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

Evidential Occupancy Grid Map Augmentation using Deep Learning

Sascha Wirges and Christoph Stiller
Mobile Perception Systems Group
FZI Research Center for Information Technology
Karlsruhe, Germany
Email: {wirges, stiller}@fzi.de

Felix Hartenbach
Institute of Measurement and Control
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Email: uvdsu@student.kit.edu

***Abstract*—A detailed environment representation is a crucial component of automated vehicles. Using single range sensor scans, data is often too sparse and subject to occlusions. Therefore, we present a method to augment occupancy grid maps from single views to be similar to evidential occupancy maps acquired from different views using Deep Learning. To accomplish this, we estimate motion between subsequent range sensor measurements and create an evidential 3D voxel map in an extensive post-processing step. Within this voxel map, we explicitly model uncertainty using evidence theory and create a 2D projection using combination rules. As input for our neural networks, we use a multi-layer grid map consisting of the three features *detections*, *transmissions* and *intensity*, each for ground and non-ground measurements. Finally, we perform a quantitative and qualitative evaluation which shows that different network architectures accurately infer evidential measures in real-time.**

I. Introduction

For a safe use of mobile robotic systems detailed maps of the environment are required. However, maps created only from most recent, single measurements are often sparse and subject to occlusions. In order to gather additional knowledge about the scene further domain knowledge is needed.

There are different ways to accumulate information on the environment. With odometry estimation one way is to improve object reconstruction by accumulating multiple registered measurements. However, as the scene is usually dynamic, past sensor information can only be used up to a certain extent. Another way to augment information is to decompose the environment into independent objects, perform classification and reconstruct these objects using highly accurate ground-truth data (e.g. [1]). Although these approaches lead to accurate results for certain object classes, they are computationally expensive, not generic and usually need manually labeled data.

Therefore, our objective in this work is to estimate a generic, precise and augmented environment model using only a single range sensor measurement represented as occupancy grid map. As there is a large amount of sensor data available, this process should also take advantage of data-driven optimization methods such as Deep Learning.

Our main contribution is to provide a framework for evidential grid map augmentation using Deep Learning. By estimating motion in an offline process, we are able to create a highly accurate map of the static environment w.r.t. a fixed reference frame. As the views change due to motion, this map typically becomes denser and contains less occlusions. We use multiple registered measurements to estimate an evidential 3D occupancy grid map of the environment, perform ground surface segmentation and project the evidential masses onto a planar occupancy grid map using combination rules. As this approach is computationally expensive it is only performed for generating training data labels. Using these labels, we then train different neural networks using an occupancy grid map of only one corresponding range sensor measurement as input. Our method is suitable for learning accurate maps with different neural network architectures.

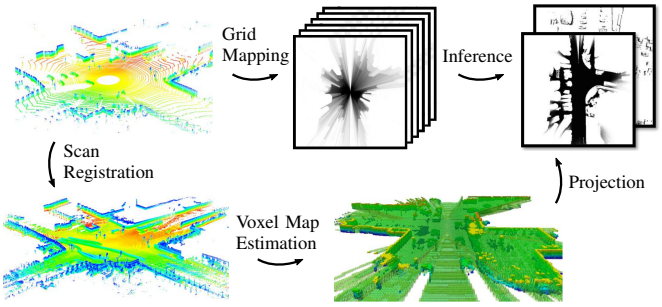


Figure 1: Occupancy map augmentation processing steps. We create a multi-layer grid map containing detections, transmissions and intensities from a single range sensor scan which serves as input of the deep inference network. The inference output is an evidential two-layer occupancy grid map. The network is trained on projections from an evidential voxel map estimated from registered, subsequent range sensor scans.

At first we review related work on occupancy grid maps and neural network architectures in Section II. We then explain the range sensor data preprocessing to obtain training examples in Section III. Then, after recalling the training and validation metrics, we provide information on the training process and network parameters in Section IV. We perform a quantitative and qualitative evaluation of different configurations in Section V. Finally, we conclude our work and propose future plans for grid map inference in Section VI.

