Traversal algorithms are considered computationally affordable and accurate, as, in contrast to line drawing algorithms, all the cells traversed by the beam are considered, compare the respective examples in Figures 3.22. In this work, cells are selected if the beams' angle is within the minimum and maximum angles of the cell. A variant with lower computational cost can be found in [110].
- **Weighted line-based method:** Xiaolin Wu's algorithm [164] is chosen for line-based accurate method. An example of an OG obtained using this algorithm and an example of its application can be found in [134].

The three angular-based methods are:

- **Beam-by-beam method:** The work of [107] named as *Beam-by-beam* is selected as representative of fast angular-based.
- **Polar grid-based method:** Polar grid-based methods are considered a balance between accuracy and computational cost. Illustrative examples of polar OGs can be found in [83, 167].
- **Weighted angular sector-based method:** The method introduced in Section 3.4.1.1 is selected as angular-based accurate method.

The calculation of the occupancy along the distance traveled by the beam is modeled following the method introduced in Section 3.4.2.1, but for the Dirac model, equations (3.8) and (3.9) are

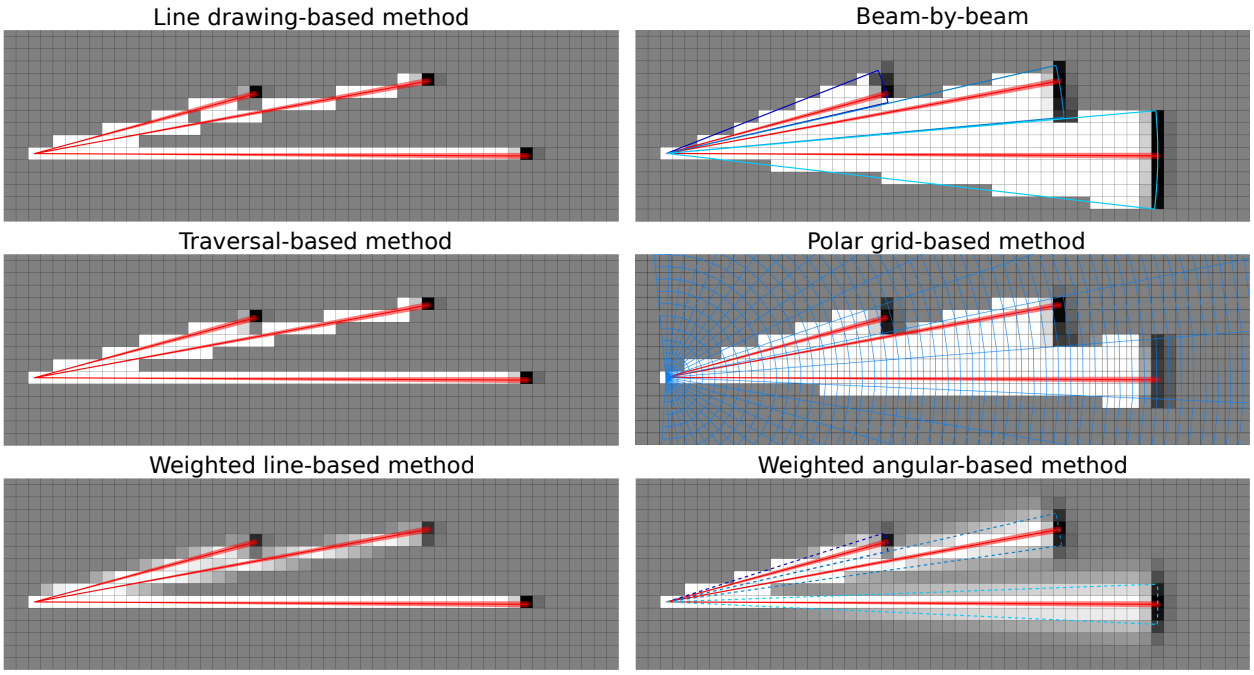


Figure 3.22: Illustrative example of the scan rendering tested methods using Gaussian occupied space model. Occupancy probability is drawn in grayscale. Laser beams are drawn as thin red cones. For the angular-based methods, the area covered by each beam is colored with bluish lines.

modified accordingly. Moreover, the function weighting the free space based on the beam's height $f(z_h^{\beta,c})$ is simplified to a height thresholding that divides the free space information into reliable or not—the implementation details of these modifications are provided in [60].

Generally, the methods using the line shape assumption or the Dirac function model the occupied space only in the cell that gathers the impact. In contrast, methods using the angular sector shape assumption or the Gaussian function (with a standard deviation correctly defined according to the cell size) tend to propagate the occupied space along the impact's vicinity. Essentially, their results are similar to the ones shown in Figure 3.21.

3.9.2.3. Quantitative evaluation

As exposed in Section 3.9.2.1, visual conclusions are hardly achievable and the differences between methods can lead to both advantages and disadvantages. Therefore, in order to obtain concluding results, a quantitative evaluation has to be performed. In the following, the DOGs resulting from the twelve measurement OGs are evaluated using the proposed evaluation method.

Table 3.2 shows the obtained results corresponding to the object detection capability evaluation. As can be seen, all methods detect approximately the same number of objects. Nevertheless, DOGs employing the Gaussian model with the weighted line, polar grid, and weighted angular methods tend to perform the best with respect to the scores of noise, split, and IoU and the worst concerning the merge score. On the contrary, DOGs employing the Dirac model and line drawing, traversal,

and beam-by-beam methods behave reversely. These results can be explained by the amount of occupied space modeled by each strategy. Line drawing- and traversal-based methods with Dirac model only set as occupied the cells gathering the laser impacts. As a result, they tend to model thin occupied spaces and unknown or free gaps between occupied cells (see Figure 3.21). The Beam-by-beam approach performs similarly because the assumption that each cell is only updated by one beam per layer makes that, at closer distances, some information is dismissed and objects can also be modeled with non-occupied gaps in between. While these facts benefit the objects differentiation (n^{GTO}_{merge}), they also result in less representative footprints (n^{GTO}_{noise}, n^{GTO}_{split}, $MIoU(DCO)$). Regarding the dynamic classification (F_1^{Dyn}), the variations on the scan rendering method do not severely affect the dynamic estimation performed by the DOG's underlying PF, but, again, the Gaussian model seems to perform slightly better.

Table 3.2: Detection capability quantitative evaluation. The results for each score are colored from red (worse) to green (best). LD: Line Drawing, T: Traversal, WL: Weighted Line, BB: Beam-by-beam, PG: Polar Grid, WA: Weighted Angular.

		$n^{GTO}_{detected}$	n^{GTO}_{noise}	n^{GTO}_{merge}	n^{GTO}_{split}	$JQCS$	F_1^{Dyn}	$MIoU(DCO)$
LD	**Dirac**	4989	95	420	2807	0.778	0.851	0.299
	Gaus	4992	19	491	2476	0.801	0.853	0.336
T	**Dirac**	4987	127	358	2930	0.772	0.848	0.280
	Gaus	4993	5	501	2385	0.807	0.856	0.343
WL	**Dirac**	4993	25	484	2568	0.795	0.854	0.319
	Gaus	4990	4	533	2256	0.813	0.860	0.349
BB	**Dirac**	4991	65	356	2869	0.780	0.844	0.276
	Gaus	4993	12	472	2525	0.799	0.859	0.327
PG	**Dirac**	4993	29	484	2507	0.798	0.837	0.322
	Gaus	4993	4	578	2237	0.812	0.847	0.352
WA	**Dirac**	4993	13	457	2572	0.797	0.843	0.319
	Gaus	4993	3	536	2282	0.812	0.854	0.348

Table 3.3 summarizes the results obtained for the object features evaluation. Recall that the MAVOE score is only computed for dynamic objects and the MABOE score does not include pedestrians, only vehicles. It can be seen that, with an ideal clustering, all DOGs present similar object feature estimation capabilities. The Gaussian model performs better than the Dirac model; and the weighted line, polar grid, and weighted angular sector methods obtain slightly better results. Nevertheless, no strategy stands out from the others significantly in all metrics.

Lastly, Table 3.4 shows the overall score OES. It can be easily noticed that methods employing the Gaussian model perform better. The differences between the line- and angular-based methods are minor, but there is a clear tendency to obtain better results when using more accurate scan rendering methods.

Table 3.3: Object features quality evaluation. The results for each score are colored from red (worst) to green (best). LD: Line Drawing, T: Traversal, WL: Weighted Line, BB: Beam-by-beam, PG: Polar Grid, WA: Weighted Angular.

		MATE m	MASE	MAVE m/s	MAVOE °	MABOE °	JFMS	JFMSS	$MIoU(ICO)$
LD	**Dirac**	0.543	0.623	0.568	10.510	4.586	0.799	0.875	0.435
	Gaus	0.535	0.586	0.568	10.382	4.552	0.807	0.883	0.459
T	**Dirac**	0.540	0.637	0.578	10.752	4.584	0.795	0.870	0.421
	Gaus	0.524	0.586	0.571	9.951	4.367	0.809	0.885	0.466
WL	**Dirac**	0.543	0.598	0.563	10.236	4.575	0.805	0.881	0.451
	Gaus	0.531	0.583	0.557	10.021	4.407	0.809	0.885	0.467
BB	**Dirac**	0.553	0.617	0.571	10.699	5.045	0.797	0.876	0.432
	Gaus	0.532	0.592	0.555	10.397	4.868	0.805	0.883	0.459
PG	**Dirac**	0.536	0.589	0.603	11.374	4.664	0.803	0.882	0.455
	Gaus	0.533	0.560	0.574	10.533	4.570	0.812	0.891	0.474
WA	**Dirac**	0.531	0.593	0.579	10.352	4.722	0.804	0.882	0.456
	Gaus	0.522	0.574	0.566	10.068	4.402	0.811	0.888	0.473

Table 3.4: Overall result summarized by the Object Estimation Score (OES).

	Line drawing	Traversal	Weighted line	Beam-by-beam	Polar grid	Weighted angular
Dirac	0.698	0.691	0.710	0.694	0.707	0.708
Gaussian	0.714	0.719	0.722	0.714	0.721	0.722

3.9.2.4. Evaluation with respect to the IOU(PC) and radial distance

As already explained in Sections 2.6.3, 3.8.2.1 and 3.9.2.1, the quality of object detection is strongly related to the distance between the sensor and the object, and the footprint captured by the laser beams. Therefore, in the following, the main evaluation scores are evaluated against the distance to the sensor and the $IoU(PC)$ metric.

Figure 3.23 shows the results obtained with respect to the distance to the sensor. As distance increases, the complexity of the estimation also rises. This is clearly reflected in a decline in the achieved results, especially in the estimation of object features (JFMS) and footprint (IoU). Regarding the quality of the clustering (JQCS), the performance of line-based methods is more adversely affected, whereas the angular-based methods demonstrate a comparatively less distance-dependent outcome. The results begin to diverge at approximately 15 m supporting the observations made with the qualitative analysis of Figure 3.21.

Figure 3.24 displays the obtained results against the footprint captured by the LiDAR sensor – $IoU(PC)$ metric. It can be seen how the JQCS decreases as the $IoU(PC)$ increases. This can be explained by the fact that objects perceived with small footprints are unlikely to be split or merged, contrary to objects with higher footprints. Conversely, the JFMS improves with the increase of the $IoU(PC)$. This is an expected result since the JFMS is computed with ideal clustering and the better the object is described, the easier the object's features estimation is. Note in this connection that capturing the complete footprint facilitates the estimation of the center point and size.

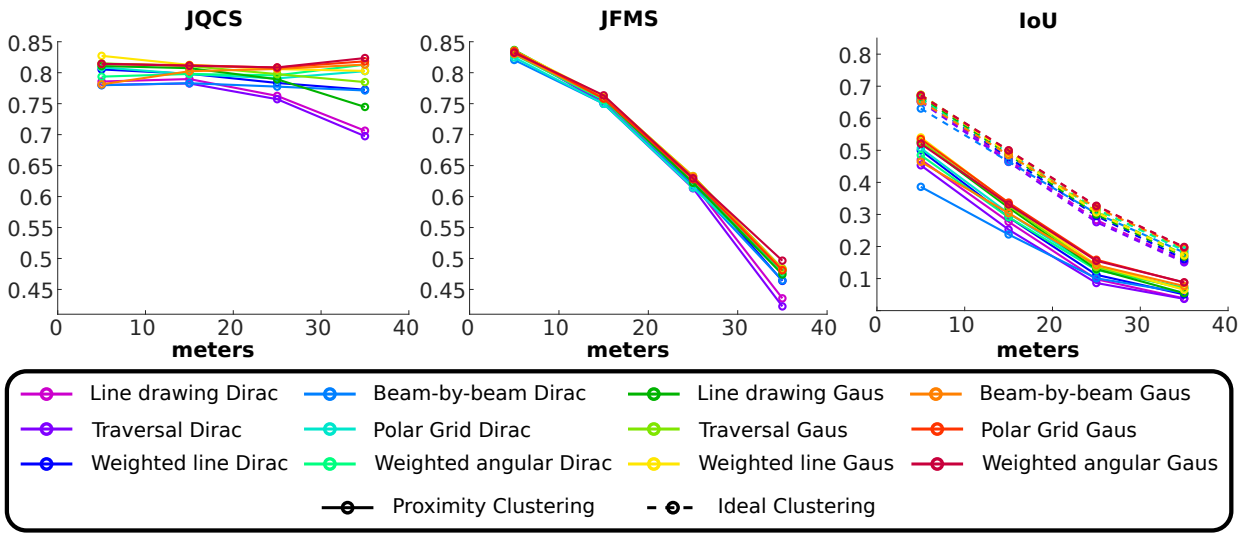


Figure 3.23: Evaluation results represented against the distance to the ego-vehicle.

Regarding the quality of the estimated footprint (IoU), the results obtained with both clustering methods increase with the quality of the footprint captured by the sensor. The results obtained for the ideal clustering (dashed lines) show that in general the OG representation enhances the object's shape and location estimation with respect to the LiDAR footprint. At the maximum $IoU(PC)$ values, the grid representation results primarily decrease because occupied space is modeled outside the GTOs' bounding boxes. In contrast, the results for the distance-based clustering (solid lines) show that the obtained clusters in general perform worse than the $IoU(PC)$. This is mainly explained because the obtained clusters do not gather all the occupied space modeled for the object (cluster splitting), while the $IoU(PC)$ is computed using ideal clustering.

Overall, the methods: polar grid, weighted angular, and weighted line, with a Gaussian model tend to obtain the best results.

3.9.2.5. Measurement Occupancy Grid evaluation conclusions

All tested methods exhibit an acceptable behavior under the evaluation conditions, but some differences can be appreciated. The most significant dissimilarities are obtained from the choice of the Dirac or Gaussian function to model the occupied space along the impact. In general, the Gaussian model provides significantly better results than the Dirac model.

Regarding the difference between using line-based or angular-based methods, qualitative differences are visually noticeable, but these are not translated to large quantitative dissimilarities. Angular-based methods perform slightly better at farther distances, but the line-based method focused on higher model accuracy (weighted line-based method) obtains results alike.

With respect to the choice between (i) fast computation, (ii) balance between computation speed and model accuracy, and (iii) more accurate models; the results are consistent with the accuracy

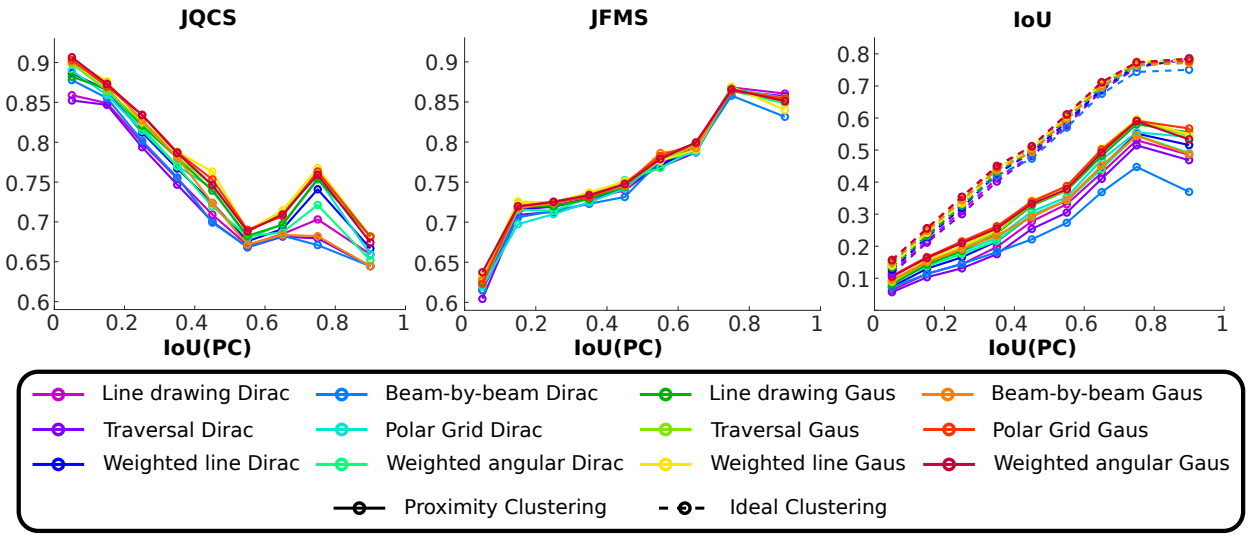


Figure 3.24: Evaluation results represented against the IoU(PC) metric.

sought by each strategy. Methods focused on more accurate models perform better in general. Nevertheless, they are closely followed by balanced methods. On the contrary, methods that make more simplifying assumptions, seeking faster computation, perform worse.

3.10. Summary and conclusions

As presented in Section 1.2.3, the grid-based module is intended to be the backbone of the proposed perception framework by (i) fusing all input data, (ii) feeding the subsequent perception modules and (iii) obtaining a reliable and general estimation of the surrounding environment, including all types of obstacles and free space.

To address the above-outlined objectives, this chapter has introduced a low-level grid map approach. Additionally, an evaluation framework for OGs from an object detection perspective has also been presented. This work led to two publications, one regarding the integration of cooperative perception into a grid-based framework [41], and one regarding the evaluation framework [60].

The proposed grid map strategy fuses information obtained from different sources and describes the environment using several features. Four input sources are considered: (i) LiDAR sensors, (ii) digital road maps, (iii) V2X communications, and (iv) object-level feedback from the object-based module. Six features are used to describe the environment: (i) occupancy state, (ii) dynamic state, (iii) the height of the obstacles and ground surface, (iv) semantic data related to the road, (v) semantic data related to obstacles, and (vi) semantic data categorizing the cells' occupancy state.

This chapter has focused on the first three input sources and four describing features. The occupancy and dynamic features (modeled from LiDAR and V2X data) are aimed to provide an accurate and reliable estimation of the obstacles and free areas in the scene. Therefore, they are

estimated over time using a recursive prediction/update process that interconnects an evidential OG—that models the occupancy state—with a PF—to model the dynamic state. Conversely, the height and road features are a simple rasterization of the input data (LiDAR and digital maps) with the intention of making their interpretation and integration easier for the subsequent tasks.

In the context of this grid map, contributions have been made in the domains of information modeling and fusion. With the intention of achieving a reliable occupancy estimation from LiDAR data, a novel approach to compute the measurement OG has been introduced. The proposed approach weights the influence of the laser beams over the cells traversed and leverages the classification and ground height estimation presented in Chapter 2 to better define the occupancy values set for each cell. Furthermore, if V2X data is available, it is integrated into the perception framework as an input to the grid-based module instead of fusing it at object-level. This is done by rasterizing the received objects into the grid in terms of occupancy and velocity, and fusing them with the LiDAR data. This allows to minimize the object-level well-known problem of data association, while also verifying the received information with the data gathered by the embedded sensors.

In addition to the grid-based module, an evaluation framework for OG approaches has been presented. The proposed framework evaluates the results of OG approaches from an object detection perspective covering its two main steps: object segmentation and feature estimation. Compared to other state-of-the-art methods, it introduces multiple metrics addressing different aspects and aims at enabling a repeatable application-specific analysis—as it relies on the ground truth data typically included in public datasets (road users' dynamic state and shape). This evaluation framework has allowed the comparison of various scan-rendering strategies and the obtention of a quantitative evaluation for the DOG developed, which is properly aligned with one of the main objectives of this thesis: road users detection and estimation.

In the next chapter, two deficits of the grid map presented in this chapter are addressed: (i) the decrease in the accuracy of velocity estimation in certain situations and (ii) the no-differentiation between the different types of obstacles present in the scene. While the dynamic estimation achieved is generally accurate, in certain situations, estimation delays or ghost movements can be found. Regarding object classification, the grid map presented in this chapter does not provide any information about obstacle classes, yet road users identification is fundamental for autonomous driving tasks.

