2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Object Detection on Dynamic Occupancy Grid Maps Using Deep Learning and Automatic Label Generation

Stefan Hoermann, Philipp Henzler, Martin Bach, and Klaus Dietmayer
Institute of Measurement, Control, and Microtechnology, Ulm University, Germany

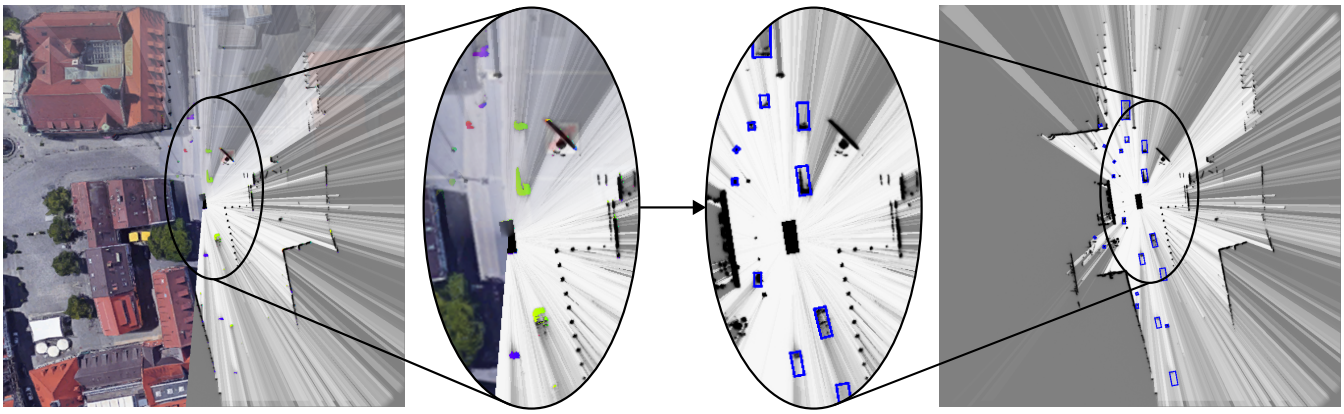

Fig. 1: Object detection using deep learning and grid fusion.

***Abstract*— We tackle the problem of object detection and pose estimation in a shared space downtown environment. For perception multiple laser scanners with 360° coverage were fused in a dynamic occupancy grid map (DOGMa). A single-stage deep convolutional neural network is trained to provide object hypotheses comprising of shape, position, orientation and an existence score from a single input DOGMa. Furthermore, an algorithm for offline object extraction was developed to automatically label several hours of training data. The algorithm is based on a two-pass trajectory extraction, forward and backward in time. Typical for engineered algorithms, the automatic label generation suffers from misdetections, which makes hard negative mining impractical. Therefore, we propose a loss function counteracting the high imbalance between mostly static background and extremely rare dynamic grid cells. Experiments indicate, that the trained network has good generalization capabilities since it detects objects occasionally lost by the label algorithm. Evaluation reaches an average precision (AP) of $75.9\,\%$.**

I. Introduction

On the path to autonomous driving, a thoroughly modeled environment is essential for high-level software modules like behavior and trajectory planning [1]. Two environment representation strategies are commonly used: object-model-based and object-model-free. We refer to an object model in terms of a vector containing the size and pose, while dynamics, existence probability and the according covariance matrix can also be included. *Object-model-free* grid maps gained great success fusing raw sensor data in one environment representation. The task of object hypotheses generation is intentionally avoided [2]–[4]. Instead, grid fusion aims at estimating the occupancy probability and dynamic states at independent, discretized locations in the environment. Thus, the strength of multiple sensors can be fused in a single dynamic occupancy grid map (DOGMa) [5], without the need to decide what object type caused the measurement. However, modeled objects are fundamentally required for many applications such as decision-making [6], [7]. *Object-model-based* tracking aims for extracting moving objects represented as a list or distribution of object vectors [8], [9], while stationary objects are widely ignored. Initializing objects and associating measurements is one of the most critical tasks in object tracking. While sophisticated shape models were designed to associate measurements to objects [10]–[12], background models are mostly rudimentary, e.g. assuming uniform distributed clutter measurement. Consequentially, objects are falsely detected as positive. Convolutional neural networks (CNNs), alternatively, are known for their capability to exploit context information, or in other words, establishing an intrinsic background model.

In this work we present a learning based approach to finding objects in terms of width, length, orientation, and position in a DOGMa as illustrated in Fig. 1. The left side depicts a top view satellite image from Google Maps with a fading overlay of a DOGMa. The right side depicts detected objects as blue rectangles.

Whereas the extensive task of manual labeling in learning applications is a main drawback, we propose a fully automated approach to extract labels of moving objects. By collecting object data over time and feeding gained information back to earlier time steps where important information like the object shape wasn't observed yet, an acausal object extraction algorithm is presented for offline label generation.

978-1-5386-4451-5/18/$31.00 ©2018 IEEE

The remaining paper is organized as follows: Related work is reviewed in Section II. Details about dynamic occupancy grid maps, used as single input to a deep convolutional neural network (CNN), are given in Section III. An object extraction algorithm used to generate hours of training data without manual labeling is introduced in Section IV. Section V explains the network architecture and its output which is based on the concept of 'anchors', where the network predicts the best fit within a set of default bounding boxes plus the offset to the true object. Section VI proposes a loss function, particularly adapted to the extremely imbalanced character of the data. Subsequently, the data itself is examined in Section VII. Experiments, showing the precision recall behavior of the detection network as well as the bounding box error objects, are carried out in Section VIII, followed by conclusions in Section IX.

II. Related Work

A common object tracking approach unites raw measurements by box fitting and tracking these boxes considered as single pseudo measurements. While sensor fusion and tracking approaches are advanced and theoretically supported, object detection or initialization is reasonably engineered. Hand engineered object detection based on box fitting with L-shapes in laser [11], [13] or radar measurements [12], suffers from limiting assumptions and simplifications regarding sensor, object, and environment features. Commonly, heuristic parameter tuning is required, e.g. to adjust the measurement noise and clutter assumptions. A data driven alternative to box fitting is proposed by Scheel and Dietmayer [14], where the radar measurement model of a car is learned and can be probabilistically conditioned on the perspective. So far, however, the aforementioned approach only focuses on cars.

Detecting objects in grid maps widely decouples sensor fusion from object tracking, i.e. performing object detection after dynamic grid mapping. To generate spatially extended object models, highly engineered methods were proposed to find cell clusters representing an object [15]–[17]. Experiments showing extracted and tracked objects seem promising, however, the engineered initialization requires easy separable cells with small velocity variance. Generally, object detection in grid maps suffers from corrupted object silhouettes, occlusions, and false velocity estimates in static regions. We claim, that a CNN can deal with these cases.

A DOGMa provides a neural network friendly representation of the entire environment and its dynamics. Piewak et al. [18] trained a neural network to reduce false positive velocity estimation in a DOGMa sequence. In particular, their approach refers to a pixel-wise classification task to determine whether a cell in a DOGMa is dynamic or static. Yet, clustering is still necessary to obtain objects. In contrast, our approach directly predicts objects with position, shape and orientation. In our previous work [19], a network was trained to separate static regions in a DOGMa and predict future cell occupancy caused by dynamic regions. Similarly, Dequaire et al. [20] propose an end-to-end recurrent neural network for object tracking. Again, their approach infers pixel states while our results, in this work, are bounding boxes.

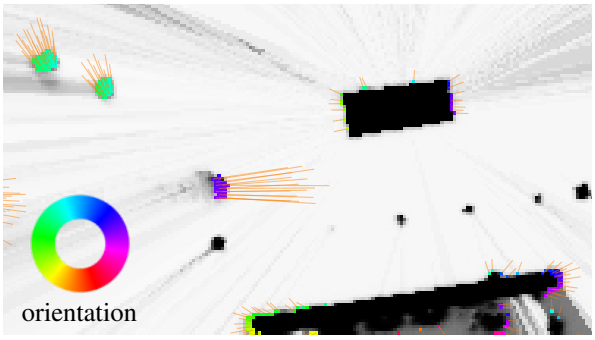

Fig. 2: Excerpt of a DOGMa. Orange lines indicate the estimated cell velocity, while the cell color indicates the movement orientation. Grayscale indicates the cell occupancy probability P_{O}.

Object detection in general, or more specifically in images, is a very active research area. Recent approaches like Fast R-CNN [21], Faster R-CNN [22], or Mask R-CNN [23], are based on two stages. The first stage delivers a region of interest (ROI) in an image, e.g. provided by a Region Proposal Network (RPN) [22]. The second stage is applied on the ROIs to predict a classification and ROI offset. The recent Mask R-CNN [23] implements a similar two-stage procedure but adds a binary mask output for every ROI for additional object segmentation.

Other approaches, like YOLO [24] and SSD [25], employ a deep convolutional neural network (CNN) only in a single stage. The former tries to infer object bounding boxes by regression, which however lacks of position accuracy and couples bounding box regression with classification. On the other hand, SSD follows the strategy of anchors, i.e. default boxes defined by aspect ratio and scale, to find bounding boxes of classified objects. In addition to classifying default boxes, the shape offset is predicted by the neural network for each default box. To counteract the imbalance in the data with respect to default boxes (most boxes have negative decision), only the top 3 negative decisions are sampled during training. We also follow the approach of default boxes but additionally add object orientation as an attribute of an anchor. An optimization approach is used on a large dataset, to choose the set of anchors.

Focal Loss [26] investigated the great success of two-stage approaches and found, that the extreme imbalance between background and object pixels was compensated by the ROI extraction. Using a novel loss function, which considers this imbalance implicitly, enabled a one-stage network to gain even better performance. We also faced this problem in our previous work [19] when dealing with the high imbalance between static and dynamic cells in a DOGMaand proposed a pixel balancing loss function which we adopt in this work.

III. Filtered Dynamic Input

We use the DOGMa from Nuss et al. [5]. Fig. 2 shows an excerpt of a DOGMa created from multiple laser scanners. The perceived environment is spatially discretized in grid cells