Chapter 4

CLASSIFIED GRID MAP AND OBJECT-LEVEL TRACKING AND FEEDBACK

The proposed perception framework counts on four modules. Two of them, the grid-based and object-based modules have a strong interaction sharing information in both directions. This chapter presents the (i) object-based module, including the object clustering and road users multi-object tracking, (ii) the COG, which is an extension of the DOG to also include object classification, and (iii) the object-level feedback steps, that allow to obtain input data for the grid-based module concerning velocity and semantics.

The chapter is organized as follows: Section 4.1 introduces the motivation for adopting an object-level representation and overviews the tasks addressed in this chapter. Next, Section 4.2 provides a review of the state-of-the-art concerning the strategies that combine OGs and object-level tracking. Also, it introduces the techniques that allow the extension of DOGs to perform object classification and to improve the dynamic estimation. Afterward, the tasks addressed (highlighted in Figure 4.1) are detailed across five sections. Section 4.3 introduces the clustering techniques used to extract objects from the grid map and Section 4.4 the techniques used to describe them using box-based models. As can be noticed in the scheme, these extracted objects serve as the main input data for the other three tasks. Section 4.5 explains how to calculate the probabilities of the extracted objects belonging to different types of objects, and the extension of the DOG to also perform object classification (COG). Section 4.6 presents an approach for tracking at object-level the extracted objects that correspond to road users. Lastly, Section 4.7 introduces a novel feedback designed to estimate the dynamic state of certain objects and provide it as additional input data to the DOG. The experiments conducted to validate the proposed algorithms are presented in Section 4.8 and the overall conclusions of the chapter are drawn in Section 4.9.

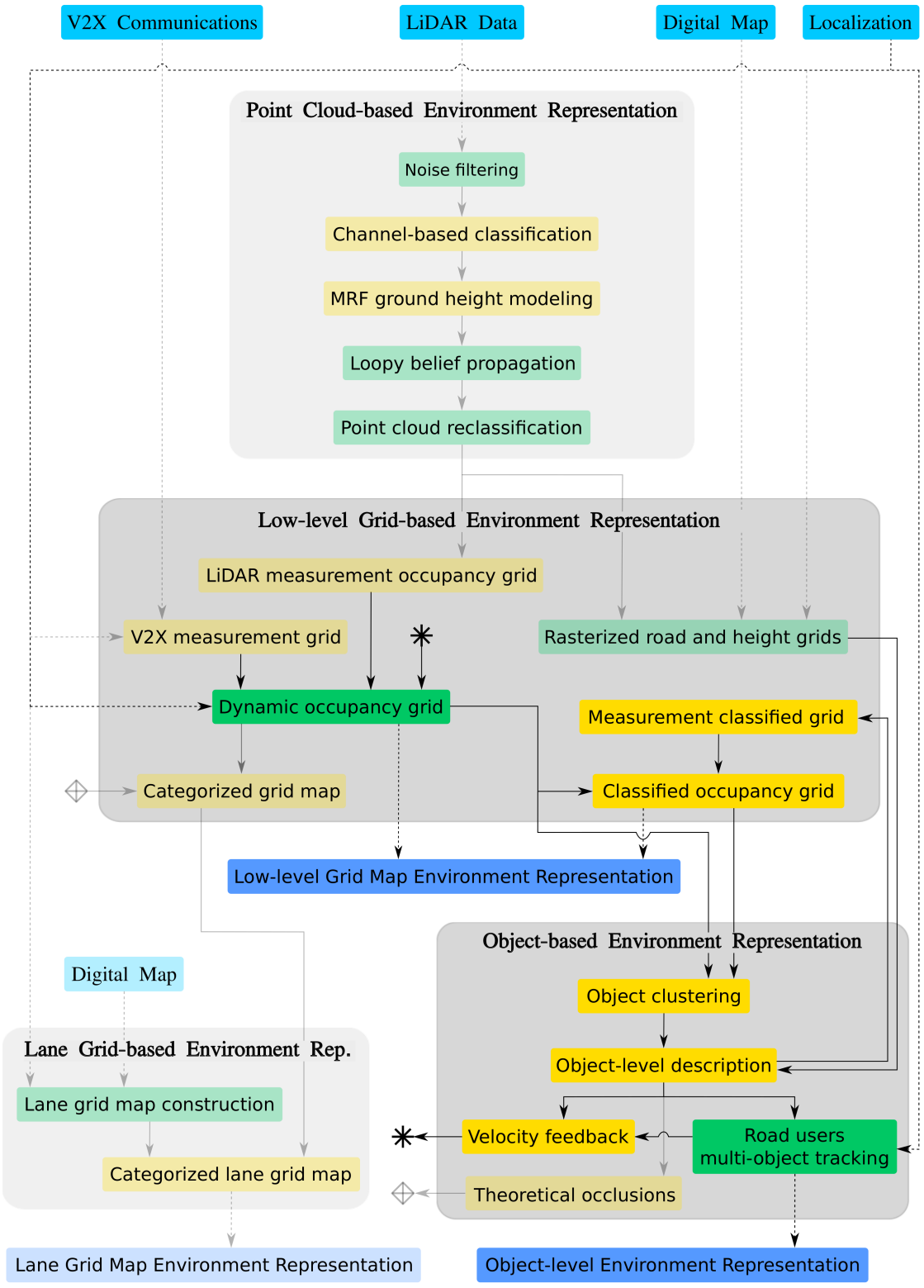


Figure 4.1: Perception framework scheme focused on the object-based module and the extension of the grid-based module. Modules constituting the framework are denoted in grey, input and output data in different shades of blue, commonly used tasks in green, and tasks including contributions in yellow. The tasks not addressed in this chapter are depicted in faded colors.

4.1. Introduction

This section presents the motivation for the development of an object-based module. Then, a brief introduction to the proposedobject-level tracking and feedback strategies is provided.

4.1.1. Object-level representation motivation

The grid-based module provides a comprehensive and accurate representation of the surrounding environment. It is able to address all obstacles despite their shape and type, while also estimating the free and unknown space. However, to achieve these benefits, the representation format used consists of a set of small-sized independent cells. This implies a low-level representation that may not be adequate for every module consuming the information estimated by the perception framework. Such modules prefer a representation of the road users in the scene in a more object-centric description, i.e., represented as (i) single objects and not as groups of multiple cells, (ii) individual entities constant over time, and (iii) described by specific known state and shape definitions.

When aiming for an object-centric description, object-level tracking approaches that are specifically focused on identifying and tracking certain types of known objects exhibit outstanding performance. On the one hand, addressing specific objects as individual entities enables the estimation of the number of objects and their identification throughout the consecutive iterations. This is particularly useful for tasks that analyze object behavior, such as motion prediction and decision-making. On the other hand, addressing known object types enables model assumptions. If the selected models are appropriate, several advantages can be obtained, such as modeling the full shape of partially detected objects, delimiting their possible displacements, inferring hidden variables, or classifying them into object classes.

The advantages provided by model assumptions can be used not only during the object-level representation steps but also to enhance the grid-based module's estimation. For instance, motion and shape assumptions can be used to infer information about obstacles' velocity and class, thus, obtaining data that LiDAR sensors do not naturally provide.

As a result, the object-based module is suitable to fulfill certain requirements specified in Sections 1.1 and 1.2, such as (i) the improved identification and estimation of road users, (ii) their generalized and commonly used representation, and (iii) the generation of input information for the grid-based module.

4.1.2. Overview of the Classified Occupancy Grid, object-level tracking and feedback

As introduced in the previous section, this module serves a dual purpose: the object-level representation of the road users in the scene and the calculation of input data that allows to enhance the estimation of the grid-based module.

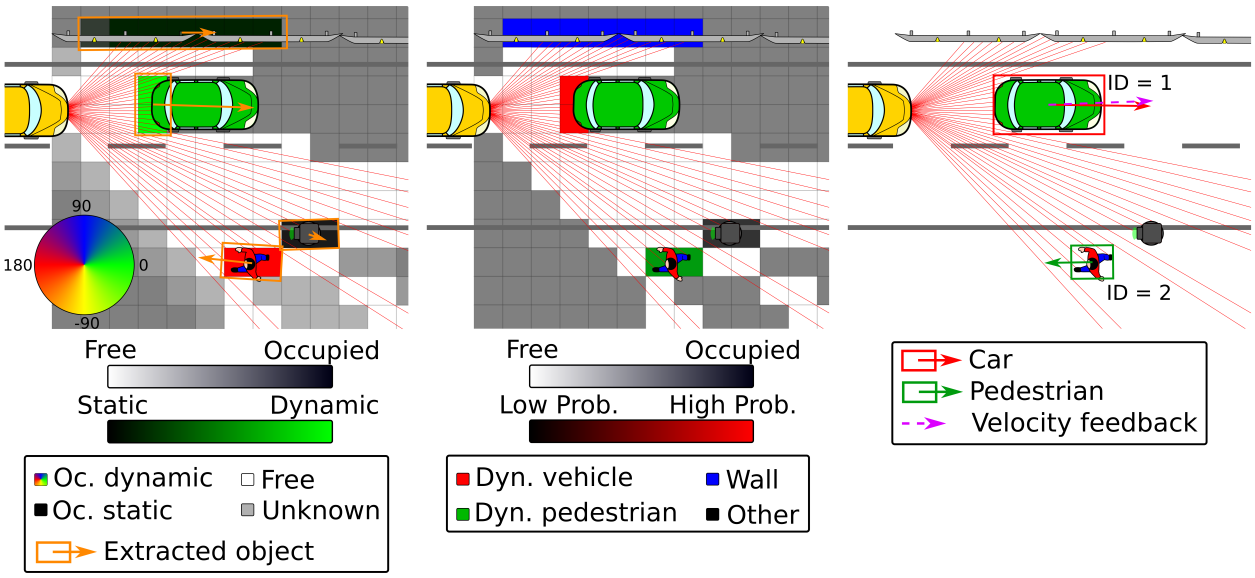
For the road users object-level tracking, a strategy consisting of three steps is selected: (i) de-

tection, (ii) matching and (iii) tracking. Therefore, this strategy first extracts objects from the grid map representation and describes them as accurately as possible using the selected object-level representation, i.e., dynamic state, box-based shape and class. Then, these extracted objects are matched one-to-one with existing object tracks by evaluating their similarity. Once the matching is solved, object tracks are updated using the information of their matched extracted object, thus, obtaining a filtered estimation of the road users over time.

The feedback information is calculated in a similar way, extracting objects from the grid and describing them at object-level. However, instead of using this data to feed the object-level tracking, it is used to feed the grid-based module. Velocity data is inferred for large static off-road obstacles, which tend to present false dynamics, and for dynamic vehicles, whose velocity is computed based on the displacement between consecutive iterations. Classification data is calculated by evaluating certain object-level features with respect to the possible object classes.

The velocity data can be directly integrated into the DOG through the velocity update and particle state initialization steps. On the contrary, the grid-based module introduced in Chapter 3 does not include any method to integrate and estimate semantic data. This chapter presents a novel method that extends the already developed DOG in order to also perform object classification, (COG). As for the dynamic estimation, object classification over time is carried out by the particles. Each particle is assigned a set of probabilities corresponding to the possible object classes that are updated at cell-level using the input classification data and a naive Bayes classifier.

Figure 4.2 provides an illustrative example of the object extraction from the grid and the desired cell-level object classification, object-level tracking and velocity feedback calculation.



a) Input data: extracted objects.

b) Classified Occupancy Grid.

c) Object-level tracking and velocity feedback.

Figure 4.2: Illustrative examples of proposed COG, object-level tracking, and feedback.

4.2. Related works

As introduced in the previous section, this chapter focuses on two tasks: (i) the development of an object-level tracking that provides the desired object-level representation of road users, and (ii) the enhancement of the DOG approach by extending its estimation to object classification and fusing new input data gathered from the object-level module. The state-of-the-art related to these topics is reviewed below.

4.2.1. Combination of Occupancy Grid and object-level tracking strategies

The task of estimating the road users in the scene constitutes a multi-object detection and tracking problem with a varying number of targets over time. This is a widely studied problem but without a well-established solution. Some examples including extensive analysis and comparisons of different techniques are [3, 10, 36, 43, 44, 47, 161].

The approaches combining grid-based and object-based modules can be divided into two groups: (i) those that compute both modules as independent perception frameworks and then fuse their estimations, and (ii) those that interconnect both modules seeking to enhance the estimation of one or both modules. Figure 4.3 visualizes some examples of the works reviewed in the following.

4.2.1.1. Works with independent grid-based and object-based modules

The strategies that compute the grid maps and object-level tracking as independent modules usually have as main goal the development of a perception framework with redundant estimations that guarantees security or a modular perception framework that facilitates the exchangeability of algorithms and sensors.

In [39], both modules are developed separately and then combined in order to reduce false positives and obtain complementary environment estimations. On the one hand, at object-level, dynamic obstacles are identified in radar and camera measurements and tracked over time using a Labeled Multi-Bernoulli filter (LMB) [112]. On the other hand, a DOG is used to estimate the surrounding environment using LiDAR sensors. Dynamic object hypotheses are extracted from the grid by clustering occupied cells using a DBSCAN [34] based on distance and velocity criteria and the Mahalanobis distance towards zero velocity. Afterward, given the two sets of dynamic objects hypotheses, a high-level fusion module combines both estimations into a new set of meta objects. This combination module matches the estimations of both modules using the Hungarian method [67] regarding Euclidian distance and validates them by computing different confidence criteria, e.g. by assessing the position and orientation of the vehicles with respect to the lanes.

A similar approach combining the estimation of a DOG based on particles and the estimation of an object-level tracking calculated with LMB filter is presented in [113], see Figure 4.3a. In this case, in order to avoid clustering errors, the cells corresponding to the vehicles are detected using the object hypotheses generated by the LMB filter and expanding the obtained clusters to the immediate

vicinity of dynamic cells.

In the works [68, 94] an environmental modeling based on a static OG, a multi-object tracking and digital road maps, is proposed, Figure 4.3b. In these cases, the different modules are computed separately in order to achieve modularity and easy adaptability. An additional combination module is used to evaluate the consistency between them and create the outcome environmental model. This environmental model is constituted by two complementary environment representations: (i) the set of dynamic obstacles, calculated through the object-level tracking—again the LMB filter is selected in order to better deal with ambiguous associations—and (ii) the boundaries of the drivable space, which are identified as the occupied grid cells inside of the road but not associated with a

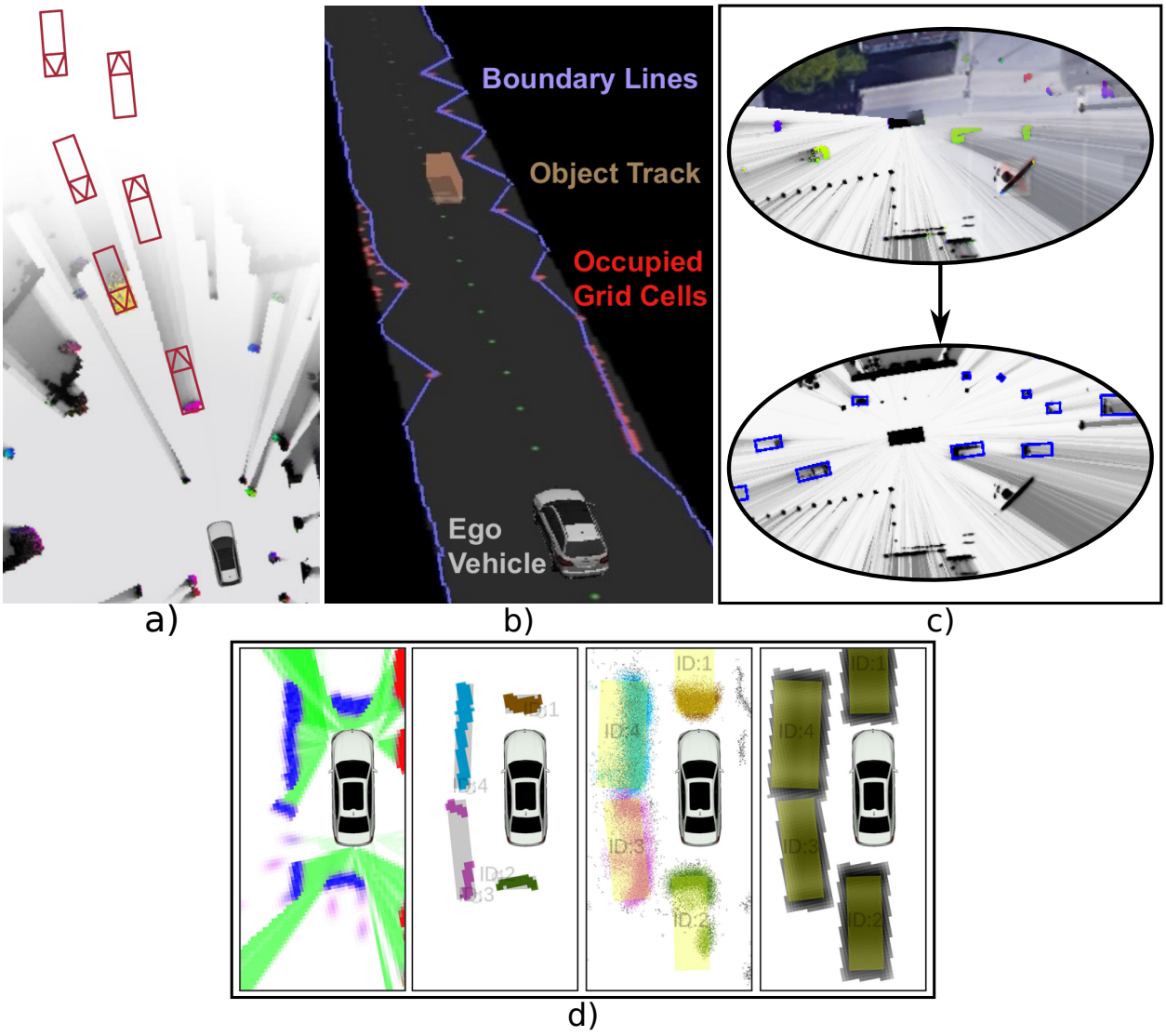


Figure 4.3: Example of four different approaches combining OGs and object-level tracking. a) Exploitation of distance-based strengths of each approach [113]. b) Environment model resultant from the combination [68]. c) Object detection in DOGs using neural networks [50]. d) Object detection in DOGs leveraging tracks' IDs and DOG's inner particles.

dynamic object track—this is done following a probabilistic approach that evaluates the cells with respect to their estimated state and existence probability of object tracks.

4.2.1.2. Works interconnecting grid-based and object-based modules

The family of methods that interconnects both modules, grid-based and object-based, attempts to enhance the estimation of one or both modules by exploiting the outputs of the other. In these works, usually, the grid-based module is computed first in order to obtain an estimation of the complete surrounding environment, and then the object-level tracking is calculated by taking the information of occupied cells as input and focusing only dynamic obstacles. However, in some cases, the information of the object-level tracking is back-channeled to the grid seeking to enhance the overall estimation.

In [158] a static OG is computed using LiDAR data in order to map the surrounding environment and address localization. This grid map is also used to detect moving objects among the sensor measurements which, in turn, are used by an object-level tracking to track dynamic obstacles. The idea is to identify laser measurements corresponding to dynamic obstacles by (i) analyzing inconsistencies between free space and occupied space, and (ii) developing an additional grid that denotes the areas with frequent dynamic obstacles. Moving objects hypotheses are then generated using a distance-based clustering algorithm and estimated over time using an object-level tracking. Since moving objects may correspond to multiple types of objects, e.g. vehicles and pedestrians, these are tracked over time using an Interacting Multiple Model [79] Kalman Filter (KF) [63]. Additionally, in order to tackle complex association situations it is combined with a Multi Hypothesis Tracking method [150].

In a similar approach, [62] employs an OG based on the DST to map the static environment and identify dynamic evidence. Then, taking the dynamic cells as input, an object-level tracking focused on dynamic obstacles is developed. In this case, the tracking over time is accomplished using a PF aiming to estimate the state of an object based on multiple hypotheses. The association problem is addressed with the Joint Probability Data Association method [7] so association thresholds of simpler methods are replaced by an association likelihood.

Bouzouraa and Schueler [11, 133] explored the combination of an OG and an object-level tracking in order to obtain advantages at both stages. On the one hand, the OG is used to identify sensor measurements corresponding to dynamic obstacles, so that object-level tracking receives an already filtered input. On the other hand, the object-level estimation is employed to predict the position of the cells associated with dynamic obstacles and, therefore, prevent the occupancy trail that dynamic objects produce in classic OG approaches. In both cases, the object-based module is implemented using a multi-object KF to track objects' state, assuming a constant acceleration motion model and an oriented bounding box shape model, and the Global Nearest Neighbor approach [65], based on distance, to solve the association between tracks and extracted objects. Additionally, in order to better track the obstacles that are only partially perceived and avoid wrong velocities resulting from

moving center points, [133] introduces the tracking by reference point, which corresponds to the best observable point, being the set of possibilities any corner or middle side point of the bounding box.

These last works expose that grid-based and object-based modules can benefit from each other, particularly to avoid the inconsistencies generated by dynamic obstacles in static OGs. However, the growing computational resources motivated the development of new OG strategies—such as popular DOGs based on PFs [92, 118, 144]—that can perform standalone in dynamic environments while estimating continuous velocity values. Therefore, the development of algorithms identifying dynamic cells by analyzing inconsistent measurements or correcting the displacement of cells corresponding to dynamic obstacles is no longer required. Moreover, the inclusion of the particles or the dynamic state of the cells introduces novel tools to tackle complex object-level steps such as the association problem.

Therefore, similarly to the works introduced above, in [41] (which leveraged some of the developments from this thesis) a DOG is used as input data for the object-based module. Objects are extracted from the grid using a density-based clustering algorithm and tracked over time using a multi-object KF approach. This work does not rely on any further information than the cell's features and is able to obtain acceptable object-level tracking results. However, in complex situations, using such a simple single-stage object extraction method can lead to wrong object estimations and, hence, poor tracking performance.

Seeking to address challenging urban scenarios, Steyer et al. proposed a number of works [141, 142, 143] in which the grid-based and object-based modules are closely related in order to improve the overall performance. The grid-based module is based on a PF and the DST to further distinguish between static and dynamic occupied space. The object-based module focuses on dynamic road users and is calculated using a bounding box-based Unscented Kalman Filter. Different interesting modifications and improvements are proposed throughout these works.

In [143], new tracks are extracted from the grid using a DBSCAN algorithm that takes into account the distance between cells, the velocity difference, and the in-between free space. However, the association between objects and existing object tracks is directly solved at cell-level, without performing clustering steps. In order to achieve this, the existing tracks are projected into the grid and the probability that a cell is associated with it is computed by assessing distance and velocity criteria. Moreover, the free space information is used to validate if the measured length and width correspond to the actual dimensions of the obstacle or just to a lower bound. Similarly, this criterion is used to obtain the best reference point in order to perform robust object tracking. However, [142] explains that the association method of [143] is prone to error in situations with multiple tracks close to each other. Therefore, a new association method based on labeling the particles of the DOG with the existing tracks' ID is presented (Figure 4.3d displays this procedure). Furthermore, the object tracks of low-speed dynamic obstacles are used to avoid the convergence of the cells from dynamic to static. A similar approach is proposed by [25] which also proposes to include objects

ID into the particles' state seeking to track objects' individual identity. However, no additional object-level tracking is performed, only a clustering step based on a Connected Component labeling is employed. Finally, [141] consolidates the strategies of [143, 142] into a more comprehensive method and introduces certain new aspects such as the use of radar measurements to update object tracks, incorporating free space information in the estimation of the oriented bounding boxes, and performing an object classification to better estimate their shape.

Approaches based on neural networks, such as [32, 50], yield good results in cells' dynamic-static segmentation and object extraction due to their ability to take context into account, see Figure 4.3c. However, as it is well-known, the performance of these strategies depends on the existence of extensive training datasets.

4.2.2. Enhanced Dynamic Occupancy Grids

As already stated in Section 3.10, the grid map environment representation introduced in Chapter 3 presents two weaknesses: (i) the decrease of the dynamic estimation accuracy in certain situations, and (ii) the lack of an object classification step. In the following, the works that motivated the development of the object-level feedback and the COG in order to address these issues are presented. Figure 4.4 already illustrates the results of some of these approaches.

4.2.2.1. Dynamic Occupancy Grid velocity estimation enhancement

The DOG presented in Section 3.6, is able to obtain accurate velocity estimations using only LiDAR data in most of the cases, but its accuracy diminishes in certain situations. This was already highlighted in the original work [91, 92, 96], where it is shown how the incorporation of radar measurements provides a faster convergence over the real velocity of the estimated objects, more homogeneity among the cells modeling the same obstacle, and reduces the occurrence of false positive movements in static areas. Figure 4.4a provides an example of this enhancement. Indeed, the algorithm is specifically designed to be able of integrating velocity measurements through the update and initialization steps of the PF (see Sections 3.6.3.3 and 3.6.3.5).

In the existing literature, several similar DOG-based approaches using different resources to improve the dynamic estimation can be found. For instance, [140] and [148] also include radar data with this intention. In [140], camera information is additionally used to estimate the orientation of the objects and then evaluate and initialize particles accordingly. Furthermore, in [148], seeking to address the convergence of static obstacles, particles with zero velocity can be sampled proportionally to prior static evidence. These last two approaches for enhancing the dynamic estimation of the DOG demonstrate that velocity input information can be obtained not only from raw sensor measurements but also through alternative processes. In this regard, for example, [171] approximates the velocity of objects by comparing their position at successive time steps using the cross-correlation coefficient metric. This possibility is particularly relevant to the work presented in this manuscript as it inspires the velocity feedback introduced in this chapter.

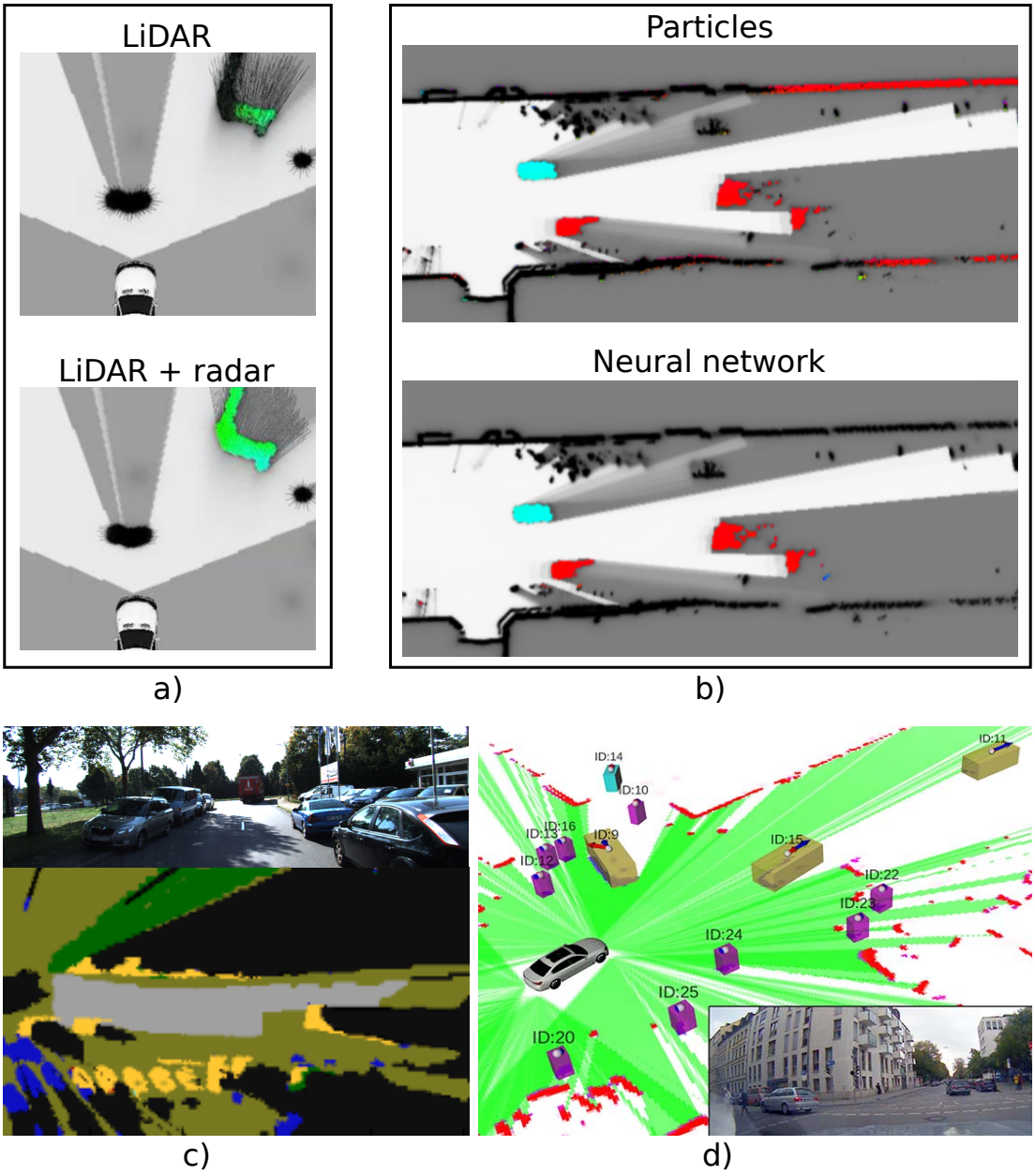


Figure 4.4: Example of four different approaches enhancing the results of DOGs. a) Enhanced velocity estimation fusing radar data [90]. b) Enhanced velocity estimation using neural networks [129]. c) Semantic occupancy grid using RGB images [33]. d) Object-level classification [141].

Finally, it is worth mentioning that this problem has also been approached using neural networks. Schreiber et al. [128, 129] proposed to use recurrent neural networks to predict DOG taking occupancy measurement grids as input. This approach yields successful results obtaining better accuracy in the velocity estimation and minimizing the issue of independent cells, see Figure 4.4b. However, large amounts of data are typically needed for appropriate training and, in most cases, the resulting neural network performance degrades if the sensor configuration is changed.

4.2.2.2. Grid map object classification

DOGs are able to classify cells into *dynamic* or *static*, either using a single frame analysis, as presented in Section 3.6.3.4, or estimating this classification over time, for example [101, 144, 148]. This classification is useful for the object-level tracking since it is commonly assumed that every dynamic obstacle is a potential risk and therefore should be further tracked at object-level. However, in order to obtain a more comprehensive description of the driving scene or take advantage of specific model assumptions, a further classification identifying the relevant obstacles is required.

The classification of the space into multiple classes in OG frameworks is usually known as Semantic Occupancy Grids (SOG) and is commonly accomplished through the use of camera sensors or neural networks. For example, in [33], a DOG, computed with LiDAR data, and a semantic segmentation, calculated using RGB images, are fused into a SOG using a deep neural network. This approach is able to label the cells into multiple classes, e.g. *cars*, *pedestrians*, *road*, *vegetation*, etc. see Figure 4.4c. Similarly, [156] also combines a DOG with a semantic grid computed using stereo vision sensors and neural networks [127]. However, in this case, the semantic estimation is additionally filtered over time by attaching the semantic labels to sets of particles. [130] proposes a DOG approach that employs a recurrent neural network instead of particles. This modification allows to obtain a representation regarding occupancy, velocity and semantic information, including classes such as *vehicles*, *pedestrians*, *two-wheelers*, and *drivable area*. As mentioned above, these types of approaches expose promising results.

In object-level approaches, object classification is usually accomplished in a similar way, e.g. [15, 106, 137]. However, some approaches take advantage of the object-level features and tracking resources to estimate the tracked objects' classes over time. [41] analyzed the historical data of the tracked objects with respect to size and velocity features in order to identify vehicles from other dynamic objects. [103, 104] combined a Gaussian Mixture Model, based on features such as centroid, speed and size, and a Bayesian classifier in order to classify the detected objects into *vehicles* and *pedestrians*. A further classification into the classes *car*, *pedestrian*, *truck*, *cyclist*, *motorcycle* and *other*, is performed in [141] using a naive Bayes classifier also based on the geometry and velocity of the tracked objects, an example of its results is given Figure 4.4d.

4.2.3. Progress beyond the state-of-the-art

As can be noticed from the works reviewed above, the combination of both modules, grid-based and object-based, is a common and advantageous approach. The use of a grid map as low-level environment representation from which an object-level tracking is computed is an already well-established strategy. On the contrary, although different advantages have been exposed in some works, back-channeling object-level information to the grid is yet underexploited. Also, different methods for enhancing the estimation of DOGs regarding velocity and classification can be found in the literature. However, most of them rely on additional sensors or neural networks.

Taking this into account and seeking to address the weaknesses of the grid-based module presented in the previous chapter, the following strategies are developed:

- An object-level tracking algorithm where objects are (i) extracted from the grid map based on similarities between neighboring cells and (ii) tracked over time using a multi-object KF approach.
- Inspired by other works that extend the particles' state including object-level data, e.g. [25, 142], and by the object classification method based on object-level features presented in [141], a novel approach for object classification at cell level is proposed. This method analyzes the features of the extracted objects to infer their corresponding object class and takes advantage of the PF to perform the classification over time.
- In view of the works that use alternative methods to enhance the dynamic estimation of the filter rather than, or in addition to, sensor raw velocity measurements, e.g. [140, 148], a new step intended to infer the velocity of the surrounding objects at each time step is proposed. This velocity is then included in the DOG through the particles' update and initialization equations.

4.3. Object clustering

The work presented in this thesis includes four tasks that require object clustering from the grid map environment representation: (i) the evaluation framework (Section 3.8), (ii) the COG (Section 4.5), (iii) the CG (Section 5.5) and (iv) the road users object-level tracking (Section 4.6).

The first three tasks can obtain successful performance using a clustering method based on similarity between neighboring cells. Conversely, the object-level tracking requires a more elaborated clustering strategy. In this way, the same clustering method is used for all of them, but for the object-level tracking, some modifications are introduced.

4.3.1. Basic clustering approach

The clustering method used in this thesis is primarily based on the Connected Components algorithm [117]. This algorithm evaluates the similarity between clusterable cells inside a local neighborhood. If two neighboring cells are similar enough, they are considered to belong to the same object. The process consists of three main steps:

- Only the cells with sufficient occupied evidence are considered potential obstacles. Therefore, the set of cells $\boldsymbol{\mathcal{C}}^{obstacle}$ taken into account for clustering is defined by:

$$\boldsymbol{\mathcal{C}}^{obstacle} = \left\{ c \in \mathcal{G} \mid m(O^c) > \mathcal{T}_o^{cl} \right\} \tag{4.1}$$

  where $m(O^c)$ is the occupied mass of a cell c in the grid map $\mathcal{G}$ and $\mathcal{T}_o^{cl}$ is an occupancy

threshold.

- The similarity between two cells is measured by evaluating the proximity and velocity difference. Thus, two cells c^i and c^j are considered to correspond to the same object if the similarity criterion $s^{cl}(c^i, c^j)$ is met:

$$s^{cl}(c^i, c^j) = \left(c^i, c^j \in \boldsymbol{\mathcal{C}}^{obstacle}\right) \wedge \left(s^{cl}_{cheb}(c^i, c^j) < \mathcal{T}^{cl}_{cheb}\right) \wedge \left(s^{cl}_{mod}(c^i, c^j) < \mathcal{T}^{cl}_{v}\right) \tag{4.2}$$

  where $s^{cl}_{cheb}(c^i, c^j)$ denotes the Chebysev distance between both cells, $s^{cl}_{mod}(c^i, c^j)$ the absolute velocity module difference and $\mathcal{T}^{cl}_{d}$ and $\mathcal{T}^{cl}_{v}$ are distance and velocity thresholds.

- In order to filter noise, it is common to apply a minimum size threshold. Therefore, once all cells have been clusterized, only those clusters with at least $n^{cl,min}_{c}$ are accepted as valid objects.

For the tasks of (i) evaluation framework, (ii) COG and (iii) CG, this clustering approach is applied using a conservative parametrization: $\mathcal{T}^{cl}_{d} = 1$ and $n^{cl,min}_{c} = 1$. A minimum cluster size of one grid cell serves the purpose of ensuring the detection of every possible obstacle, especially considering that pedestrians can be modeled as small groups of cells due to far distances and sparse LiDAR data. A distance threshold defined as the distance to the adjacent cell minimizes object merging at its most and leads to compact clusters.

4.3.2. Modifications of the clustering approach for the object-level tracking

OGs are prone to object splitting, especially in cases of large obstacles such as vehicles. While the COG and the CG do not rely on grouping all cells corresponding to the same object in a single cluster, the performance of the object-level tracking algorithm is strongly affected by this fact. This is because the grid-based methods only require the clusters to be sufficiently representative of the actual object to classify and categorize cells. In contrast, for object-level tracking, clusters serve as hypotheses of individual objects.

Therefore, for the object-level tracking, the clustering approach presented in the previous section is extended by considering (i) additional features for the similarity evaluation, (ii) a wise distance threshold definition and (iii) the predictions of the object-level tracking.

4.3.2.1. Similarity calculation enhancement and wise distance threshold definition

The similarity calculation between two cells is extended to also include the cells' object classification and the free space between both cells. Therefore, (4.2) is modified as follows:

$$\begin{aligned} s^{cl}(c^i, c^j) = &\left(c^i, c^j \in \boldsymbol{\mathcal{C}}^{obstacle}\right) \wedge \left(s^{cl}_{cheb}(c^i, c^j) < \mathcal{T}^{cl}_{cheb}\right) \wedge \left(s^{cl}_{mod}(c^i, c^j) < \mathcal{T}^{cl}_{v}\right) \wedge \\ &\left(s^{cl}_{free}(c^i, c^j) < \mathcal{T}^{cl}_{free}\right) \wedge s^{cl}_{obj}(c^i, c^j) \end{aligned} \tag{4.3}$$

where $s^{cl}_{free}(c^i, c^j)$ and $s^{cl}_{obj}(c^i, c^j)$ are similarity functions that consider the free space in between them and the cells' object classification, respectively. If free space is modeled between two cells, they probably correspond to different obstacles, despite the other similarity criteria. This is measured by:

$$s^{cl}_{free}(c^i, c^j) = \sum_{c^b \in \mathcal{B}(c^i, c^j)} m(F^{c^b}) \tag{4.4}$$

being $\mathcal{B}(c^i, c^j)$ the set of cells in between cells c^i and c^j calculated using Bresenham's algorithm [12]. The object classification similarity $s^{cl}_{obj}(c^i, c^j)$ is computed by restricting the possible association between cells corresponding to different classes. Similar cells must share the same object label, except for the class *other* which is considered as unknown and, thus, can be grouped with any other class.

Finally, to define the distance threshold in a safer and target-oriented manner, its value is established depending on the class of the cells. *dyn. pedestrian* and *wall* cells explore their vicinity with a distance $\mathcal{T}^{cl}_d = 1$, while cells belonging to vehicles, which are prone to splitting, explore their vicinity with a distance $\mathcal{T}^{cl}_d > 1$.

4.3.2.2. Split cluster merging based on object-level tracking predictions.

The clustering method introduced above only takes into account the information of the current iteration. However, the object-level tracking has information about the objects in the previous iteration, which can be used to locate the known objects in the current instant. Since the clustering of vehicles is prone to split errors, the knowledge of the object-level tracking is used to merge clusters corresponding to a single vehicle.

Similarly to [143], the probability $s(c, \tau_v)$ that a cell c belongs to a tracked vehicle τ_v is calculated based on: (i) its location with respect to the tracked vehicle $s_{pos}(c, \tau_v)$ and (ii) its velocity difference $s_{vel}(c, \tau_v)$ as follows:

$$s(c, \tau_v) = s_{pos}(c, \tau_v) \cdot (\alpha^{\tau_v}_{vel} \cdot s_{vel}(c, \tau_v) + 1 - \alpha^{\tau_v}_{vel}) \tag{4.5}$$

where $\alpha^{\tau_v}_{vel}$ is a value that weights the impact of the velocity-based association. If the cell can be associated with more than one track, the association with the highest value is selected.