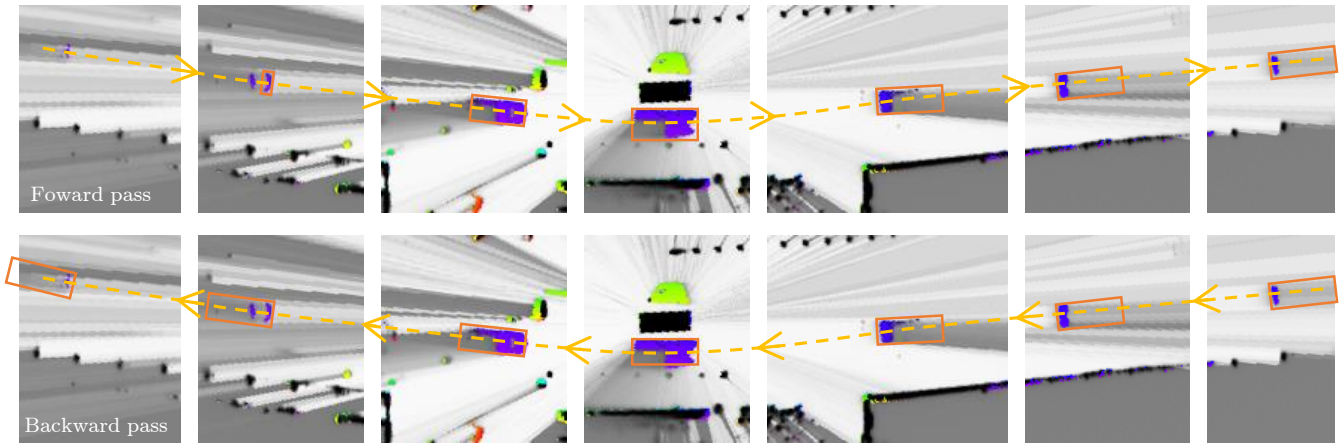

Fig. 3: Schematic of the two-pass label extraction algorithm: In the forward pass, well detectable objects are tracked through the sequence. In the backward path, the object shape and pose is refined and even a corrupted object silhouette can be used to fit the object bounding box.

c at positions (E, N), indicating east and north, respectively. A particle-filter-based velocity estimation augments the classical static occupancy grid with dynamic states. Occupancy estimation is based on Dempster Shafer [27]. A cell contains the channels $\Omega = \left\{M_\mathrm{O}, M_\mathrm{F}, v_\mathrm{E}, v_\mathrm{N}, \sigma^2_{v_\mathrm{E}}, \sigma^2_{v_\mathrm{N}}, \sigma^2_{v_\mathrm{E},v_\mathrm{N}}\right\}$ with Dempster-Shafer masses for free space $M_\mathrm{F} \in [0,1]$ and occupancy $M_\mathrm{O} \in [0,1]$, the velocity pointing east v_E and north v_N, as well as the velocity variances and covariance. The occupancy probability is calculated by $P_\mathrm{O} = 0.5 \cdot M_\mathrm{O} + 0.5 \cdot (1 - M_\mathrm{F})$ where a high P_O refers to a dark pixel in Fig. 2. $P_\mathrm{O}(\mathrm{E}, \mathrm{N}, t) := P_\mathrm{O}(c, t)$ denotes the occupancy probability at a grid cell c and sequence time step t. The DOGMa data is provided in $\mathbb{R}^{W \times H \times |\Omega|}$ with the spatial width W and height H pointing east and north, respectively.

A key assumption for efficient processing is the independence of single cells. In consequence, as observable in Fig. 2, borders of walls and static cars tend to have false velocity estimates. This causes simple clustering to fail, while a convolutional neural network can be trained to consider context.

IV. Automatic Label Generation

Highly engineered object detection in grid maps, i.e. estimation of position, width, length and orientation, is examined in literature [15], [17]. The algorithms are relatively restricted due to application constraints, e.g. real time processing. Our application, in contrast, is offline label extraction which allows advantageous reduction of restrictions. For example, the algorithm ignores causality: it runs forward and backward in time, and makes use of preprocessing, e.g., spatial and temporal data smoothing using multidimensional Gaussian kernels. Furthermore, post processing is used to refine trajectories and identify outliers on a pixel and trajectory level. Like most engineered algorithms, a relatively high false negative rate can be assumed, compared to human object detection. However, considering these circumstances during network training, utilizing a learning approach could lead to better generalization. Hence, endless labeled sequences are imaginable.

Fig. 3 illustrates the extraction of a vehicle. The extracted bounding box is drawn in orange at different time steps, while the cell cluster appears purple. The vehicle approaches and passes the ego-vehicle, which appears as a black, filled rectangle in the center. The purple object silhouette is corrupted due to (self) occlusion and particle filter convergence. When objects enter the field of view, they are not visible very clearly in a DOGMa, and their silhouette grows when they get closer to the ego vehicle. When the object passed the ego-vehicle, the visible object silhouette shrinks. Successively, the front, side and back of the vehicle are visible and exhibit a rectangular object shape. In the forward pass (top row in Fig. 3) object tracking is initialized when it is very certain that a cell belongs to a moving object, i.e. high $P_\mathrm{O}(c,t)$, low velocity variance, and high velocity magnitude. After the vehicle leaves the field of view, the backward pass is initialized (bottom row), refining the object pose and extent, and detecting objects in time steps before the object was initialized. At each time step in the sequence, an object corner point expected to be visible, named reference point, is found considering object orientation, size, position, as well as occlusions in the line of sight to the rectangle corner points. Thus, a bounding box can be constructed starting from the reference point even at corrupted or partially occluded object silhouettes in a far sensing region.

The silhouette clustering is straight forward, based on connected components, i.e. connected cells with similar P_O and velocity. However, some extensions are described in the following. To limit cell clusters, boundary cells are calculated ideally limiting the object silhouette at a rise or slope of the smoothed occupancy probability $P_\mathrm{O}(\mathrm{E}, \mathrm{N}, t)$. For this, the first and second spatial derivative of $P_\mathrm{O}(\mathrm{E}, \mathrm{N}, t)$ is calculated to find inflection points. This is in particular useful when objects are close to other objects or static regions. The found object silhouette is predicted to the next DOGMa time step using velocity statistics from the spatial velocity distribution as well as the velocity covariances of single cells. Cells

covered by the predicted silhouette and fitting best to the velocity profile are chosen to start a new connected component search. The number of start cells is scaled by the expected object silhouette size to include about 1 cell per $0.5\,\mathrm{m}^2$. It is assumed, that the new connected component contains outliers. Therefore, velocity statistics of n inlier cells with the least velocity variance and highest P_O are chosen to establish new object cell statistics. Remaining cells of the connected component are considered as outliers if they are outside a 2σ bound. n is the number of cells included in the previous extracted silhouette if the silhouette grows, or half of new initial cells otherwise.