The location-based association is calculated by taking into account the position and shape of the track:

$$\begin{aligned} s_{pos}(c, \tau_v) = &\min\left(1,\ \max\left(0,\ \frac{0.5 l^{\tau_v}_{\mathrm{x}} + \mathcal{T}^{cl,\tau_v}_{pos} - |x^{c,\tau_v}_{\mathrm{x}}|}{\mathcal{T}^{cl,\tau_v}_{pos}}\right)\right) \cdot \\ &\min\left(1,\ \max\left(0,\ \frac{0.5 l^{\tau_v}_{\mathrm{y}} + \mathcal{T}^{cl,\tau_v}_{pos} - |x^{c,\tau_v}_{\mathrm{y}}|}{\mathcal{T}^{cl,\tau_v}_{pos}}\right)\right) \end{aligned} \tag{4.6}$$

where l^{τ_v} denotes the size of the track, $\mathcal{T}^{cl,\tau_v}_{pos}$ is a distance value to enlarge the track's box size and

$\boldsymbol{x}^{c,\tau_v}$ denotes the position of the cell referred to a reference system located at the track's position calculated by:

$$\begin{bmatrix} x_{\mathrm{x}}^{c,\tau_v} \\ x_{\mathrm{y}}^{c,\tau_v} \end{bmatrix} = \begin{bmatrix} \cos(\theta^{\tau_v}) & \sin(\theta^{\tau_v}) \\ -\sin(\theta^{\tau_v}) & \cos(\theta^{\tau_v}) \end{bmatrix} \left(\begin{bmatrix} x_{\mathrm{x}}^{c} \\ x_{\mathrm{y}}^{c} \end{bmatrix} - \begin{bmatrix} x_{\mathrm{x}}^{\tau_v} \\ x_{\mathrm{y}}^{\tau_v} \end{bmatrix} \right) \tag{4.7}$$

Figure 4.5 provides an illustrative example of this association calculation.

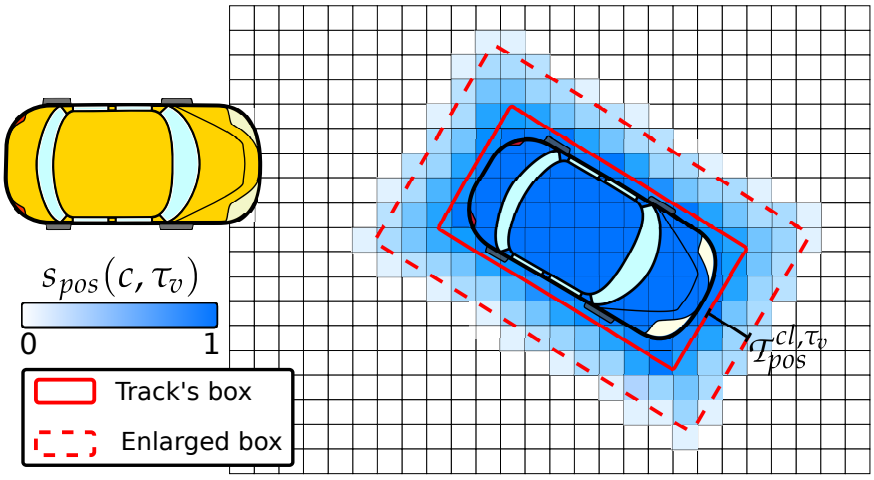


Figure 4.5: Illustrative example of the location-based association calculation.

The velocity-based association is computed taking into account the velocity difference in both components:

$$s_{vel}(c, \tau_v) = 0.5 \left(\max\left(0,\ 1 - \frac{|\mu_{v_{\mathrm{x}}}^{c} - v_{\mathrm{x}}^{\tau_v}|}{\mathcal{T}_{vel}^{cl,\tau_v}}\right) + \max\left(0,\ 1 - \frac{|\mu_{v_{\mathrm{y}}}^{c} - v_{\mathrm{y}}^{\tau_v}|}{\mathcal{T}_{vel}^{cl,\tau_v}}\right) \right) \tag{4.8}$$

where $\boldsymbol{\mu}_v^c$ denotes the mean velocity of the cell, $\boldsymbol{v}^{\tau}$ the velocity of the track and $\mathcal{T}_{vel}^{cl,\tau_v}$ the maximum velocity difference allowed.

Each cell fulfilling $s(c, \tau_v) > \mathcal{T}_{\tau_v}^{cl}$ is associated with the track τ_v and the probability for each cluster cl of belonging to the tracked vehicle τ_v is calculated based on the percentage of cells associated with it:

$$\varphi_{cl,\tau} = \frac{n_c^{\tau_v,cl}}{n_c^{cl}} \tag{4.9}$$

being $n_c^{\tau_v,cl}$ the number of cells associated with the track τ_v and n_c^{cl} the total number of cells of the cluster. A cluster is considered to belong to a tracked vehicle if its percentage of associated cells exceeds the threshold $\mathcal{T}_{merge}^{cl}$. If more than one cluster is associated to the same tracked vehicle, they are merged into a single cluster.

Figure 4.6 shows an example comparing (i) the basic clustering approach and (ii) the clustering employed for the object-level tracking after merging split objects. It can be clearly noticed that the first tends to segment objects into small groups (a decision intended to obtain compact clusters

and correct pedestrian segmentation); while the second one is able to correctly group all the cells corresponding to the same object, a behavior desired for general road users segmentation.

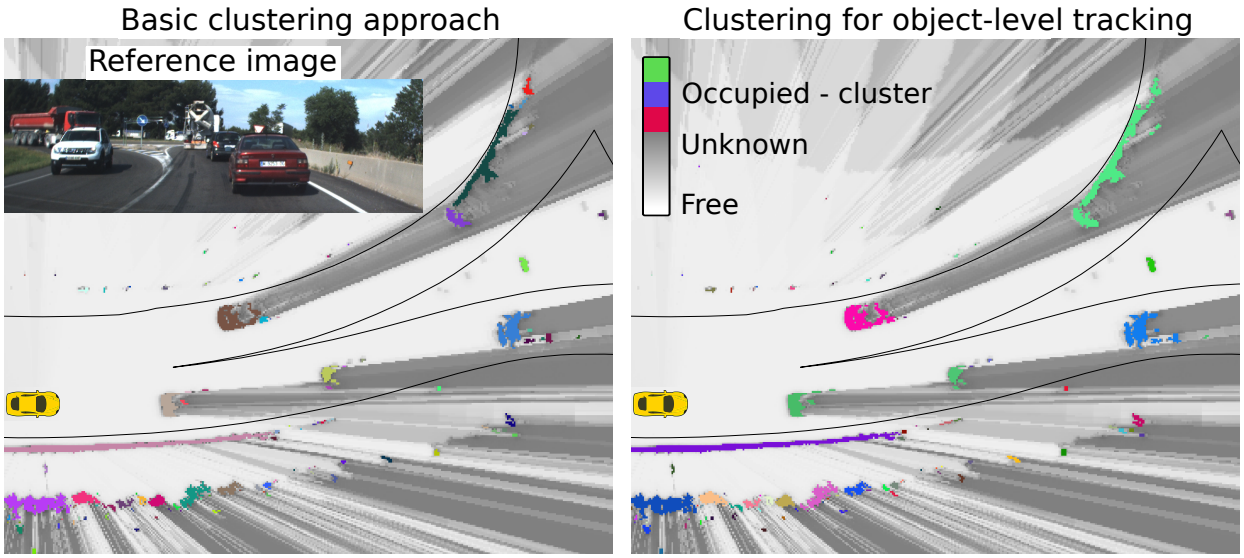


Figure 4.6: Example of the clustering approach proposed. **Left:** basic clustering approach. **Right:** enhanced clustering approach for object-level tracking.

4.4. Oriented bounding box estimation

At object-level, the representation selected for the objects is an oriented bounding box. Therefore, for each object extracted from the grid map, a bounding box representation is calculated. In the same way that for the clustering task, the bounding box calculation varies depending on the step: (i) COG, (ii) velocity feedback, and (iii) road users object-level tracking. Moreover, for some of them, more than one box is calculated, e.g. for the object-level tracking two bounding boxes are calculated, one based on geometry features and one based on the velocity vector. Four strategies for bounding box calculation are used: (i) velocity-based, (ii) geometry-based, (iii) velocity and geometry-based, and (iv) maximum length-based. Figure 4.7 shows an illustrative example of these boxes.

Each bounding box is defined as:

$$\boldsymbol{b} = \left[x_{\mathrm{x}}, x_{\mathrm{y}}, \theta, l_{\mathrm{x}}, l_{\mathrm{y}}\right] \tag{4.10}$$

being x_{x}, x_{y} the 2D-position of the box's center, θ the box orientation and l_{x}, l_{y} its length and width, respectively. All these variables are assumed as Gaussian.

In the following, the different strategies used in this thesis for calculating bounding boxes are presented.

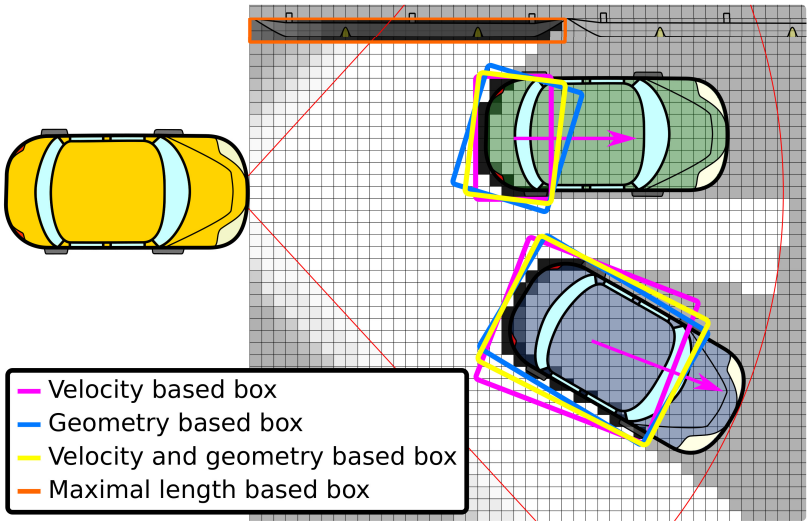


Figure 4.7: Illustrative example of oriented minimum bounding boxes.

4.4.1. Bounding box based on velocity

The velocity-based bounding box $\boldsymbol{b}^v$ is directly oriented as the velocity vector of the cluster:

$$\theta^{b^v,cl} = \arctan\left(\frac{v_{\mathrm{y}}^{cl}}{v_{\mathrm{x}}^{cl}}\right) \tag{4.11}$$

where the superindex cl denotes the cluster for which the bounding box is calculated.

The position and size are obtained by calculating the minimum bounding box that gathers all the cells corresponding to the cluster given θ^{b^v}. Since the reliability of the velocity orientation decreases with the speed value, the standard deviations of the variables defining $\boldsymbol{b}^v$ are adjusted accordingly. The calculation of the standard deviation of the orientation variable θ^{b^v} given the velocity criteria is shown in (4.12). The standard deviation for the rest of the variables is computed equivalently.

$$\sigma_\theta^{b^v} = \begin{cases} \sigma_\theta^{b^v,min}, & \text{if } ||\boldsymbol{v}|| \geq \mathcal{T}_v^{rel} \\ \sigma_\theta^{b^v,max} - \frac{\sigma_\theta^{b^v,max} - \sigma_\theta^{b^v,min}}{\mathcal{T}_v^{rel}} \cdot ||\boldsymbol{v}||, & \text{otherwise} \end{cases} \tag{4.12}$$

where $\mathcal{T}_v^{rel}$ defines the velocity threshold above which the velocity orientation can be considered reliable and $\sigma_\theta^{b^v,min}$ and $\sigma_\theta^{b^v,max}$ are minimum and maximum standard deviations for the variable θ, experimentally adjusted.

4.4.2. Bounding box based on geometry

The geometry-based bounding box $\boldsymbol{b}^g$ is computed using a variation of the approach presented in [173]. For each angle in the set $\theta = \{0, 1, 2, ..., 180°\}$ an oriented minimum bounding box gathering occupied cells and a fitting score are calculated.

The fitting score is computed as:

$$s_{fit} = s_{free} \cdot s_{contour} \tag{4.13}$$

being s_{free} a score that measures the amount of free space modeled inside the box:

$$s_{free} = \frac{1}{n_c^{cl}} \sum_{c=1}^{n_c^{cl}} m(F^c) \tag{4.14}$$

and $s_{contour}$ a score that measures how well the box fits the occupied space. It is calculated as the weighted variance of the distance between the cells and the sides of the box. Distances are additionally weighted w^d accordingly to the cell's occupied mass and whether they correspond to the contour of the cluster or not:

$$w^d = \begin{cases} m(O^c), & \text{if contour} \\ m(O^c) \cdot \alpha_{\neg contour}, & \text{otherwise} \end{cases} \tag{4.15}$$

being $\alpha_{\neg contour}$ a design parameter lowering the weight of non-contour cells. A cell is considered to be part of the contour if the number of cells belonging to the same cluster between the cell and the sensors' reference system is less than $n_c^{contour}$—the set of cells in between both is calculated using Bresenham's algorithm.

Algorithm 2 provides the pseudo-code in order to better illustrate the calculation of the geometry-based bounding box.

The reliability of geometry-based bounding box estimation is directly related to L-shaped footprints. Therefore, the standard deviations of the variables defining the geometry-based box $\boldsymbol{b}^g$ are adjusted accordingly. The calculation of the standard deviation of the orientation variable θ^{b^g} given the geometry criteria is shown in (4.16). The standard deviation for the rest of the variables is computed equivalently.

$$\sigma_\theta^{b^g} = \begin{cases} \sigma_\theta^{b^g,min}, & \text{if } l_{\mathrm{x}}^{b^g} > \mathcal{T}_{l_{\mathrm{x}}}^L \wedge l_{\mathrm{y}}^{b^g} > \mathcal{T}_{l_{\mathrm{y}}}^L \\ \sigma_\theta^{b^g,max}, & \text{otherwise} \end{cases} \tag{4.16}$$

where $\sigma_\theta^{b^g,min}$ and $\sigma_\theta^{b^g,max}$ are experimentally adjusted and $\mathcal{T}_{l_{\mathrm{x}}}^L$ and $\mathcal{T}_{l_{\mathrm{y}}}^L$ are thresholds that denote when a L-shape is detected.

4.4.3. Bounding box based on velocity and geometry

As has been introduced in the previous sections, the bounding boxes calculated based on the velocity and the geometry are more or less reliable depending on the circumstances. Some procedures, such as the track state update step (Section 4.6.2.1) can correctly handle multiple estimations for the same variable while taking into account their uncertainties. However, other steps, such as the object matching or the velocity feedback (Sections 4.6.2.2 and 4.7) require a single bounding box

Algorithm 2: Geometry-based bounding box calculation

Input: Cells corresponding to the cluster $\boldsymbol{\mathcal{C}}^{cl}$ and OG $\mathcal{G}^{O}$
Output: Geometry-based bounding box $\boldsymbol{b}^{g}$

```
// Initialize score
```
$s_{fit}^{min} = \text{inf}$

for $\theta \leftarrow 0$ **to** 180 **do**

```
    // Calculate bounding box of the cells given θ
```
    $\boldsymbol{b} \leftarrow CalculateOrientedBoundingBox(\boldsymbol{\mathcal{C}}^{cl}, \theta)$

```
    // Calculate free-based fitting score
```
    $\boldsymbol{\mathcal{C}}^{b} \leftarrow SelectCellsInsideBox(\boldsymbol{b}, \mathcal{G})$
    $s_{free} \leftarrow CalculateFreeScore(\boldsymbol{\mathcal{C}}^{b}, \mathcal{G}^{O})$

```
    // Calculate contour-based fitting score
```
    $\boldsymbol{\mathcal{C}}^{cl}_{contour} \leftarrow SelectContourCells(\boldsymbol{\mathcal{C}}^{cl})$
    $s_{contour} \leftarrow CalculateContourScore(\boldsymbol{\mathcal{C}}^{cl}_{contour}, \boldsymbol{\mathcal{C}}^{cl}, \mathcal{G}^{O})$

```
    // Calculate fitting score and update geometry-based box
```
    $s_{fit} = s_{free} \cdot s_{contour}$
    if $s_{fit} < s_{fit}^{min}$ **then**
        $s_{fit}^{min} = s_{fit}$
        $\boldsymbol{b}^{g} = \boldsymbol{b}$
    end

end

representation. For this reason, an oriented bounding box $\boldsymbol{b}^{vg}$ based on both criteria, velocity and geometry, is calculated.

This bounding box is computed as the minimum box that contains the clustered cells given the orientation $\theta^{b^{vg}}$. This orientation is obtained as the weighted mean that takes into account the standard deviations of the orientations based on the velocity $\sigma_{\theta}^{b^{v}}$ and based on the geometry $\sigma_{\theta}^{b^{g}}$:

$$\theta^{b^{vg}} = \alpha^{b^{vg}} \cdot \theta^{b^{g}} + (1 - \alpha^{b^{vg}}) \cdot \theta^{b^{v}} \tag{4.17}$$

where:

$$\alpha^{b^{vg}} = \frac{\sigma_{\theta}^{b^{v}}}{\sigma_{\theta}^{b^{v}} + \sigma_{\theta}^{b^{g}}} \tag{4.18}$$

The motivation of this bounding box calculation is illustrated in Figure 4.7. In the case of the green car, the velocity-based box is more accurate than the geometry-based box since the vehicle is moving straight and only the rear of the vehicle is detected. On the contrary, in the case of the blue car, the velocity-based box is less accurate because the velocity vector is delayed with respect to the vehicle's real orientation while an L-shape is detected leading to an accurate geometry-based box.

4.4.4. Bounding box based on maximal length

The bounding box based on the maximization of length is specifically calculated to obtain a reference of the size of large off-road obstacles. This is required by the COG explained in Section 4.5.2 in order to identify clusters corresponding to the class *wall*. The calculation is equivalent to the one used for $\boldsymbol{b}^g$ in Section 4.4.2, but replacing the fitting score by the largest side of the box.

4.5. Classified Occupancy Grid

The grid map introduced in Chapter 3 provides a general description of the surrounding environment in terms of occupancy and dynamics. Nevertheless, no distinction about the objects' types is made. This section addresses this deficit by proposing a new classification approach that combines the particle filtering of the DOG, a naive Bayes classifier and object-level assumptions. With this goal, occupied cells are labeled with respect to relevant objects:

$$\mathcal{L}^{obj} = \{dyn.\ vehicle,\ dyn.\ pedestrian,\ wall,\ other\} \tag{4.19}$$

The set of classes selected is motivated by the requirements of the posterior steps: the object-level tracking, Section 4.6—which focuses on dynamic road users—and the velocity feedback, Section 4.7—which addressed vehicles and large off-road obstacles (walls).

4.5.1. Relationship between object-level representation and Particle Filter

The COG takes advantage of both representations: cell-level and object-level. The estimation over time is performed at cell-level, while the input data is obtained at object-level.

On the one hand, the estimation over time is performed at cell-level by including the class probabilities in the state definition of the cells and the particles. Therefore, each cell and each particle stores a probability $p(\ell^{obj})$ for each object class $\ell^{obj} \in \mathcal{L}^{obj}$. In this way, the processes of birth, object existence probability estimation, position prediction and data association are directly accomplished along the already implemented DOG framework. The update step is also carried out at cell-level but through the incorporation of a naive Bayes classifier.

On the other hand, no external source of information providing classification measurements is available in this thesis. Instead, input classification data is obtained at object-level by analyzing the objects extracted from the grid map with respect to different object-level features. In this way, a measurement COG is directly obtained containing the probabilities of each cluster corresponding to the four possible classes. Note that if other sources of information about object classification are available, e.g. camera image classification, the recursive grid-based estimation can be accomplished in the same way by mapping this data into the grid cells.

4.5.2. Object-based measurement Classified Occupancy Grid

The measurement COG consists of a grid map representation that contains the probabilities of each cell of corresponding to each type of obstacle in $\mathcal{L}^{obj}$ given the input data at the current iteration. To achieve this measurement grid, objects are segmented from the grid map, as explained in Section 4.3, and evaluated with respect to four representative object-level features: velocity (f_v), size (f_l), height (f_h), and location with respect to the road (f_r).

Size and velocity are common descriptors for road users. Nevertheless, they are not robust enough for accurate class labeling in DOGs. On the hand, in grid maps, the estimated shapes are limited to the sides perceived by the sensor. Additionally, especially in cases of sparse data, it is common to find vehicles described by small sets of cells, see Section 3.8.3.3 for some examples. On the other hand, dynamic estimation can present incorrect velocities or slow convergence. Moreover, in urban scenarios, vehicles can move at low speeds, making the classification more challenging.

For these reasons, the height and location with respect to the road features are also included with the intention of obtaining a more reliable classification. For example, the location with respect to the road helps to differentiate between dynamic vehicles and dynamic false positives caused by guardrails or temporal occlusions, while the height feature allows distinguishing between low-speed pedestrians and streetlights, trees, etc.