In post processing, the extracted trajectories are smoothed using spline fitting. Trajectories with unreasonable motion are rejected. Furthermore, open street map [28] is used to eliminate static areas falsely detected as objects and mirrored objects in glass fronts of buildings.

A main drawback of the algorithm is, that if an object is lost in a late stage of the trajectory, it is hard to resume tracking. The same applies in an early stage in the backward pass. Therefore, the labeled data tends to contain more missing labels than false positives. Since we are aware of this problem, it can be considered when training the network.

V. Network Output and Architecture

We chose a simple encoder-decoder network structure with skip connections, inspired by [29]. We employ a pixel-to-pixel structure yielding equal input and output resolution. Instead of fully regressing bounding boxes, we follow the strategy to classify a limited number of rotated default boxes (anchors) and additionally regress their offset to the ground truth box. Anchors are defined by triples (w, l, ϕ), denoting the width, length and orientation, respectively. A more intuitive representation is (a, l, ϕ), where $a = \frac{w}{l}$ refers to the aspect and l can be interpreted as a scale. We refer to (a, l) as the shape. In the following, we distinguish between label and network prediction by using a $\wedge$ on top of predictions.

The network output is illustrated in Fig. 4. It is trained to produce four different outputs: The anchor score $\hat{y}^{(\mathrm{IoU})}$, the width offset $\hat{y}^{(\Delta w)}$, the length offset $\hat{y}^{(\Delta l)}$ and the orientation offset $\hat{y}^{(\Delta\phi)}$. $\hat{y}^{(\mathrm{IoU})}$ is encoded as the intersection over union between default box and true box, commonly used to compare similarity of bounding boxes [30]. $\hat{y}^{(\Delta w)}$ and $\hat{y}^{(\Delta l)}$ are

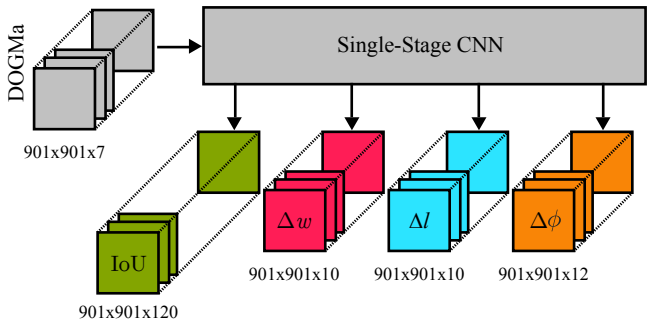

Fig. 4: Network input and output. The network predicts scores (IoU) of default 'anchor' boxes and feature offsets to the object bounding box.

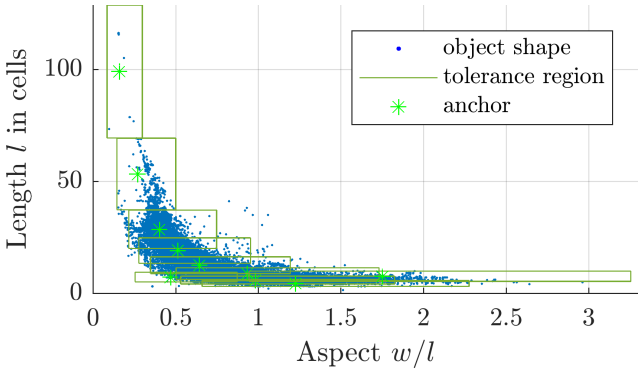

Fig. 5: Chosen anchors (green stars) over ground truth box shapes (blue dots). The rectangles illustrate the tolerance region (green rectangles).

provided relative to the anchor width and length, respectively, in order to gain similar values for all anchors. The orientation offset $\hat{y}^{(\Delta\phi)}$ is scaled to π.

With C_s default shapes and C_ϕ default orientations, $C_\alpha = C_s \cdot C_\phi$ anchors α are defined. From this follows that $\hat{y}^{(\mathrm{IoU})} \in \mathbb{R}^{W\times H\times C_\alpha}$ couples the three features l, w and ϕ in a single prediction for each grid cell $c \in \{1, ..., W \cdot H\}$. Thus, for every grid cell the score of each default bounding box is provided, assuming the cell is the center of the box. It is important to note that we chose not to train for a binary decision, but regressing the IoU between anchor and ground truth box. That way, the box fitting is essentially discretized to decide for a default box, while the decision itself is made via regression of the IoU.

An alternative to coupling box features in one prediction is to use independent box feature outputs for w, l, and ϕ. Although this could reduce the output dimension to the sum $C_w + C_l + C_\phi$, it also allows for unreasonable box results, e.g. estimating the length of a truck, the width of a bike but the orientation of a pedestrian. However, shape offset and orientation offset are assumed to be independent. In fact, the orientation offset is constant for default shapes with equal orientation. Consequently, the shape offset outputs are provided with C_s and the orientation output with C_ϕ channels, which leads to $\hat{y}^{(\Delta w)}, \hat{y}^{(\Delta l)} \in \mathbb{R}^{W\times H\times C_s}$ and $\hat{y}^{(\Delta\phi)} \in \mathbb{R}^{W\times H\times C_\phi}$. Offset regression is trained for all default boxes, no matter if the according anchor box fits best or not at all to the ground truth object.

A. Anchor Selection

We acquired numerous rotated rectangle labels with width w, length l and orientation ϕ. While the orientation is considered uniformly distributed, the anchor shapes should only cover a sparse set of reasonable aspects and scales. Therefore, we define anchor orientation and shape independently where $C_\phi = 12$ anchor orientations were defined equally distributed in $[0, 2\pi)$ and $C_s = 10$ default rectangle shapes were found by optimization. Anchor shape optimization aims to cover most label shapes within a preliminarily defined offset tolerance of $\delta = 30\,\%$. Since the algorithm operates in (a, l) space, we define tolerance ranges by $l_\mathrm{min} = l \cdot (1-\delta)$, $l_\mathrm{max} = l \cdot (1+\delta)$,

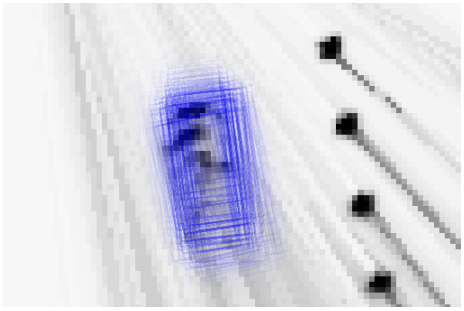

Fig. 6: Network result for object bounding boxes, interpretable as a hypothesis density. Rectangles have decreasing transparency with increasing score.

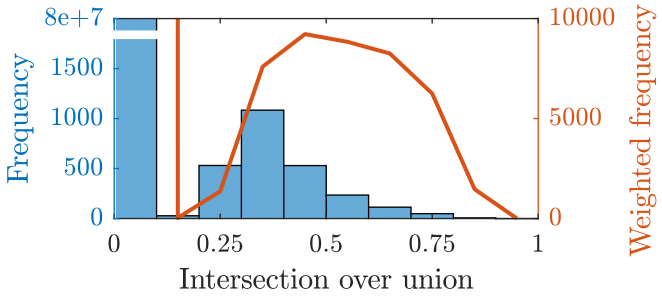

Fig. 7: IoU histogram (blue) of 100 labels $\mathbf{A}$ and the weighted frequency of IoU occurrence (red) used for balancing during training with $\lambda_{\mathrm{I}} = 400$ and $f = 4$.

$a_{\min} = a\frac{l_{\min}}{l_{\max}}$ and $a_{\max} = a\frac{l_{\max}}{l_{\min}}$. The algorithm operates in the following manner: A 2D histogram over a and l is established from ground truth boxes. Optimization for the first anchor shape is initialized at the highest peak in the histogram, varying a and l to maximize the ground truth boxes within the resulting tolerance region. Label shapes within the optimal tolerance region are removed from the histogram, and optimization for the next anchor is initialized at the next peak.

The result is illustrated in Fig. 5. Blue dots represent label shapes, the green stars represent the 10 anchors resulting from optimization and the green boxes illustrate a 30 % tolerance region.

B. Calculating Labels

The 3D label arrays $y^{(\mathrm{IoU})}$, $y^{(\Delta w)}$, $y^{(\Delta l)}$ and $y^{(\Delta\phi)}$ are initialized with 0. By iterating through labeled objects, relevant cell locations (E, N) occupied by label rectangles are filled. Cells outside label boxes are 0. For each relevant location, the IoU of all anchors α is calculated with the considered cell c as the center of the rotated rectangle. The result is stored at $y^{(\mathrm{IoU})}(\mathrm{E}, \mathrm{N}, \alpha) := y^{(\mathrm{IoU})}(c, \alpha)$. The offset labels $y^{(\Delta w)}$, $y^{(\Delta l)}$ and $y^{(\Delta\phi)}$ are filled accordingly. However, while the IoU decreases rapidly in a spatial surrounding of the true center cell, the offset labels are kept constant, since orientation and size is constant for all pixels covered by an object bounding box.

For training purposes, explained later in Section VI, we create a 2D map $\mathbf{A} \in \mathbb{R}^{W \times H}$ containing the maximum IoU for all cells c along the anchor dimension of $y^{(\mathrm{IoU})}$ by $\mathbf{A}(c) = \max_{\alpha} \left(y^{(\mathrm{IoU})}(c, \alpha)\right)$.

C. Inferring object bounding boxes

For each cell C_α, boxes can be constructed using the anchors and their predicted offset. Each resulting box comes with a score, i.e. the predicted IoU. Fig. 6 illustrates the result, where the bounding box transparency refers to the score. The normalized result can be seen as a distribution of object hypotheses.

However, for many applications a single winning box is desired. For this task, $\hat{\mathbf{A}}(c) = \max_{\alpha} \left(\hat{y}^{(\mathrm{IoU})}(c, \alpha)\right)$ is calculated and boxes enclosing a higher $\hat{\mathbf{A}}(c)$ are refused. To speed up computation, the process starts only at local maxima in $\hat{\mathbf{A}}(c)$, thresholded to a minimum score. As there might be similar anchor scores for different orientations, the four best anchors in a cell are investigated first. Among these four, the anchor $\alpha_{\max}$ with the least orientation offset $\Delta\phi_{\min}$ is considered as the winning anchor. Only the winning boxes were used for evaluation and in Fig. 1.

VI. Spatial Balancing Loss Function

We adopt the loss function from our previous work [19] with slight modifications explained below. Similar to [19] we face the problem of high imbalance between static background and dynamic objects where dynamic cells occur extremely rarely, as discussed in section VII. The necessity of counteracting such imbalance in the loss function when training a single stage neural network was thoroughly investigated by [26].