Three of these features are directly obtained from the cluster's cells: (i) the height feature corresponds to the maximum height of the clustered cells, (ii) the velocity feature is calculated as the module of the weighted average velocity, and (iii) the road feature is calculated as the percentage of cells inside the road. On the contrary, since the whole shape of objects is not detected, the size feature f_l is approximated as the longest side of the bounding box gathering the cluster. Additionally, taking into account that the box-based representation correctly fits vehicles and pedestrians, but not large off-road obstacles, two bounding box calculations are used. For the classes *dyn. vehicle* and *dyn. pedestrian*, the bounding box oriented as the velocity vector is used, see Section 4.4.1. For class *wall*, the oriented bounding box is selected as the one that maximizes the length, see Section 4.4.4.

In order to evaluate the likelihood of each class with respect to these four features, a likelihood function for each feature and class is experimentally defined using combinations of sigmoidal functions. The defined functions are shown in Figure 4.8 and briefly explained in the following:

- **Size feature:** As explained above, the size feature may not be as reliable as expected, hence, it is used as a loose class classifier. The size expected for pedestrians considers values from 0 to 2 m, being more probable the small size values than the values close to 2 m. The size for large off-road obstacles is tightly defined, only accepting objects with a length larger than 6 m. Conversely, a wide range of values is accepted for the class *dyn. vehicles*, since it represents a great variety of objects, from bicycles to trucks.
- **Velocity feature**: The velocity feature is used to identify dynamic road users. Additionally, it also allows to distinguish between pedestrians and fast-moving vehicles. Vehicles can move

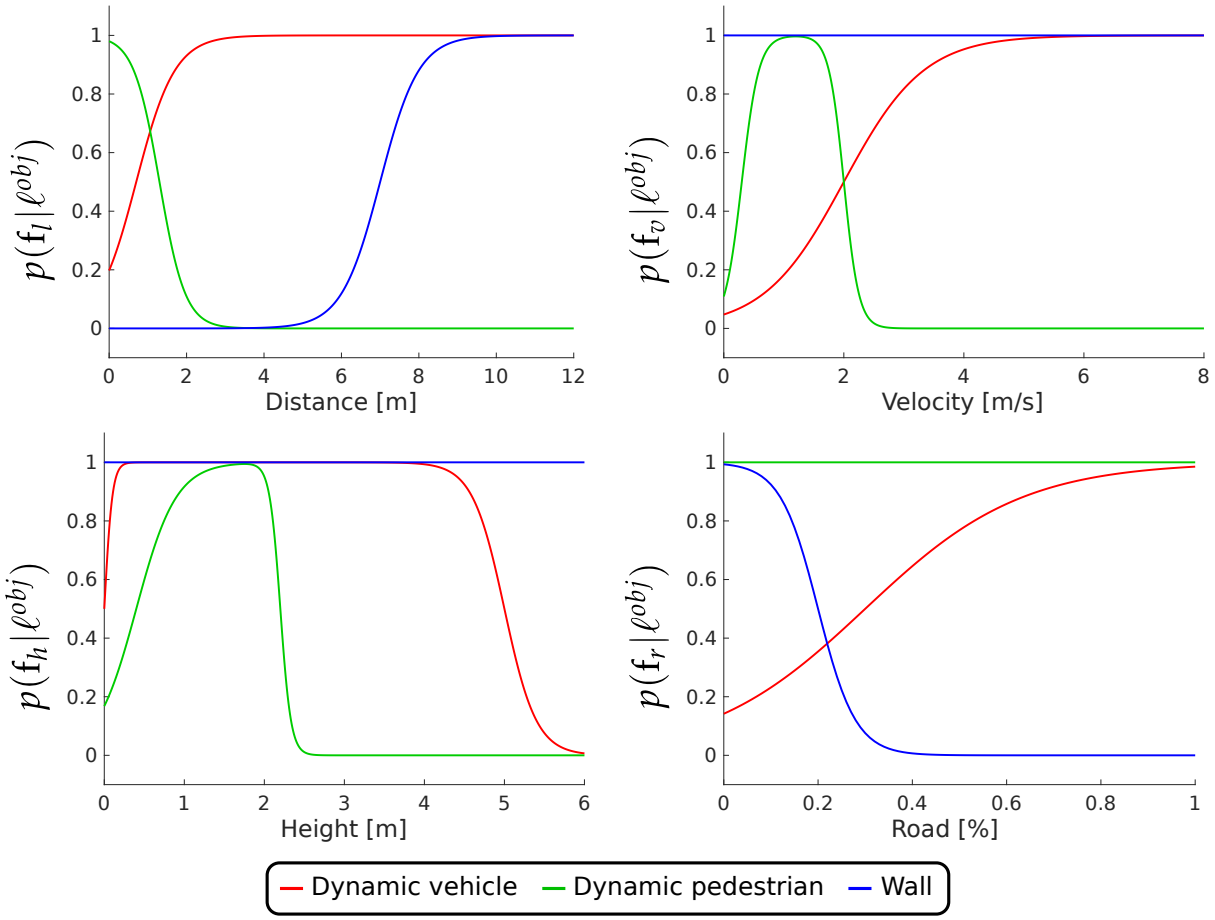


Figure 4.8: Likelihood functions with respect each class and each object-level feature.

at low speeds, thus these values are allowed for vehicles, but with less confidence. For the class *wall*, every velocity is accepted, this is motivated by a common issue of LiDAR-based DOGs which tend to model wrong dynamics for these types of obstacles.

- **Height feature:** The height feature is primarily included for the correct identification of pedestrians. It could be defined more tightly for vehicles in order to enhance their detection, but, given the available sensor configuration, vehicles located at farther distances are commonly detected by few LiDAR points.
- **Road feature:** As introduced earlier, the location with respect to the road is defined seeking to differentiate between dynamic vehicles and dynamic false positives outside the road. Taking into account that, in some cases, vehicles travel outside the road limits and large obstacles, such as guardrails, are in close proximity to the road, these functions are loosely defined. Pedestrians, on the contrary, can be expected inside and outside the road.

For the class *other*, as it addresses all the rest of the object types, no likelihood functions are defined. Instead, it is modeled as a constant value P_{other}.

Notice that, since the evaluated objects are extracted from the grid map, the mapping of the computed likelihoods to the grid cells $p(\cdot|\ell_{z_k}^{obj,c})$ is directly obtained.

The classified input data for unclustered cells is a special case since no class information can be inferred for these cells. Unclustered cells are those cells that do not have enough occupied evidence or enough neighboring occupied cells (see Section 4.3). These cells usually appear (i) with new object detections, (ii) in the proximity of sensed objects, or (iii) due to noise. Therefore, in this case, class *other* should be favored but without neglecting the rest of the classes. With this intention, P_{other} is set to 1, and the likelihoods for *dyn. vehicle*, *dyn. pedestrian* and *wall* are replaced by a lower value $P_{unclustered}$.

The classification input data just introduced is sensitive to clustering errors. For example, if a group of pedestrians in a crosswalk is incorrectly merged into a single cluster, the feature-based classification analysis may favor the class *dyn. vehicle*. In order to address this issue, a confidence value $\alpha_{z_k}^{obj}$ modeling the reliability of the classification performed is included. This value is equally set for every cell and is experimentally adjusted in order to obtain smooth changes in the classification. Nevertheless, it can be parameterized in order to properly denote the confidence of each particular estimated classification.

4.5.3. Classified Occupancy Grid recursive realization

The classification presented in the previous section corresponds to a single-frame classification and, therefore, it is prone to intermittent noise. This section introduces the recursive estimation procedure that allows to obtain a more reliable classification over time.

4.5.3.1. Prediction, birth and existence probability

As already introduced in Section 4.5.1, by introducing the class probabilities into the particles' states, the prediction of the objects' location, the birth process and the existence probability estimation are directly achieved by the PF's workflow. Recall that particles can move freely along the cells of the grid, their weights are directly related to the occupancy state of the cells and, in each iteration, new particles are generated and the set of persistent particles is resampled based on particles' weight. See Section 3.6 for further details.

During the prediction step, since particle dynamic state may not correctly model the real object displacement and in order to avoid singularities, class probabilities are limited to a minimum value $p_{\ell^{obj}}^{min}$:

$$p(\ell_{(k|k-1)}^{obj,\rho}) = \max\left(p_{\ell^{obj}}^{min},\ p(\ell_{(k-1|k-1)}^{obj,\rho})\right) \tag{4.20}$$

where $p(\ell_{(k|k-1)}^{obj,\rho})$ and $p(\ell_{(k-1|k-1)}^{obj,\rho})$ are the predicted and prior probabilities of the particle ρ for label ℓ^{obj}, respectively.

During particle initialization, the probabilities are set with full probability for the class *other* and

zero for the rest.

4.5.3.2. Classification update

In order to perform the classification over time, a naive Bayes classifier is used. This method relies on the Bayes theorem and performs the "naive" assumption that the features involved are conditionally independent of each other. Thus, the classification problem can be modeled as follows:

$$p(\ell^{obj}|\mathrm{f}_l, \mathrm{f}_v, \mathrm{f}_h, \mathrm{f}_r) = \frac{p(\ell^{obj}) \cdot p(\mathrm{f}_l, \mathrm{f}_v, \mathrm{f}_h, \mathrm{f}_r|\ell^{obj})}{p(\mathrm{f}_l, \mathrm{f}_v, \mathrm{f}_h, \mathrm{f}_r)} \tag{4.21}$$

since independence is assumed between the features:

$$p(\ell^{obj}|\mathrm{f}_l, \mathrm{f}_v, \mathrm{f}_h, \mathrm{f}_r) \propto p(\ell^{obj}) \cdot p(\mathrm{f}_l|\ell^{obj}) \cdot p(\mathrm{f}_v|\ell^{obj}) \cdot p(\mathrm{f}_h|\ell^{obj}) \cdot p(\mathrm{f}_r|\ell^{obj}) \tag{4.22}$$

where $p(\cdot|\ell^{obj})$ are the likelihoods with respect to each feature, computed as explained in Section 4.5.2.

Therefore, in each iteration, the class probabilities of a particle ρ inside a cell c are updated using a variation of (4.22) that takes into account the confidence value α_ℓ^{obj}:

$$p(\ell^{obj,\rho}_{(k|k)}) = \eta^{\ell^{obj}} \cdot \left(p(\ell^{obj,\rho}_{(k|k-1)}) \cdot l^{obj,c} \cdot \alpha^{obj}_{z_k} + p(\ell^{obj,\rho}_{(k|k-1)}) \cdot (1 - \alpha^{obj}_{z_k}) \right) \tag{4.23}$$

being:

$$l^{obj,c} = p(\mathrm{f}_l|\ell^{obj,c}_{z_k}) \cdot p(\mathrm{f}_v|\ell^{obj,c}_{z_k}) \cdot p(\mathrm{f}_h|\ell^{obj,c}_{z_k}) \cdot p(\mathrm{f}_r|\ell^{obj,c}_{z_k}) \tag{4.24}$$

and $\eta^{\ell^{obj}}$ a weight calculated in order to normalize the probabilities of the four classes.

4.5.3.3. Cell's classification estimation

Similarly to the cell's dynamic state estimation (3.46), the probability of a cell c of being occupied by an obstacle with label ℓ^{obj} is computed by:

$$p(\ell^{obj,c}) = \left(\sum_{i=1}^{\nu_U^c} \omega^{i,\rho,c} \right)^{-1} \cdot \sum_{i=1}^{\nu_U^c} \left(\omega^{i,\rho,c} \cdot p(\ell^{obj,i,\rho}_{(k|k)}) \right) \tag{4.25}$$

where i refers to the i-th particle ρ within the cell c, ω to its weight and ν_U^c denotes the number of particles that have been updated at least n_U^ρ times.

Cell's most likely class $\ell^{*,obj,c}$ is selected as the class with the highest probability:

$$\ell^{*,obj,c} = \underset{\ell^{obj} \in \mathcal{L}^{obj}}{\arg\max} \; p(\ell^{obj,c}) \tag{4.26}$$

4.5.4. Classified Occupancy Grid output visualization

The COG visualization is performed similarly to the DOG's visualization (Section 3.6.4), but substituting the dynamic state by the object classification.

Therefore, in order to visualize the classified environment estimation in a comprehensive way, this thesis represents the occupancy probability using the classical grayscale, and the most probable object class using a color code. The classification is only represented in the cells that are considered as occupied by an obstacle $m(O) > \mathcal{T}^o$. The color code is selected to represent each of the four possible classes while brightness is proportional to the probability value. Figure 4.9 displays an example of a COG representation.

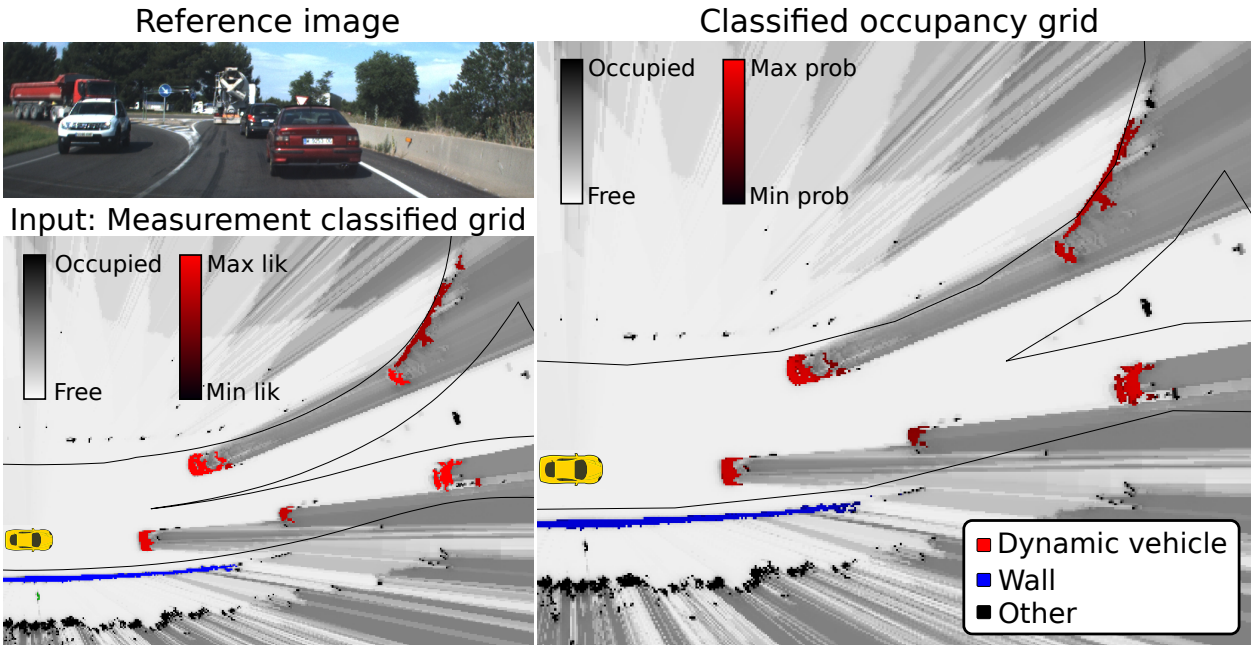


Figure 4.9: Classified Occupancy Grid example.

4.6. Object-level tracking focused on road users

As introduced in Section 4.1.1, an object-level representation of the road users is required by several of the subsequent modules. In the following, the object-level tracking strategy used in this thesis is presented making special emphasis on the adaptations required for its integration into the proposed perception framework.

4.6.1. Tracking strategy overview

The object-level estimation of the road users present in the scene is a multi-object tracking problem that requires the simultaneous state estimation of a time-varying number of objects that may appear and disappear at any time. As reviewed in Section 4.2.1, this problem has been widely

studied and, hence, multiple approaches can be found in the literature.

The approach employed in this thesis consists of three main steps: (i) the extraction of objects from the input data, (ii) the association between the extracted objects and the object tracks, and (iii) the filtering over time of the object tracks' states. The object extraction procedure has already been introduced in Section 4.3 and the matching is explained in detail in Section 4.6.2.2. For the latter step, a multi-object Extended Kalman Filter (EKF) approach is used, some works introducing its application to multi-object tracking and including different variations are [141, 150, 161].

Briefly summarizing the object-level tracking workflow, in each iteration the state of existing tracks is predicted to the current instant using a constant velocity model. Then, objects are extracted from the grid map and compared with the predicted object tracks. An association one-to-one is performed and the state of tracks that have been matched with an extracted object is updated accordingly. Extracted objects that have not been matched are used to initialize new tracks, while object tracks that have not been matched for a certain number of iterations are deleted.

4.6.2. Tracks and extracted objects representation

Tracked objects are represented by (i) a dynamic state, (ii) an identification number, and (iii) a class label. The dynamic state is defined by an oriented 3D bounding box and a velocity magnitude:

$$\boldsymbol{s}^{\tau} = \left[x_{\mathrm{x}}, x_{\mathrm{y}}, \theta, v, l_{\mathrm{x}}, l_{\mathrm{y}}, l_{\mathrm{z}}\right] \tag{4.27}$$

where $\tau \in \boldsymbol{\mathcal{T}}$ refers to a track of the set of objects tracks $\boldsymbol{\mathcal{T}}$, x_{x} and x_{y} are the 2D coordinates of the center of the box, θ the orientation, v the velocity, and l_{x}, l_{y} and l_{z} the length, width, and height, respectively. Each variable is assumed as Gaussian.

Object tracks are divided into vehicles and pedestrians. For the case of vehicles, an additional classification depending on the vehicle type is performed. Therefore, object tracks are classified into:

$$\mathcal{L}^{\tau} = \{car,\ truck,\ cycle,\ pedestrian\} \tag{4.28}$$

Objects are extracted from the grid map following the approach proposed in Section 4.3.2. Each extracted object $\gamma \in \boldsymbol{\Gamma}$ is described by:

$$\boldsymbol{s}^{\gamma} = \left[\boldsymbol{b}^{v}, \boldsymbol{b}^{g}, \boldsymbol{b}^{vg}, v_{\mathrm{x}}, v_{\mathrm{y}}, l_{\mathrm{z}}, \boldsymbol{\ell}^{obj}\right] \tag{4.29}$$

where v_{x} and v_{y} is the velocity vector computed as the weighted average velocity of the clustered cells, l_{z} is the maximum height of the clustered cells, and $\boldsymbol{\ell}^{obj}$ is a vector containing the weighted average probabilities for each class in $\mathcal{L}^{obj}$. $\boldsymbol{b}$ refers to an oriented minimum bounding box and superindices v, g, and vg, denote if the bounding box is computed based on (i) the velocity vector, (ii) the footprint geometry, or (iii) the combination of velocity and geometry, respectively—see Section 4.4.

4.6.2.1. Track state update based on object type

Tracks' update is performed differently for vehicles and pedestrians. Pedestrians' real size is well approximated by the obtained clusters. Thus, the velocity-based bounding box $\boldsymbol{b}^v$ is sufficient for location and shape estimation. Then, pedestrians' observations are defined as:

$$\boldsymbol{z}^{ped} = \left[x_{\mathrm{x}}^{b^v,\gamma}, x_{\mathrm{y}}^{b^v,\gamma}, v_{\mathrm{x}}^{\gamma}, v_{\mathrm{y}}^{\gamma}, l_{\mathrm{x}}^{b^v,\gamma}, l_{\mathrm{y}}^{b^v,\gamma}, h^{\gamma}\right] \tag{4.30}$$

where γ refers to the extracted object selected for the update.

On the contrary, the vehicles' real shape is not captured by the DOG. Therefore, for vehicles, the information about the footprint geometry and the estimated vehicle class is also taken into account. Moreover, in order to avoid errors due to sensed footprint changes, the update is performed using a reference point.

In this work, the reference point can be either the center of a side or a corner. The reference point $[a_{\mathrm{x}}^{\tau}, a_{\mathrm{y}}^{\tau}]$ of the track τ is calculated by checking the theoretical visibility of its sides as explained in [59]. The reference points $[a_{\mathrm{x}}^{b^v,\gamma}, a_{\mathrm{y}}^{b^v,\gamma}]$ and $[a_{\mathrm{x}}^{b^g,\gamma}, a_{\mathrm{y}}^{b^g,\gamma}]$ of the extracted objects' bounding boxes $\boldsymbol{b}^v$ and $\boldsymbol{b}^g$ are calculated following the approach based on free space presented in [141]. Since these three reference points can differ, the reference points of the extracted object's bounding boxes are transformed into the track's reference point. As a result, the extracted object's reference points for the update are computed as follows:

$$\boldsymbol{a}^{b,\tau,\gamma} = \boldsymbol{a}^{b,\gamma} + \boldsymbol{R}_{\theta^{b,\gamma}} \begin{bmatrix} \delta(l_{\mathrm{x},(k-1|k-1)}^{\tau}) \\ \delta(l_{\mathrm{y},(k-1|k-1)}^{\tau}) \end{bmatrix} \tag{4.31}$$

where $\boldsymbol{R}_{\theta^{b,\gamma}}$ is the rotation matrix with respect to the extracted object's box orientation and $\delta(l_{\mathrm{x},(k-1|k-1)}^{\tau})$, $\delta(l_{\mathrm{y},(k-1|k-1)}^{\tau})$ are selected with respect to the tracks' prior size and the reference points relation:

$$\delta(l_{\mathrm{x},(k-1|k-1)}^{\tau}) \in \{-1, -0.5, 0, 0.5, 1\} \cdot l_{\mathrm{x},(k-1|k-1)}^{\tau} \tag{4.32}$$

$$\delta(l_{\mathrm{y},(k-1|k-1)}^{\tau}) \in \{-1, -0.5, 0, 0.5, 1\} \cdot l_{\mathrm{y},(k-1|k-1)}^{\tau} \tag{4.33}$$

Figure 4.10 provides two examples of the definition of variables $\delta(l_{\mathrm{x},(k-1|k-1)}^{\tau})$, $\delta(l_{\mathrm{y},(k-1|k-1)}^{\tau})$.

In view of the above, the state of the observation of a vehicle is defined as:

$$\boldsymbol{z}^{veh} = [a_{\mathrm{x}}^{b^v,\tau,\gamma}, a_{\mathrm{y}}^{b^v,\tau,\gamma}, \theta^{b^v,\gamma}, l_{\mathrm{x}}^{b^v,\gamma}, l_{\mathrm{y}}^{b^v,\gamma}, a_{\mathrm{x}}^{b^g,\tau,\gamma}, a_{\mathrm{y}}^{b^g,\tau,\gamma}, \theta^{b^g,\gamma}, l_{\mathrm{x}}^{b^g,\gamma}, l_{\mathrm{y}}^{b^g,\gamma}, v_{\mathrm{x}}^{\gamma}, v_{\mathrm{y}}^{\gamma}, l_{\mathrm{z}}^{\gamma}] \tag{4.34}$$

After the update, the track center is recomputed as:

$$\boldsymbol{x}_{(k|k)}^{\tau} = \boldsymbol{a}_{(k|k)}^{\tau} - \boldsymbol{R}_{\theta_{(k|k)}^{\tau}} \begin{bmatrix} \delta(l_{\mathrm{x},(k|k)}^{\tau,\ell^{\tau}}) \\ \delta(l_{\mathrm{y},(k|k)}^{\tau,\ell^{\tau}}) \end{bmatrix} \tag{4.35}$$

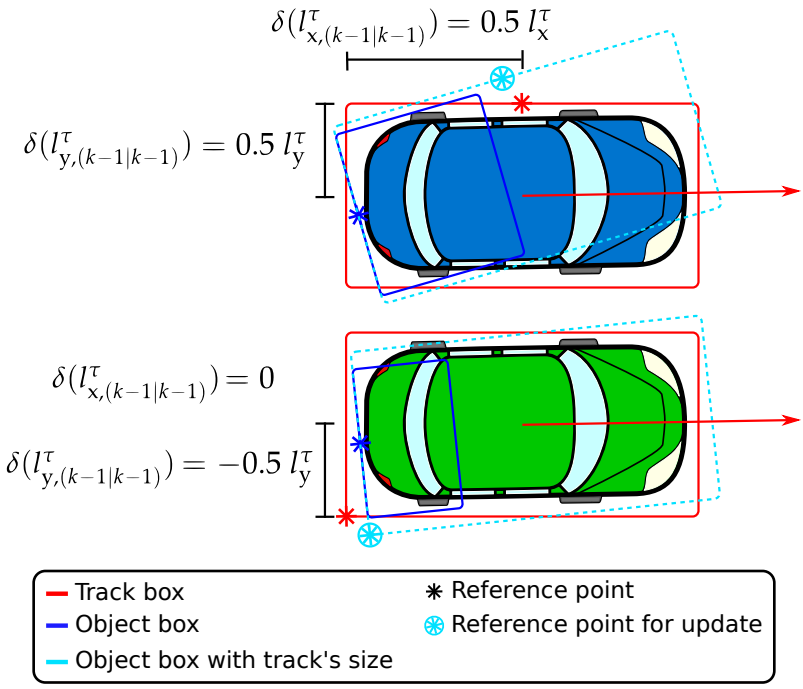


Figure 4.10: Illustrative examples of the calculation of the update reference point.

where $(k|k)$ denotes updated variables and box size variable $\delta(l^{\tau,\ell^{\tau}}_{x,(k|k)})$, $\delta(l^{\tau,\ell^{\tau}}_{y,(k|k)})$ are selected taking into account the track's updated size and vehicle class $\ell^{\tau} \in \mathcal{L}^{\tau}$ size:

$$\delta(l^{\tau,\ell^{\tau}}_{x,(k|k)}) \in \{-1, -0.5, 0, 0.5, 1\} \cdot \max\left(l^{\tau}_{x,(k|k)},\ l^{\ell^{\tau}}_{x}\right) \tag{4.36}$$

$$\delta(l^{\tau,\ell^{\tau}}_{y,(k|k)}) \in \{-1, -0.5, 0, 0.5, 1\} \cdot \max\left(l^{\tau}_{y,(k|k)},\ l^{\ell^{\tau}}_{y}\right) \tag{4.37}$$

4.6.2.2. Matching

The association problem is solved using the Global Nearest Neighbor [65] approach. This algorithm computes the cost of assigning each track to each extracted object. Then, it matches each track to a single extracted object seeking to minimize the total cost. The association cost is usually computed by comparing the state of the extracted objects and of the tracks. Nevertheless, given that the perception strategy employed in this work combines grid-based and object-based algorithms, two additional costs are included: a particle labeling association cost, inspired by [142], and an object class association cost that takes advantage of the classification of the COG.

First, the distance between predicted tracks and extracted objects is calculated. Only nearby tracks and extracted objects can be matched, i.e., located at a distance below the threshold $d^{max}_{matching}$. For associations fulfilling this requirement, an association cost $\kappa_{matching}$ is computed based on three features: (i) dynamic state similarity, (ii) class similarity, and (iii) particle association:

$$\kappa_{matching} = \kappa_{state} + \kappa_{class} + \kappa_{part} \tag{4.38}$$

where the subscripts refer to the cost of each one of the aforementioned features.

Dynamic state similarity (κ_{state}) is calculated by comparing the tracks and extracted objects position and velocity:

$$\kappa_{state} = \min\left(1, \frac{|\Delta\boldsymbol{x}(b^{vg,\gamma}, \tau)|}{\Delta_{pos}^{max}}\right) + \min\left(1, \frac{|\Delta\boldsymbol{v}(\gamma, \tau)|}{\Delta_{vel}^{max}}\right) \tag{4.39}$$

where $b^{vg,\gamma}$ refers to the object's bounding box calculated considering velocity and geometry (see Section 4.4.3), Δ_{pos}^{max} and Δ_{vel}^{max} are design parameters included to normalize the similarity between 0 and 1; therefore, the maximum dissimilarity for the dynamic state is $\kappa_{state}^{max} = 2$. Note that in the case of vehicles, the similarity is computed using the reference point $\boldsymbol{a}$ instead of the center $\boldsymbol{x}$.

The similarity regarding the COG's classification (κ_{class}) is computed taking into account whether the track corresponds to a vehicle or a pedestrian. The calculation for vehicles is as follows:

$$\kappa_{class} = \begin{cases} 0 & \ell^{*,obj,\gamma} = \text{vehicle} \\ \kappa_{class}^{max} & \ell^{*,obj,\gamma} = \text{other} \\ \infty & \text{otherwise} \end{cases} \tag{4.40}$$

the calculation for pedestrians is equivalently determined. Notice that the matching between objects of different classes is not allowed.

The similarity by particle association (κ_{part}) is calculated taking into account the number of particles labeled with the track's ID. Once a track is updated, particles belonging to its associated extracted object are labeled with the track's ID. In the next frame, if the majority of particles of the potential matching extracted object are labeled with the track's ID, κ_{part} is set to zero; otherwise, $\kappa_{part} = \kappa_{part}^{max}$.

Lastly, the cost of an object track to remain unassigned, i.e. not associated with any extracted object, is defined using the maximum similarity costs:

$$\kappa_{matching}^{max} = \kappa_{state}^{max} + \kappa_{class}^{max} + \kappa_{part}^{max} \tag{4.41}$$

4.6.2.3. Tracks initialization and deletion

As explained in Section 4.5, many obstacles are modeled in the grid map, but not all of them may be dynamic road users. Therefore, new tracks are only initialized from extracted objects that are likely to be a vehicle or a pedestrian i.e., the probability for the class *dyn. vehicle* or *dyn. pedestrian* exceeds the respective thresholds $\mathcal{T}_{veh}^{born,\tau}$ and $\mathcal{T}_{ped}^{born,\tau}$.

Additionally, vehicles can only be initialized if their velocity is high enough to provide a reliable orientation, i.e., above the velocity threshold $\mathcal{T}_{v}^{born,\tau}$ and if there are no other vehicles in its immediate vicinity. The immediate vicinity surrounding a track is considered as the area covered by the track's

box enlarged by a distance $\mathcal{T}_d^{born,\tau}$.

Tracks are confirmed if they are detected at least $n_{confirm}^{\tau}$ times and deleted if they are not detected during n_{delete}^{τ} consecutive frames.

4.6.3. Vehicle classification

Object tracks are classified into: $\mathcal{L}^{\tau} = \{car,\ truck,\ cycle,\ pedestrian\}$. When a track is initialized, it is classified into *vehicle* or *pedestrian* based on the label $\ell^{*,obj,\gamma}$ of the associated extracted object γ. If the label corresponds to *dyn. vehicle*, the extracted object is further classified with respect to the three vehicle possible classes and assigned to the most likely one (4.42). Each time a vehicle track is updated, the associated extracted object is evaluated with respect to the larger size classes, i.e., (i) *cycles* can change to *car* or *truck*, (ii) *cars* can change into *truck*. A new class is assigned if it is positively evaluated at least $n^{\ell^{\tau}}$ times.

$$\ell^{\tau} = \begin{cases} \text{truck}, & l_{\mathrm{z}}^{\gamma} \geq h^{truck} \ \wedge \ \left(l_{\mathrm{x}}^{b^{vg},\gamma} \geq l_{\mathrm{x}}^{truck} \ \vee \ l_{\mathrm{y}}^{b^{vg},\gamma} \geq l_{\mathrm{y}}^{truck} \right) \\ \text{cycle}, & l_{\mathrm{z}}^{\gamma} < h^{truck} \ \wedge \ \left(l^{b^{vg},\gamma} \leq l_{\mathrm{x}}^{car} \ \wedge \ l_{\mathrm{y}}^{b^{vg},\gamma} \leq l_{\mathrm{y}}^{car} \right) \\ \text{car}, & \text{otherwise} \end{cases} \tag{4.42}$$

where γ refers to the extracted object, b^{vg} to the bounding box based on velocity and geometry, and h^{truck}, l_{x}^{truck}, l_{y}^{truck}, l_{x}^{car} and l_{y}^{car} are design parameters defining the size of a truck and of a car.

Figure 4.11 provides an example of the road users object-level tracking obtained from applying the proposed strategy. It can be seen that all road users are correctly segmented from the grid map and accurate bounding boxes are assigned according to their perceived shape and vehicle class. Additionally, the vehicle classification process can be observed in the truck located in the same lane as the ego-vehicle. Due to occlusions and high sparsity of LiDAR data at that distance, its real size has not been detected yet. Therefore, while the class *cycle* has been dismissed, the confirmation of its classification as belonging to the *truck* class is still pending.

4.7. Object-level velocity feedback for Dynamic Occupancy Grid

As explained in Section 4.2.2.1 and exposed in the results of Chapter 3, DOGs are able to provide reliable results using only LiDAR data. However, better dynamic estimations can be obtained if velocity measurements are added. Moreover, the assumption of independent cells sometimes leads to a non-uniform velocity representation of the same object.

This section introduces a velocity feedback method that seeks to address both issues. It takes advantage of the extracted objects and object-level assumptions to infer velocity data which is then used to update the dynamic estimation of the grid map. Moreover, since a single velocity value is modeled for each object, all the corresponding cells are fed with the same velocity information, thus, the independence condition between the cells is mitigated.

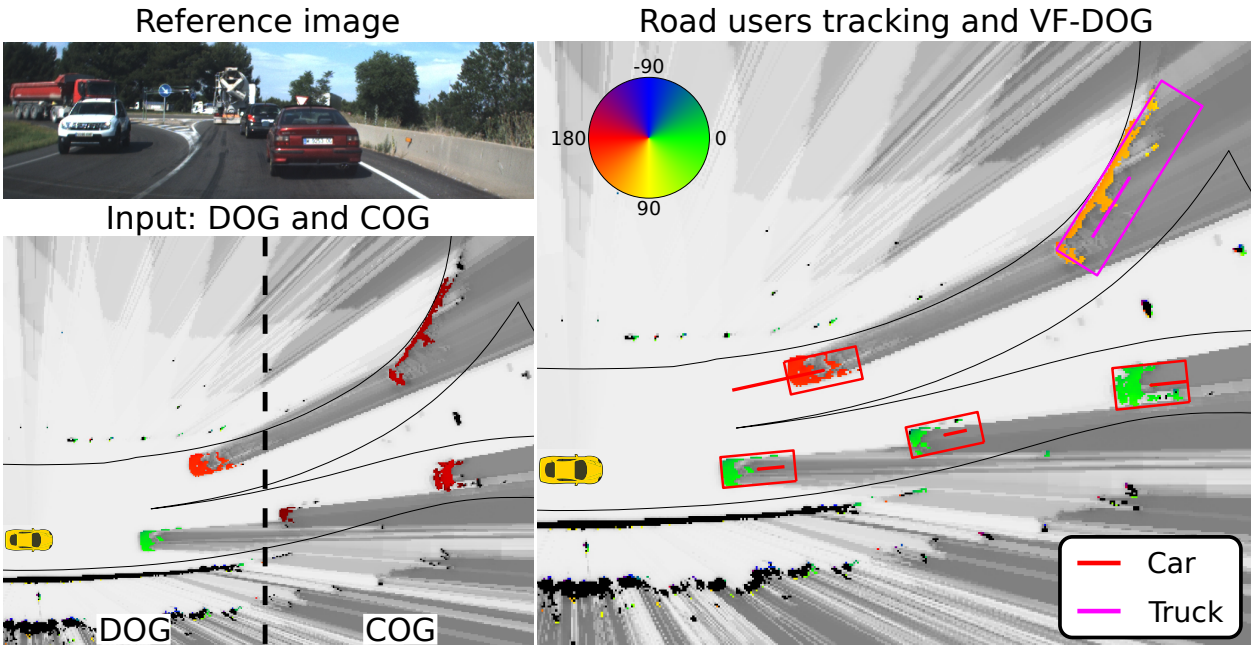


Figure 4.11: Road users object-level tracking example.

4.7.1. Velocity feedback calculation for the Dynamic Occupancy Grid

The DOG introduced in Section 3.6 is already prepared to integrate velocity measurements through the update and initialization steps. The proposed velocity feedback is intended to serve as a substitute or complement to these velocity measurements. Thus, the velocity data calculated is expressed as required by these steps: (i) rasterized into the grid cells and (ii) described in terms of a Gaussian velocity measurement ϑ_{z_k} coupled with a confidence value $\alpha_{z_k}^{\vartheta}$ (see Section 3.3).

The velocity feedback data is calculated for two types of objects: (i) dynamic vehicles and (ii) large off-road obstacles. For the rest of the objects or unclustered cells, no velocity feedback is computed, and therefore the confidence value is set to zero, $\alpha_{z_k}^{\vartheta} = 0$.

4.7.1.1. Velocity feedback for large off-road obstacles

The velocity feedback for large off-road obstacles addresses the known problem of large structures wrongly detected as dynamic by particle-based DOGs [143, 148, 129].

For this purpose, the semantic information estimated by the COG is used. In this way, occupied cells labeled as *wall* are assumed to correspond to a static obstacle; hence, the velocity feedback for the cell c is set to zero:

$$\vartheta_{z_k}^{c} \sim \mathcal{N}(0, \mathbf{\Sigma}_{\vartheta}) \tag{4.43}$$

where $\mathbf{\Sigma}_{\vartheta}$ is the covariance matrix experimentally parametrized through a quantitative evaluation with ground truth data.

The reliability of this estimation is proportional to the probability of the cell correctly correspond-

ing to the label *wall*:

$$\alpha_{z_k}^{\vartheta,c} = \alpha_{\vartheta}^{max} \cdot p(\ell^{obj,c} = wall) \tag{4.44}$$

4.7.1.2. Velocity feedback for dynamic vehicles

The accurate estimation of other vehicles in the scene is one of the main objectives of the perception framework developed in this thesis. While LiDAR-based DOGs are able to correctly model most of the dynamic vehicles, slow convergence may occur in certain situations such as turning maneuvers or large targets for which multiple motion interpretations can explain the perceived shape. In order to enhance the estimation in these situations, velocity data is inferred by calculating vehicle's displacement in consecutive iterations.

The same extracted objects as for the object-level tracking are used. Also, in order to achieve a robust displacement calculation, the box-based shape assumption and reference point calculation explained in Section 4.6.2.1 are also employed.

Therefore, the estimated velocity $\boldsymbol{v}^{\gamma}$ for the extracted object γ, associated to the vehicle track τ at time instant (k) is calculated as follows:

$$\boldsymbol{v}^{\gamma} = \frac{\boldsymbol{x}_{(k)}^{\gamma,\tau,b^{vg\prime}} - \boldsymbol{x}_{(k-1)}^{\gamma,\tau,b^{vg\prime}}}{\Delta \mathrm{t}} \tag{4.45}$$

where Δt is the time difference between iterations and $\boldsymbol{x}^{\gamma,\tau,b^{vg\prime}}$ is the center of the bounding box b^{vg} recalculated with respect to track's size to avoid false dynamic estimations due to estimated size changes (4.46). Figure 4.12 provides an illustrative example of the calculation of this center point. The respective cluster at the previous iteration $(k-1)$ is obtained using the track's historical data, i.e., the extracted object used to update track τ at time $(k-1)$.

$$\boldsymbol{x}^{b^{vg\prime}} = \boldsymbol{a}^{b^{vg}} - \boldsymbol{R}_{\theta^{b^{vg}}} \begin{bmatrix} \delta(l_{\mathrm{x}}^{\tau,\ell^{\tau}}) \\ \delta(l_{\mathrm{y}}^{\tau,\ell^{\tau}}) \end{bmatrix} \tag{4.46}$$

This estimated velocity is used to define the velocity feedback for the cells corresponding to the cluster γ as follows:

$$\vartheta_{z_k}^{c,\gamma} \sim \mathcal{N}(\boldsymbol{v}^{\gamma}, \boldsymbol{\Sigma}_{\vartheta}) \tag{4.47}$$

The confidence value $\alpha_{z_k}^{\vartheta,\gamma}$ is calculated taking into account (i) the reliability of the matching between the track and the object ($\alpha_{matching}^{\vartheta}$), (ii) a reference of correct dynamic behavior ($\alpha_{behavior}^{\vartheta}$), computed comparing the estimated velocity and track's dynamic state, and (iii) a design parameter included to avoid full initialization and updating of the particles based on the estimated velocities ($\alpha_{z_k}^{\vartheta,max}$):

$$\alpha_{z_k}^{\vartheta,\gamma} = \alpha_{matching,\gamma}^{\vartheta} \cdot \alpha_{behaviour,\gamma}^{\vartheta} \cdot \alpha_{z_k}^{\vartheta,max} \tag{4.48}$$

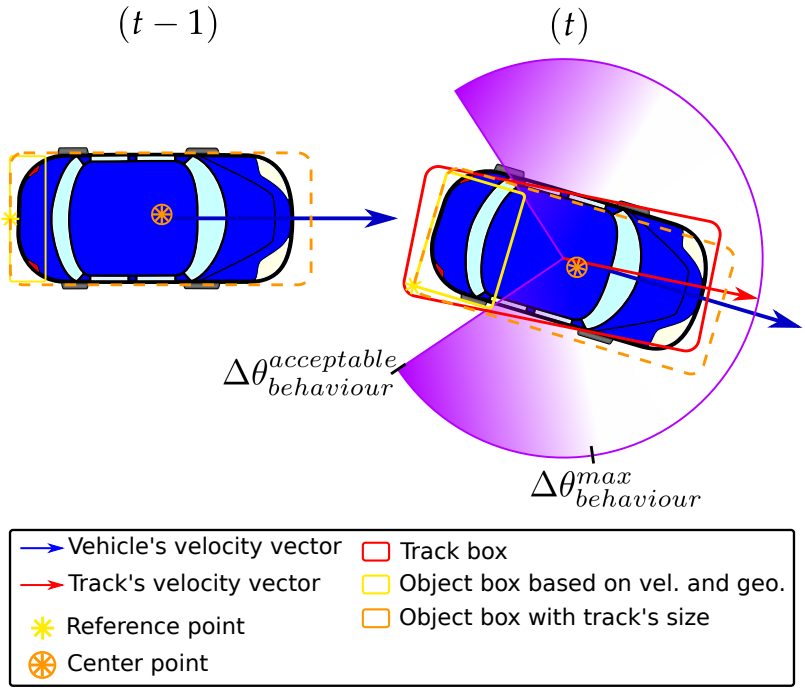


Figure 4.12: Illustrative example of velocity feedback calculation.