We use $\mathbf{A}$ as a spatial map to adjust weighting of the cells. In particular $\mathbf{A}(c) = 0$ for all background cells and $0 < \mathbf{A}(c) \leq 1$ for all cells that are occupied by an object. The weighting follows

$$L_y = \frac{\lambda_y}{2} \sum_c \sum_\alpha \left(1 + \lambda_{\mathrm{I}} \cdot \mathbf{A}(c)^{f_y}\right) \left(\hat{y}(c, \alpha) - y(c, \alpha)\right)^2 \tag{1}$$

where $y \in \left\{y^{(\mathrm{IoU})}, y^{(\Delta w)}, y^{(\Delta l)}, y^{(\Delta\phi)}\right\}$ and

$$L = L_{y^{(\mathrm{IoU})}} + L_{y^{(\Delta w)}} + L_{y^{(\Delta l)}} + L_{y^{(\Delta\phi)}} \tag{2}$$

is the total loss. The factor λ_y is used to mix the influence of output type, e.g. to weight the orientation offset similar to the anchor score (IoU). In the term $(1 + \lambda_{\mathrm{I}} \cdot \mathbf{A}(c)^{f_y})$, λ_{I} is used to reduce unbalancing between background and object cells. Background cells are weighted by 1, since here $\mathbf{A}(c) = 0$, while the weight of object cells is increased up to $1+\lambda_{\mathrm{I}}$. In our case, a low λ_{I} results in numerous false negative predictions, while choosing λ_{I} an order of magnitude higher than the ratio of object cells to occupied cells results in many false positives. The parameter f_y is introduced to adjust the weighing of cells within object bounds where $0 < \mathbf{A}(c) \leq 1$. Center cells with high IoU are relatively rare compared to cells with lower IoU, as illustrated in Fig. 7 where the histogram of IoU frequency among 100 samples of $\mathbf{A}$ is given. Without f_y, i.e. $f_y = 1$, the network output tends to mostly predict values at highest IoU occurrence, i.e. ≈ 0.35. A strategy to find

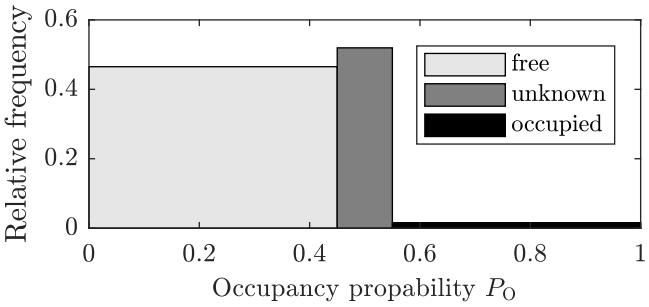

Fig. 8: Occupancy histogram of input data.

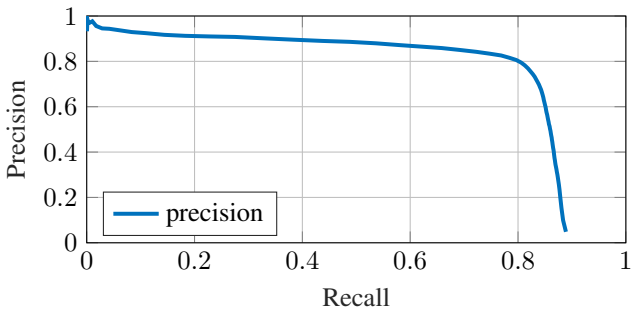

Fig. 9: Object detectio precision recall curve.

TABLE I: Bounding box error (RMSE) and average precision (AP)

position	width	length	orientation	AP
0.47 m	0.21 m	0.76 m	8.72°	0.7594

good training parameters is: first choose λ_I in an order of magnitude as the foreground to background ratio, and second choose f_y to approximate a uniformly weighted frequency of IoU occurrence. An example for weighted frequency is illustrated by the red curve in Fig. 7.

VII. Dataset and Training

We ran the automatic label generation algorithm on about 2 h recordings of an urban shared space junction with pedestrians, bikes and motor vehicles. The sequences were recorded at three different days observing the junction from east and west. We used 68927 samples for training and 1800 samples to run evaluation experiments.

The input DOGMa has spatial dimensions of 901×901 cells with a width of 0.15 m. The histogram of P_O over 100 random samples is given in Fig. 8. The ratio of occupied to free is about $1:31$, the ratio occupied to not occupied is $1:65$. The ratio of dynamic foreground cells to total occupied background cells is about $1:400$. A common training strategy, hard negative mining, is to use only a sparse set of background examples for back propagation, e.g. the worst 3 predictions, to reduce the imbalance, e.g. to $1:3$ (c.f. [24]). We, however, assume about 5 % missing labels in our dataset and therefore decided to use all cells for back propagation but employ loss balancing. This way, the effect of a missing label vanishes in the mass of correct background labels.

The histogram in Fig. 7 illustrates the extreme imbalance between $y^{(\mathrm{IoU})} = 0$ and $y^{(\mathrm{IoU})} > 0$, but also, a high imbalance between labels with $0 < y^{(\mathrm{IoU})} < 0.5$ and $y^{(\mathrm{IoU})} \geq 0.5$. To counteract the imbalance between dynamic and static cells we chose $\lambda_\mathrm{I} = 400$, according the ratio between dynamic and occupied background cells in our training data. To reduce the imbalance within dynamic cells ($0 < y^{(\mathrm{IoU})} \leq 1$), $f_{y^{(\mathrm{IoU})}} = 4$ was chosen. The resulting weighted frequency of IoU occurrence is illustrated with a red curve in Fig. 7. In contrast to the anchor score (IoU), yielding a local maximum at the object center cell, the offset labels have equal values in a surrounding of the center. Therefore, we set $f_{y^{(\Delta l)}} = f_{y^{(\Delta w)}} = f_{y^{(\Delta \phi)}} = 1$. The mixing parameters were chosen to $\lambda_y^{(\Delta l)} = 0.01$, $\lambda_y^{(\Delta w)} = 0.05$, $\lambda_y^{(\Delta \phi)} = 0.25$ and $\lambda_y^{(\mathrm{IoU})} = 1$.

The ADAM solver [31] was used for training. We chose the exponential decay rates to $\beta_1 = 0.9$ and $\beta_2 = 0.999$, as suggested in [31]. A base learning rate of 0.0001 was used. The training process was stopped after about 3.3 epochs (200000 iterations) with a batch size of 1.

VIII. Results and Evaluation

We evaluate the object detection and pose estimation performance in a crowded downtown scenario with numerous pedestrians, bikes, cars and other road users. The scene used for evaluation is illustrated in Fig 1. It was recorded by four Velodyne VLP 16 and one 4-layer IBEO LUX laser scanner. Each Velodyne provides 360° perception ranging up to 100 m at 10 Hz. The IBEO LUX runs at 12.5 Hz and has a range up to 200 m in front of the experimental vehicle with 100° opening angle. Fusing the sensors in a DOGMa covering $135.15\,\mathrm{m} \times 135.15\,\mathrm{m}$ takes about 30 ms on a GPU and is triggered at 10 Hz.

A video illustrating the object detection is made available online[1]. The network takes 66.8042 ms on a Nvidia GTX 1080ti to process one DOGMa input. The evaluation sequences cover 1800 example frames (3 minutes) including 28351 labeled objects. The precision recall curve for object detection performance is given in Fig. 9. The curve was created by varying a minimum IoU threshold γ between 0.1 and 1. A precision of 0.79 is achieved at recall 0.8, while the average precision is 0.7594. The prediction error, in terms of root mean square error (RMSE) over true positive object predictions with $\gamma = 0.55$, of bounding box features is given in table I. Please note, that for the orientation error, there is a 180° ambiguity for static objects. Therefore, the orientation error is calculated excluding 328 objects ($\approx 1\,\%$) where the error was about 180°.

Fig. 10 shows the result for an example time step where predicted objects are depicted in blue and labels in orange. It shows in particular that the network was trained not to detect objects, mirrored in high reflective building fronts. It also shows examples where the training data contains false

[1]The video under `https://youtu.be/Rr9LOrQMgKA` illustrates the network performance.

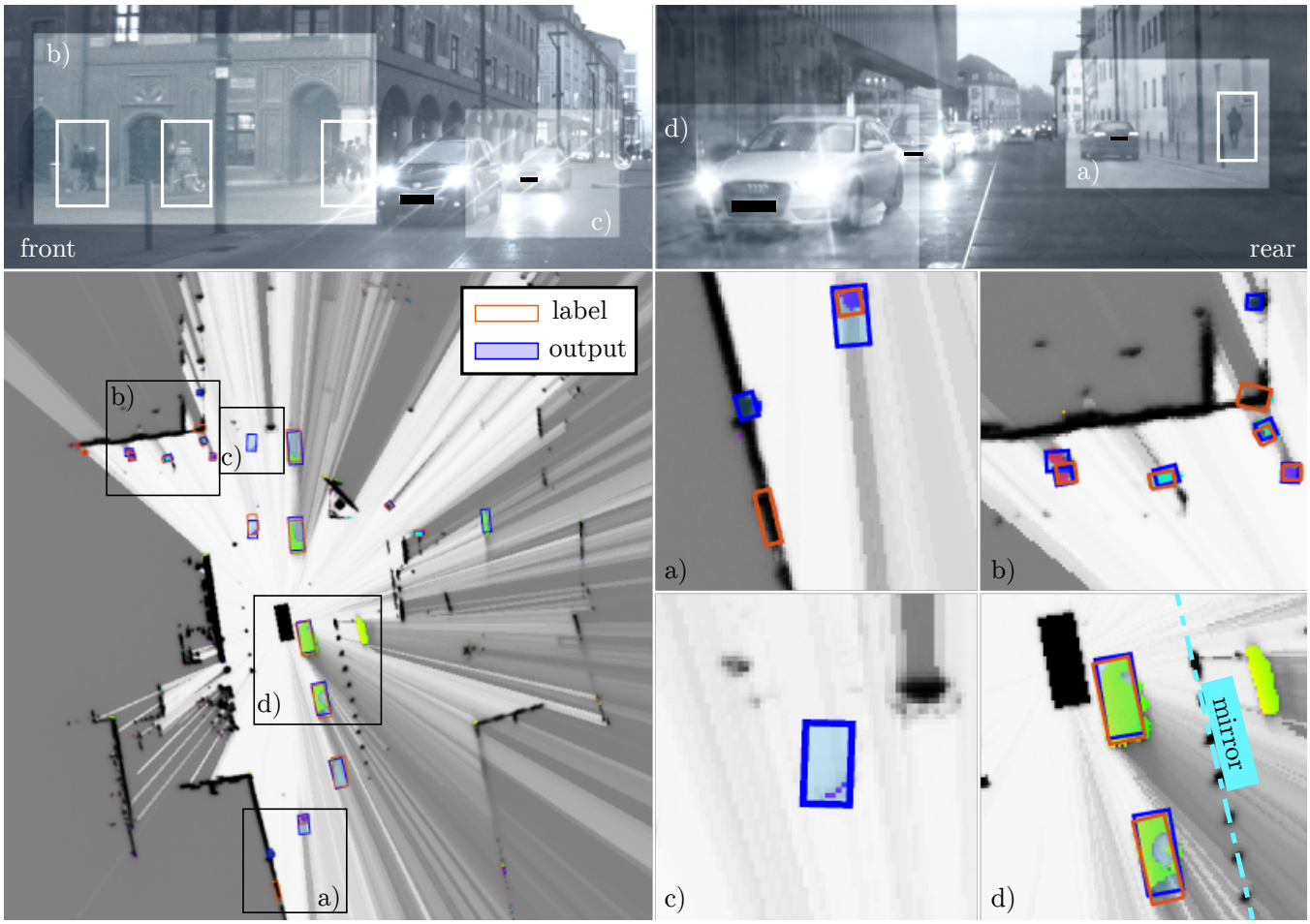

Fig. 10: Example for detected and labeled objects. Regions of interest (a-d) are emphasized in the camera images (top row), marked in the DOGMa and shown enlarged. a) illustrates corrupted labels in terms of false positives, false negatives or wrong size while the network predictions are correct. In b), the label algorithm fails to separate two very close pedestrians, whereas our approach yields one box per pedestrian. The car in c) is missed by automatic label generation but detected by the network. Excerpt d) illustrates a mirrored vehicle trained to be not detected using openstreetmap [28].

positives and false negatives, while the network predicts the correct result. More examples can be seen in the video online.