The reliability regarding the matching is measured considering the association cost between the track and the matched object, and all the other potential matches (see Section 4.6.2.2):

$$\alpha^{\vartheta,\gamma}_{matching} = \frac{\kappa^{max}_{matching} - \kappa^{\tau,\gamma}_{matching}}{\sum_i^{\mathbf{\Gamma}} (\kappa^{max}_{matching} - \kappa^{\tau,i}_{matching})} \tag{4.49}$$

$\alpha^{\vartheta}_{behaviour}$ seeks to filter atypical velocity estimations by considering the track's estimation as an approximated reference of correct behavior:

$$\alpha^{\vartheta,\gamma}_{behaviour} = \min\left(1,\ \max\left(0, \Delta^{behaviour}_{v} \cdot \Delta^{behaviour}_{\theta}\right)\right) \tag{4.50}$$

being $\Delta^{behaviour}_{v}$ and $\Delta^{behaviour}_{\theta}$ similarity measures regarding the velocity and orientation, respectively, defined as follows:

$$\Delta^{behaviour}_{v} = \min\left(1,\ \max\left(0, \frac{\left|v^{\tau} - ||\boldsymbol{v}^{\gamma}||\right| - \Delta v^{max}_{behaviour}}{\Delta v^{acceptable}_{behaviour} - \Delta v^{max}_{behaviour}}\right)\right) \tag{4.51}$$

$$\Delta^{behaviour}_{\theta} = \min\left(1,\ \max\left(0, \frac{\left|\theta^{\tau} - \arctan\left(v^{\gamma}_{y}/v^{\gamma}_{x}\right)\right| - \Delta\theta^{max}_{behaviour}}{\Delta\theta^{acceptable}_{behaviour} - \Delta\theta^{max}_{behaviour}}\right)\right) \tag{4.52}$$

Notice that $\Delta v^{acceptable}_{behaviour}$, $\Delta v^{max}_{behaviour}$, $\Delta\theta^{acceptable}_{behaviour}$ and $\Delta\theta^{max}_{behaviour}$ should be defined widely in order to

avoid convergence over track's estimation, e.g. $\Delta\theta^{acceptable}_{behaviour} = 45°$ and $\Delta\theta^{max}_{behaviour} = 120°$ (as depicted in Figure 4.12).

Figure 4.13 displays an example of the improvements obtained from using the proposed velocity feedback. The result of the DOG obtained without velocity feedback is partially shown at the right of the Figure and can be completely seen in Figure 3.12. As can be noticed, in the baseline grid, the dynamic estimation of the truck in the opposite lane has not totally converged yet: some of the cells are wrongly classified as static due to a Mahalanobis distance close to zero—caused by a non-homogeneous velocity distribution, see Section 3.6.3.4. Also, the estimation of the right guardrail presents some wrong dynamic cells "moving" in the same direction as the ego-vehicle. On the contrary, when including the velocity feedback, better results are obtained in both cases. Most of the cells modeling the guardrail are correctly identified as static. While all the cells corresponding to the truck are classified as dynamic (low velocity variance) and are correctly modeling the trucks' displacement.

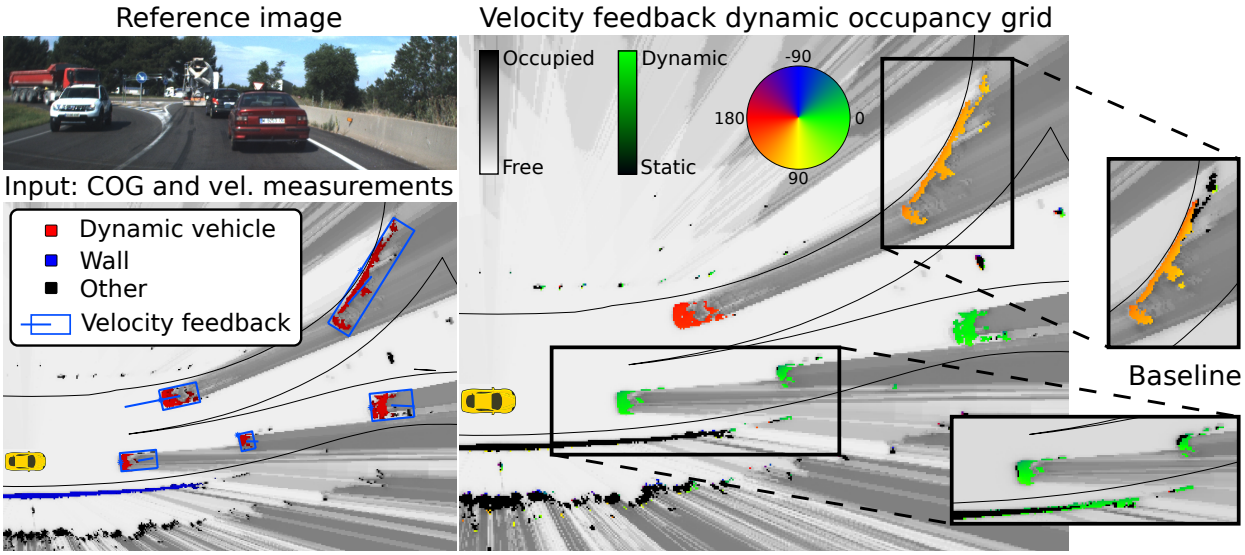


Figure 4.13: Velocity feedback example. The result of computing the DOG without velocity input data (baseline) is displayed at the right of the image.

4.8. Evaluation results

This section presents the set of experiments performed to validate the proposed COG and velocity feedback. First, the capabilities of the COG in different urban scenarios are shown. Then, the improvements obtained with the use of the velocity feedback are analyzed qualitatively and quantitatively. Lastly, the results of the object-level tracking are briefly displayed.

4.8.1. Classified Occupancy Grid results

To illustrate how the COG works, a simple example consisting of a start-moving maneuver is shown in Figure 4.14. At first, the detected vehicle is stopped (t = 1). The size and location with respect to the road features are met for the *dyn. vehicle* class, but not the velocity feature. Therefore, it is classified as *other* by the COG. Once the vehicle starts moving (t = 11) the *dyn. vehicle* class likelihood increases and the COG progressively labels the detected vehicle as *dyn. vehicle*—the gradual increase in the red color brightness from (t = 11) to (t = 19). Recall that the smoothness of the label change depends not only on the class likelihoods but also on the confidence variable $\alpha_{z_k}^{obj}$, which should be defined in order to find a balance between noise filtering and speed responsiveness.

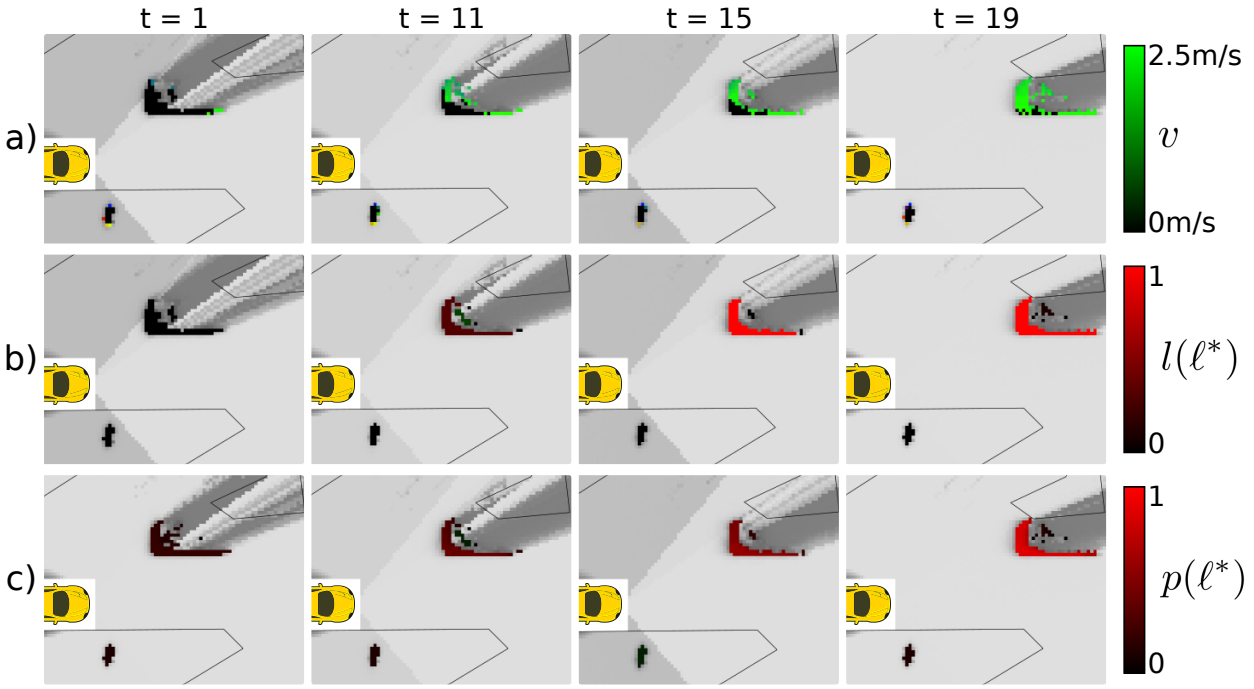


Figure 4.14: COG functioning example. a) DOG. b) Measurement COG. c) COG.

To expose the capabilities of the COG, a scenario with different objects and dynamic states is selected. The obtained results are compared with the outcome of (i) a common dynamic cluster segmentation approach and (ii) the measurement COG, Figure 4.15.

The dynamic cluster segmentation method is the typical object extraction approach used in DOGs. Objects are clustered using the approach proposed in 4.3.2. Subsequently, clustered obstacles with a significant speed are selected. This method yields acceptable results since most dynamic obstacles are correctly identified. However, it entirely depends on the good dynamic estimation of the DOG and the defined velocity threshold. Therefore, wrongly estimated dynamics lead to incorrect clusters, such as the obstacle modeled at the position of a van (1) or the left wall (2) which are both estimated as dynamic.

The measurement COG relies on more object features than the classic dynamic cluster segmen-

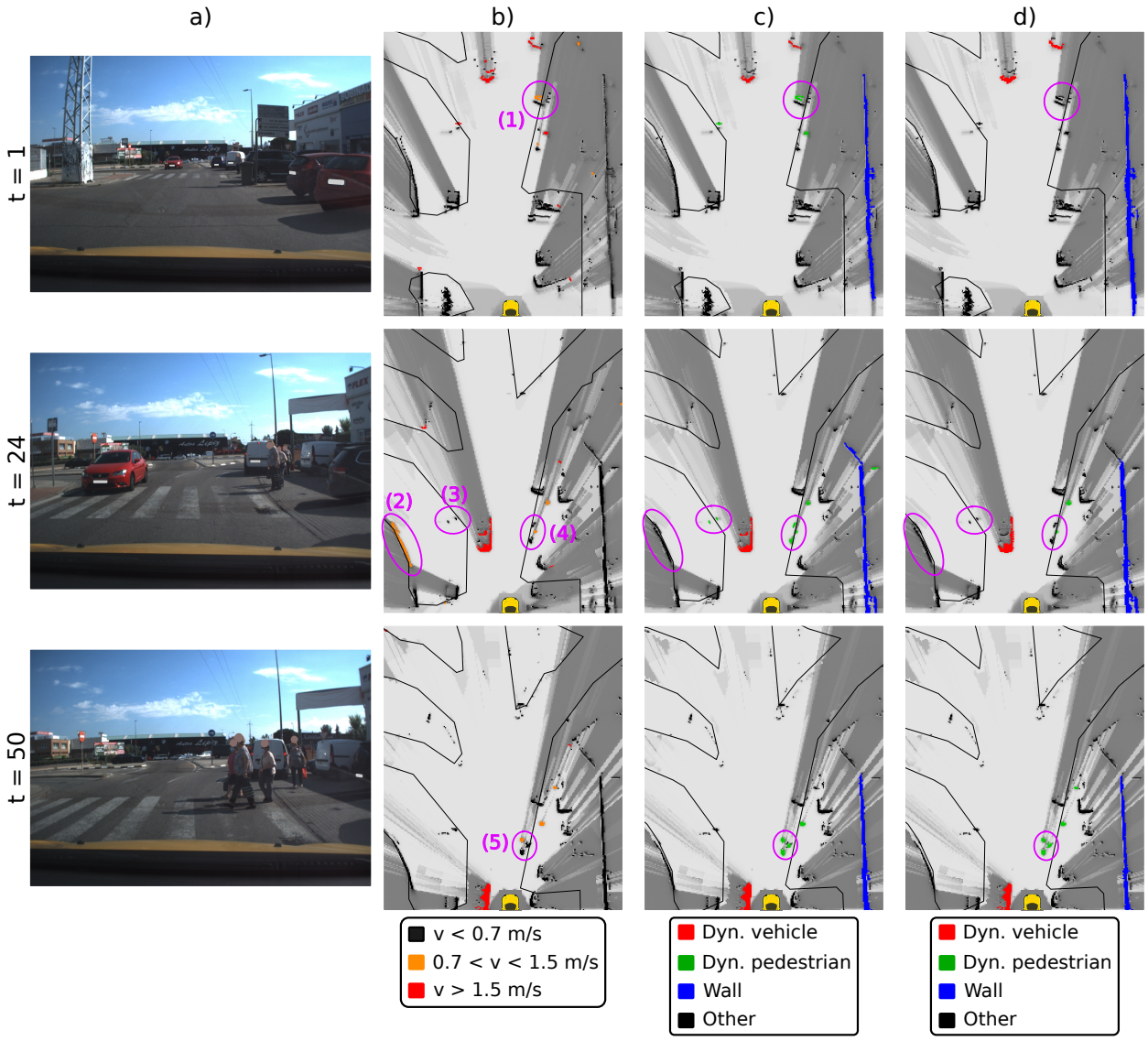


Figure 4.15: COG evaluation in urban scenario. a) Reference camera image. b) Dynamic cluster segmentation. c) Measurement COG. d) COG. Dynamic cluster segmentation is colored with respect to clusters' velocities.

tation method just presented, thus some noise is avoided, e.g. the incorrect dynamic wall (2). Moreover, by addressing different classes and not a binary segmentation, specific feature values can be defined for each class. For example, the velocity threshold for class *dyn. pedestrian* is set close to zero and, hence, pedestrians starting to cross are correctly detected—circles number (4) and (5). Nevertheless, despite this class-specific search, some objects are still incorrectly classified – circles number 1 and 3.

These two approaches, dynamic cluster segmentation and measurement COG are single-frame detectors and, therefore susceptible to degradation with isolated noise. In contrast, the COG achieves smoother label changes. This can be clearly noticed in the pedestrians crossing the crosswalk (5). In

the second row, the COG has not yet converged over the labels *dyn. pedestrian*, but in the third row all pedestrians are correctly modeled. Despite this convergence dependency, the filtering allows to correctly avoid the incorrect *dyn. pedestrian* classification made by the measurement COG at the locations of the van (1) and traffic signals (3).

4.8.2. Results of the Dynamic Occupancy Grid with velocity feedback

The benefits of including velocity input data computed with the object-level feedback method proposed are exposed with the following experiments. For the sake of readability, in the following, the DOG fed with this feedback will be denoted as Velocity Feedback-Dynamic Occupancy Grid (VF-DOG) and the DOG computed only with LiDAR measurements as Baseline-Dynamic Occupancy Grid (B-DOG).

Figure 4.16 shows a scenario with two complex situations for the dynamic estimation of the grid map: (i) the guardrail whose real shape is detected as the ego-vehicle moves ahead, circle tagged as (1), and (ii) the truck entering the FoV, circle tagged as (2). In both cases, the VF-DOG provides a better dynamic estimation.

The guardrail is modeled as a dynamic obstacle by the B-DOG since its real size is larger than the perception framework's FOV, thus, new occupied space is constantly modeled as the ego-vehicle moves ahead. In this way, particles moving with the same velocity as the ego-vehicle are the best explanation for the presence of this constantly detected obstacle. On the contrary, the VF-DOG is able to estimate the guardrail as static since the COG identifies it as a large off-road obstacle, and therefore the velocity feedback back-channels velocity estimations centered on zero.

The truck is a new object entering the sensors' FOV thus, the detected size changes with time. Moreover, as the ego-vehicle moves ahead, the occlusion caused by the guardrail is reduced, allowing the detection of the rear of the truck. In this case, different particles' behaviors can explain the occupied space modeled. This can be noticed in the non-homogeneous dynamic estimation of the cells belonging to the rear of the truck. In the VF-DOG approach, the calculated velocity feedback data approximately models the real displacement and, thus, a faster and more homogeneous representation is achieved.

A quantitative evaluation comparing the results of the B-DOG and the VF-DOG is performed using the AUTOPIA's autonomous vehicles prototypes (see Appendix A.1). Several tests are recorded, in all of them, the sensorized vehicle follows the detected vehicle through different trajectories. The total duration of the tests sums up to 12836 iterations, approximately 17 minutes. In order to evaluate the results, all particles within the ground truth vehicle box are clustered and the errors obtained for the weighted mean velocity module and orientation are calculated. Since the velocity orientation for static objects is meaningless, the orientation error is only computed for those iterations where the sensed vehicle drives faster than 0.5 m/s. Table 4.1 displays the obtained results demonstrating that the inclusion of the proposed object-level velocity feedback is able to

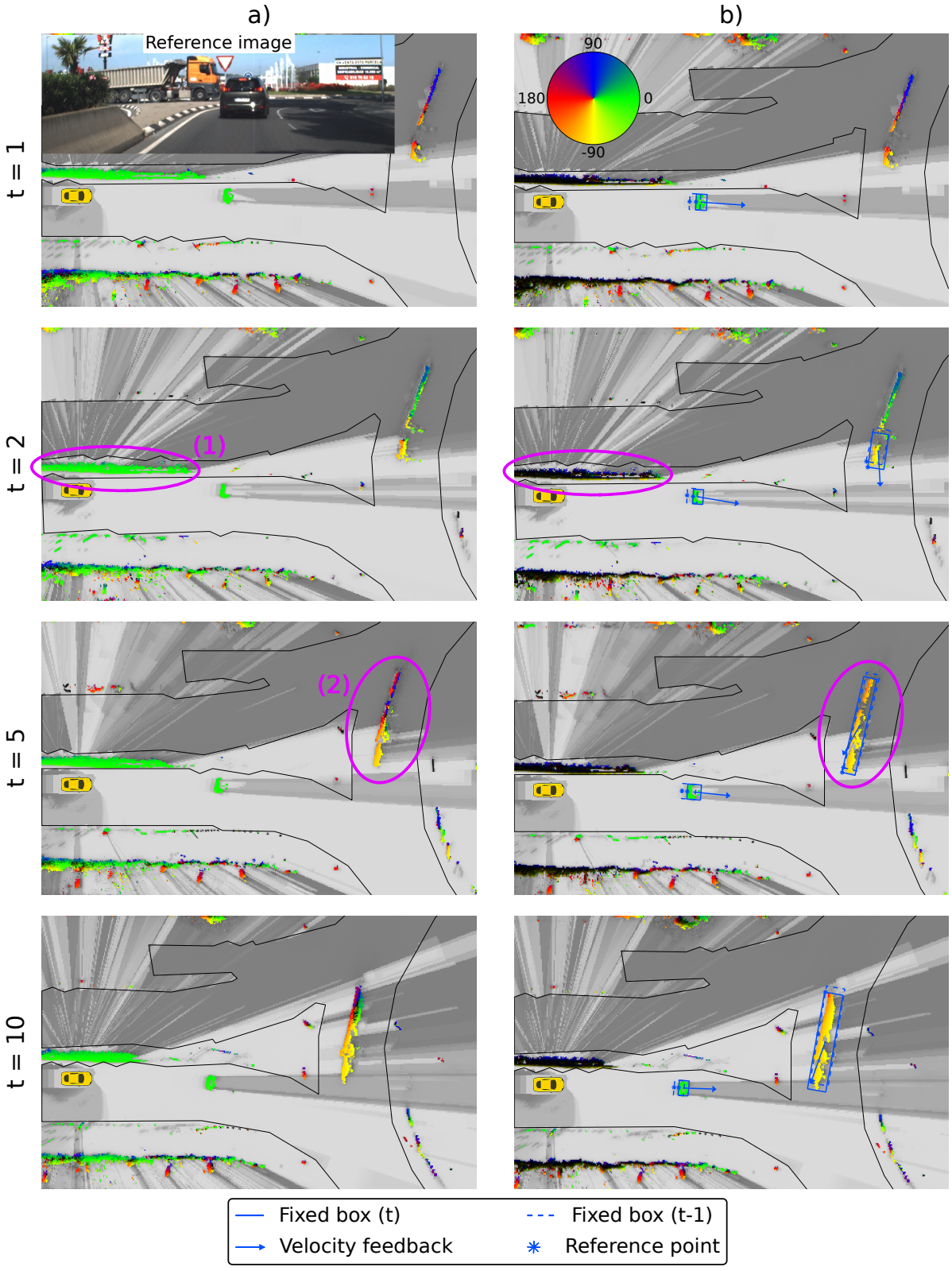


Figure 4.16: Comparison between B-DOG a) and VF-DOG b). In order to show convergence towards zero velocity, the Mahalanobis distance criteria for static-dynamic classification is not applied in this case.

enhance the dynamic estimation of the DOG.