IX. Conclusions

In this paper, we presented, to the best of our knowledge, the first deep learning approach to detect objects on DOGMas. As an object we understand a bounding box that is defined by a width, length, and orientation. A hand-engineered object tracking has been devised to bypass manual labeling of the data by using acausal information of the future movement of objects. Furthermore, we suggest a single-stage CNN that is capable of detecting the shape and orientation of objects.

We show that our learned approach achieves similar results as the hand-engineered algorithm despite the use of solely causal information. Furthermore, our trained network seems to have better generalization capabilities because it is able to recognize objects which the employed label algorithm lost track of and failed to reinitialize.

Since the goal of this work is to show the general potential of utilizing deep neural nets for object extraction on DOGMas, our dataset so far exclusively entails data recorded from a stationary platform rather than a moving one. Thus, for future work, the presented techniques should be adapted and evaluated on a moving platform.

Acknowledgements

The research leading to these results has received funding from the European Union under the H2020 EU.2.1.1.7. ECSEL Programme, as part of the RobustSENSE project, contract number 661933. Responsibility for the information and views in this publication lies entirely with the authors. The authors would like to thank all RobustSENSE partners.

References

[1] F. Kunz, *et al.*, "Autonomous driving at ulm university: A modular, robust, and sensor-independent fusion approach," in *IEEE Intelligent Vehicles Symposium*, June 2015, pp. 666–673.

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

[3] D. Nuss, *et al.*, "Fusion of laser and radar sensor data with a sequential monte carlo bayesian occupancy filter," in *IEEE Intelligent Vehicles Symposium*, June 2015, pp. 1074–1081.

[4] A. Ngre, L. Rummelhard, and C. Laugier, "Hybrid sampling bayesian occupancy filter," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, June 2014, pp. 1307–1312.

[5] D. Nuss, *et al.*, "A random finite set approach for dynamic occupancy grid maps with real-time application," *arXiv preprint arXiv:1605.02406*, 2016.

[6] S. Ulbrich and M. Maurer, "Probabilistic online pomdp decision making for lane changes in fully automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct 2013, pp. 2063–2067.

[7] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2014, pp. 392–399.

[8] S. Reuter, *et al.*, "Tracking extended targets in high clutter using a ggiw-lmb filter," in *2015 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, Oct 2015, pp. 1–6.

[9] A. Danzer, S. Reuter, and K. Dietmayer, "The adaptive labeled multi-bernoulli filter," in *2016 19th International Conference on Information Fusion (FUSION)*, July 2016, pp. 1531–1538.

[10] T. Yuan, *et al.*, "Extended object tracking using imm approach for a real-world vehicle sensor fusion system," in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Nov 2017, pp. 638–643.

[11] B. Duraisamy, *et al.*, "Track level fusion of extended objects from heterogeneous sensors," in *2016 19th International Conference on Information Fusion (FUSION)*, July 2016, pp. 876–885.

[12] F. Roos, *et al.*, "Reliable orientation estimation of vehicles in high-resolution radar images," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 9, pp. 2986–2993, Sept 2016.

[13] M. Munz, K. C. J. Dietmayer, and M. Mahlisch, "A sensor independent probabilistic fusion system for driver assistance systems," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*, Oct 2009, pp. 1–6.

[14] A. Scheel and K. Dietmayer, "Tracking Multiple Vehicles Using a Variational Radar Model," *ArXiv e-prints*, Nov. 2017.

[15] M. Schtz, *et al.*, "Occupancy grid map-based extended object tracking," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, June 2014, pp. 1205–1210.

[16] G. Tanzmeister and D. Wollherr, "Evidential grid-based tracking and mapping," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1454–1467, June 2017.

[17] S. Steyer, G. Tanzmeister, and D. Wollherr, "Object tracking based on evidential dynamic occupancy grids in urban environments," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 1064–1070.

[18] F. Piewak, *et al.*, "Fully convolutional neural networks for dynamic object detection in grid maps," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 392–398.

[19] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic Occupancy Grid Prediction for Urban Autonomous Driving: A Deep Learning Approach with Fully Automatic Labeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, to be published.

[20] J. Dequaire, *et al.*, "Deep tracking in the wild: End-to-end tracking using recurrent neural networks," *The International Journal of Robotics Research*, 2017.

[21] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1440–1448.

[22] S. Ren, *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 91–99.

[23] K. He, *et al.*, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2980–2988.

[24] J. Redmon, *et al.*, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788.

[25] W. Liu, *et al.*, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, 2016, pp. 21–37.

[26] T. Y. Lin, *et al.*, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2999–3007.

[27] A. P. Dempster, "A generalization of bayesian inference," in *Classic works of the dempster-shafer theory of belief functions*. Springer, 2008, pp. 73–104.

[28] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2017.

[29] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1520–1528.

[30] M. Everingham, *et al.*, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980