Table 4.1: VF-DOG quantitative evaluation. Mean Absolute Error (MAE). Root Mean Square Error (RMSE).

	Velocity [m/s]		**Orientation [°]**	
	MAE	**RMSE**	**MAE**	**RMSE**
B-DOG	0.346	0.427	3.434	4.778
VF-DOG	0.165	0.242	2.279	3.413

With the intention of clearly exposing when the velocity feedback provides an estimation enhancement, Figure 4.17 exposes the results of a single test scenario, divided by stretch type. As can be appreciated, the VF-DOG provides an improvement over the B-DOG, mostly in turning situations, i.e., dynamic changing situations. For the straight stretches, both grids provide similar results. While, for stretches where the sensed vehicle is turning, the VF-DOG provides a remarkably better estimation, particularly regarding the orientation estimation.

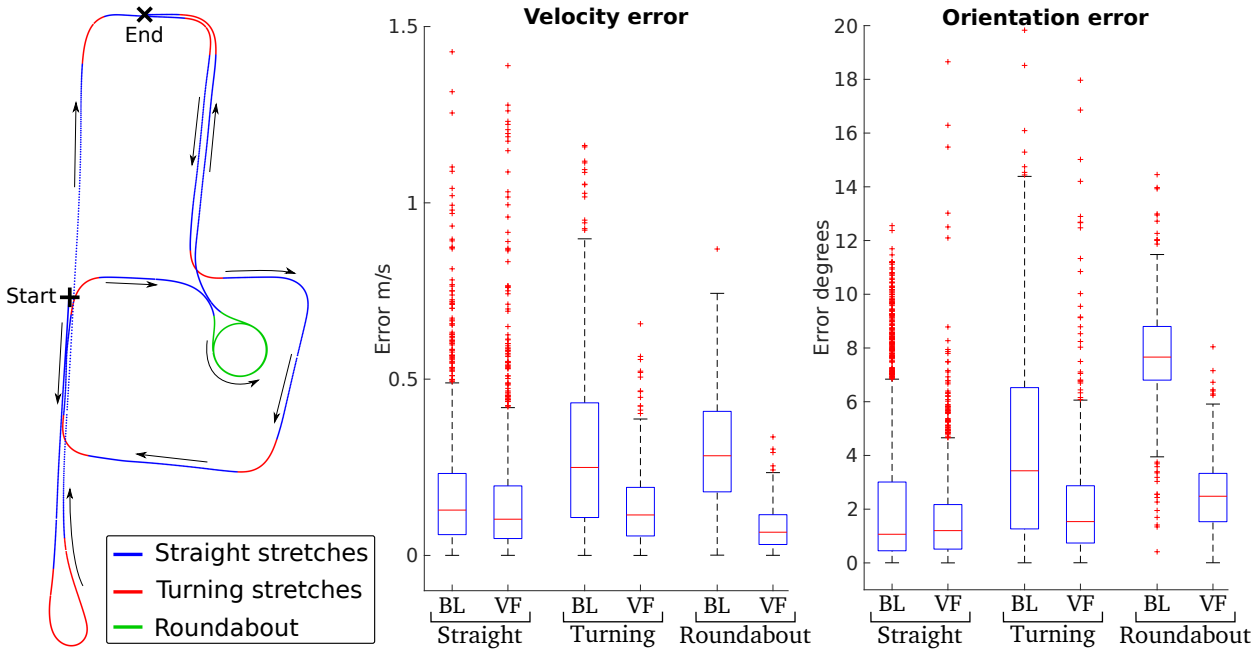


Figure 4.17: Velocity feedback quantitative evaluation separated by stretches. Baseline (BL) and velocity feedback-based (VF) DOGs.

This performance difference is illustrated in Figure 4.18, where a snapshot of the roundabout section is shown. For the B-DOG, the cells corresponding to the side of the vehicle correctly model the vehicle orientation, but those on the rear model a straight trajectory. This effect occurs because, in the B-DOG, only the occupancy update step is performed. This update allows particles with any dynamic behavior as long as they are located within a cell with occupied evidence. Moreover, in this case, the vehicle self-occludes the space ahead of it, hence particles moving straight cannot be dismissed by free space measurements. This wrong estimation is translated to the object-level features as illustrated by the velocity vector and velocity-based box. Contrarily, the outcome of

the VF-DOG shows a more homogeneous cell orientation estimation and less delay in the velocity vector. This is thanks to the velocity-based update and particle initialization performed using the velocity data approximated with the velocity feedback step.

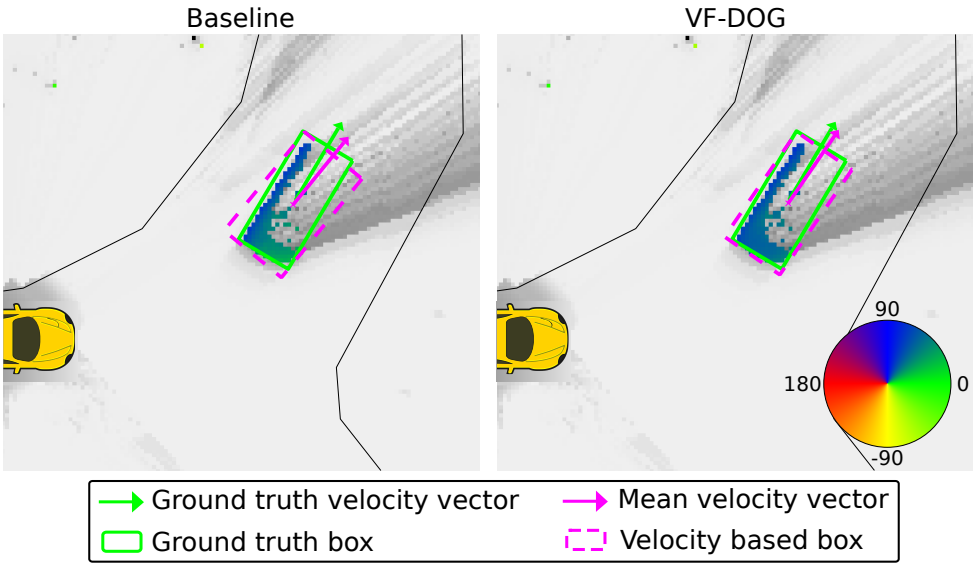


Figure 4.18: Example of the dynamic estimation obtained from the B-DOG and the (VF-DOG) at the roundabout.

Lastly, Figure 4.19 displays the velocity feedback data calculated during the roundabout stretch and compares the mean velocity vectors obtained from the B-DOG and VF-DOG with the ground truth data. It can be seen that the calculated velocity feedback presents a non-negligible error. However, it provides an approximation centered on the real velocity values that is capable of correctly modeling dynamic behavior changes. Therefore, it leads to a consistent enhancement over the B-DOG's dynamic estimation.

4.8.3. Combined grid/object-based environment representation results

This section introduces the benefits of using a hybrid environment representation to describe the surrounding scene, emphasizing how both complement each other.

Figure 4.20 shows different iterations of a common urban scenario of the AUTOPIA dataset. Both representations are displayed; the DOG along with the estimated extracted objects and the COG coupled with the road users object-level tracking.

Good results are obtained overall for both representations. The DOG is able to obtain an accurate velocity estimation of the cells with few dynamic false positives mostly found in the vegetation while the COG is able to correctly identify the vehicles and the guardrail at the right of the road. Regarding the object-level tracking, although there are multiple obstacles in the scene, including some dynamic false positives, all road users are accurately identified. This is thanks to the COG, which detects the cells corresponding to vehicles, thus, enabling that object-level tracking new

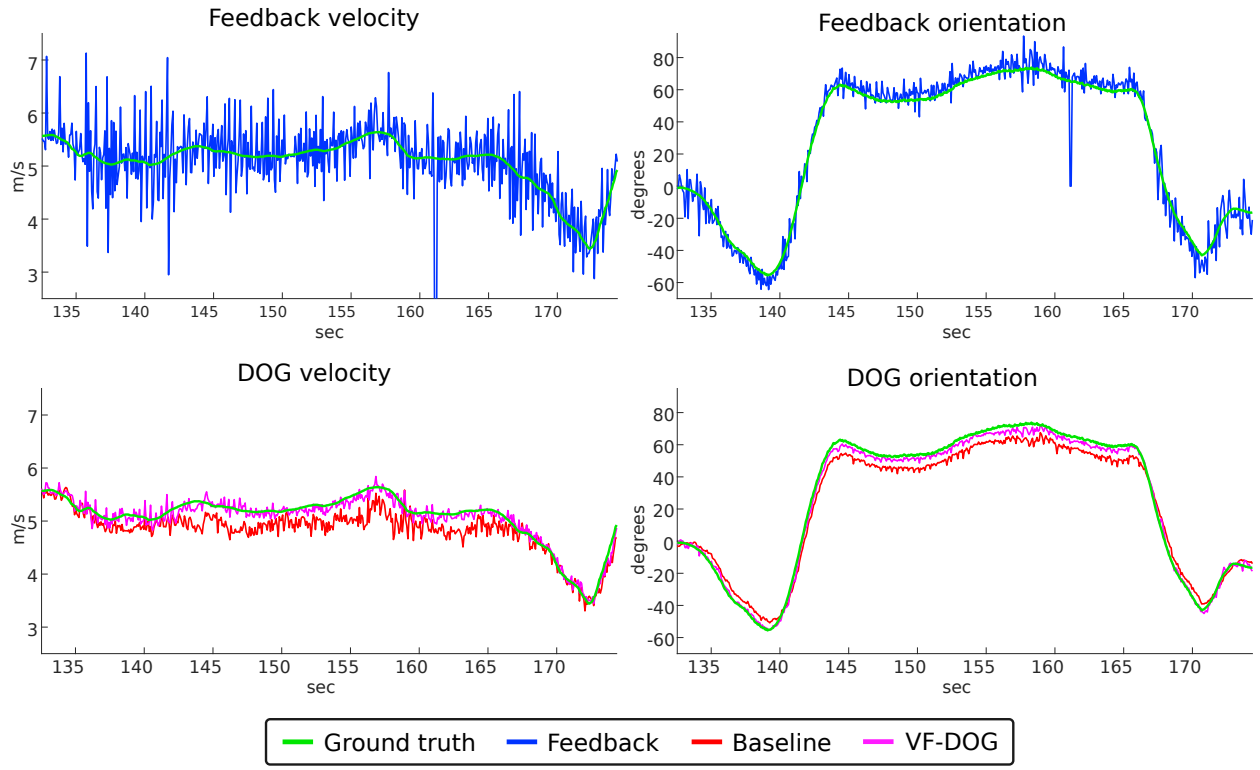


Figure 4.19: Velocity feedback data and velocity estimated by the DOGs in the roundabout stretch denoted in Figure 4.17.

hypotheses are only generated for these cells. Moreover, the objects extracted from the grid in each iteration are adequately computed in terms of clustering, velocity, and bounding box, leading to accurate object-level tracking results. This can be appreciated by the IDs assigned, which remain constant over time, and the accurately estimated boxes and velocity vectors.

Also, it can be noticed that the combination of both representations, grid-based and object-based, provides a more robust and detailed description of the surrounding environment:

- **Grid-based representation:** The grid-based module provides a broad description of the environment. All the surrounding environment is represented in terms of occupancy, dynamics and object class. Particularly, the estimation of free and unknown areas is valuable information that the object-level module cannot obtain accurately. Moreover, the perceived shapes of the different objects are accurately modeled despite their complexity. This is an ability that the object-level module's box-based model does not possess.
- **Road users tracking:** The grid-based representation can be too general and, to some extent, challenging to handle. On the contrary, the object-level tracking focuses exclusively on the dynamic road users, which are considered the objects that involve higher risk and level of interaction. Therefore, they are identified over the rest of the objects, classified, described with model-based shape and state, and assigned with a unique ID constant over time. Also,

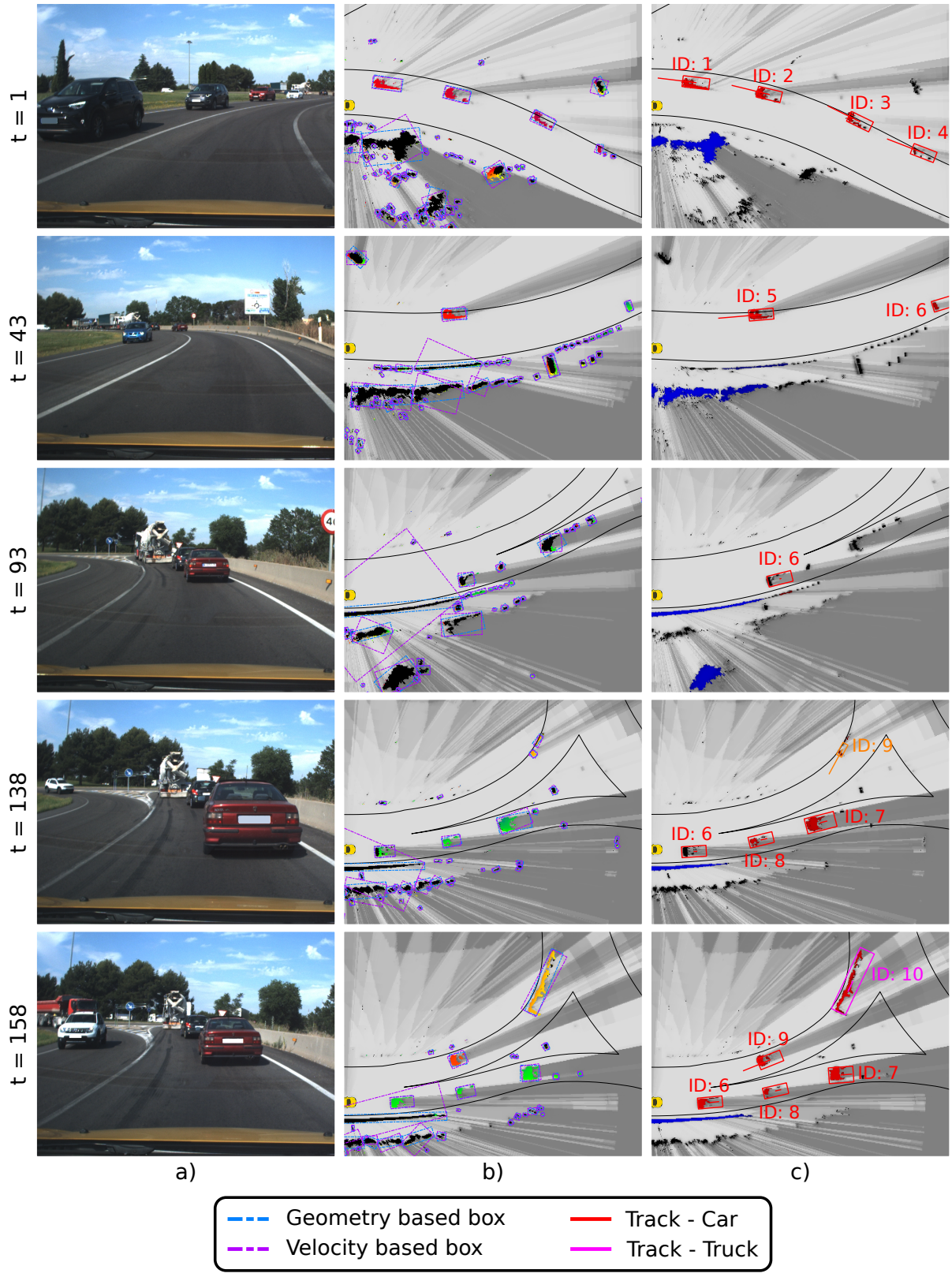


Figure 4.20: Object-level tracking evaluation in urban scenario. a) Reference image. b) DOG and bounding boxes computed for extracted objects. c) COG and object-level tracking.

unlike the grid-based representation, road users are described as single entities rather than a collection of independent cells. Furthermore, object-level assumptions help to improve the estimation and compensate for the lack of information, e.g. obtaining a better calculation of the position and area covered by the vehicles.

An example of the usefulness of this combination is illustrated by the vehicles stopped at the entrance of the roundabout. Initially (t = 93), these are not represented at object-level since they are static. However, they are correctly represented in the grid map, denoting that the lane is occupied and, thus, the ego-vehicle cannot drive along it. Once they start moving (t = 138), their state is variable and, hence, it is more challenging to interact with them. Therefore, they are identified as individual relevant objects and additionally estimated at object-level, obtaining a better description of their shape and the representation format required by some of the subsequent modules, e.g. motion prediction module.

Another example is the objects' shape representation. Complex shapes are not correctly represented at object-level, as can be clearly seen in the boxes estimated for the clusters concerning the vegetation at the right of the road. The grid, however, is able to represent these shapes accurately thanks to its discretization of the space into small cells. On the other hand, the shape of road users is correctly approximated by a box-based model. In cases of reduced visibility, the grid is only able to represent the perceived side, whereas, the object-level tracking is capable of completing the remaining unperceived shape, thanks to the boxes predefined for each type of vehicle.

4.9. Summary and conclusions

As introduced in Section 1.2.3, the object-based module is intended to (i) provide an object-level representation of the road users in the scene and (ii) infer velocity and semantic input data for the grid-based module.

This chapter has introduced the development of an object-level tracking strategy that provides the desired object-level representation. Additionally, two novel methods to back-channel object-level information to the grid-based module are proposed seeking to enhance its performance and extend its level of description of the surrounding environment. This work led to two publications, one regarding the velocity feedback method [59], and another extending it and also presenting the COG and the object-level module [57].

The object-level tracking method implemented is based on several works of the state-of-the-art and, therefore, follows a common strategy of detection, association, and filtering. The results achieved are accurate and aligned with the requirements of the subsequent modules consuming this environment representation. Moreover, the different algorithms implemented during its development have also enabled the adequate development of the object-level feedback-based steps.

Taking advantage of the extracted objects and the PF of the DOG, the grid-based module has been extended with the development of a COG that estimates the probability that the occupied

cells correspond to relevant object classes. Furthermore, its usefulness has been demonstrated through the incorporation of this classification information in the object-level tracking, enhancing some of its steps. In comparison to other semantic grid approaches, the classification is limited to the occupied space, but no image input or training data are required. Regarding object-level classification approaches, it relies on the existence and dynamic behavior of particles, but it is not dependent on the correct performance of an object-level tracking algorithm. Moreover, complex steps such as initialization and matching are solved in an easier and more reliable way.

Similarly, leveraging on the extracted objects, the matching of the object-level tracking and the COG, velocity data regarding relevant objects is approximated at each iteration. This velocity information is used as input data to update the dynamic state of the DOG. Its development has allowed the enhancement of the velocity estimation in different challenging situations where using only LiDAR data does not provide the desired accuracy.