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

II. Related Work

Occupancy mapping was first developed for planar grids (2D) and later extended to the 3D domain [2]. Today, implementations for both 2D [3] and 3D [4] applications are available open source. Occupancy grid mapping has applications in collision avoidance [5], sensor fusion [6], object tracking [7], and simultaneous localization and mapping [8]. A variety of sensor models suitable for grid maps exist, e.g. correlation-based [9] or beam-based [10].

The field of environment augmentation is still ongoing research. In an offline process Menze et al. fit 3D CAD models to manually labeled vehicles [1]. Engelmann et al. perform pose estimation and shape reconstruction with compact shape manifolds on stereo camera images [11]. Although these approaches yield accurate results, we rate these approaches as not suitable for real-time / online applications as they work only on detected obstacle instances and a computationally expensive optimization is performed.

In the last years, the accuracy of deep convolutional neural networks (CNNs) in image classification, object detection and localization [12] continuously increased. Today, CNNs yield the best results in these domains. In [13], Ronneberger et al. generalize the approach of fully convolutional neural networks [14] and outperform previous approaches in cell segmentation tasks. The network is similar to encoder-decoder models using convolution-pooling layers to increase the receptive field, thus decreasing the spatial resolution. However, their *Unet* architecture uses skip connections between complementing pooling / unpooling layers to maintain a high spatial resolution.

In [15], the authors show that using a deeper residual network (*Resnet*) the accuracy w.r.t. other architectures can be improved. The performance of Resnets can then be made more computationally efficient by using 1x1 convolutions [16].

III. Preprocessing

As depicted in Fig. 1, the inference framework consists of several preprocessing steps. After providing the foundations in sections III-A and III-B, we describe the methods used to create inference input grid maps in Section III-C and evidential target occupancy grid maps in Section III-D.

A. Range Sensor Scan Registration

In our two-part registration, we represent range sensor scans as point sets. First, we perform globally consistent generalized Iterative-Closest-Points (GICP) [17] in batches of six scans in parallel (see Fig. 2). Second, we use the resulting pose estimates as observations in a subsequent pose graph optimization, e.g. as described in [18]. Each points' covariance is estimated based on its 10 nearest neighbors.

Point correspondences between two scans are obtained by nearest neighbor search in the reference frame. We then keep the reference pose fixed and estimate the remaining poses by minimizing the sum of GICP error functions induced by all point correspondences. In the pose graph optimization step we insert the pose difference between adjacent scans as observations which results in a multi-edge pose graph to optimize. Finally, we determine the poses similar to algorithm 2 presented in [18]. We perform both steps using a nonlinear Least-Squares solver [19]. An exemplary registration result compared to a single range sensor scan is depicted in Fig. 3.

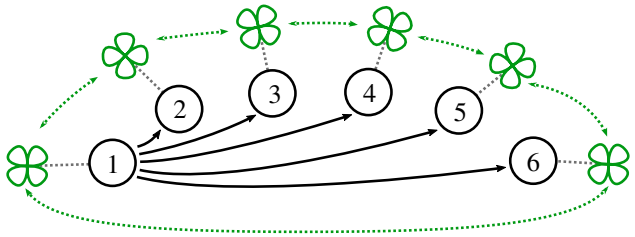
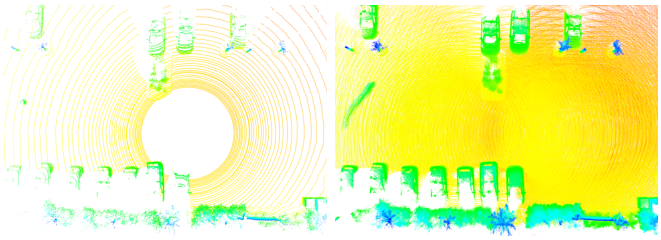


Figure 2: Per-batch registration scheme. Each range sensor scan (green) corresponds to a pose relative to the reference pose 1, indicated by black arrows. We search for closest points between adjacent scans and between the first and the last scan, illustrated by the green arrows.



(a) Single scan (b) 41 scans

Figure 3: Scene top view (colored by height) measured by one scan and 41 registered scans. The scene is mainly composed of cars and trees with a pedestrian walking on the left side. As the pedestrian is moving, they are visible on multiple positions in the registered point set.

B. Ground Surface Estimation

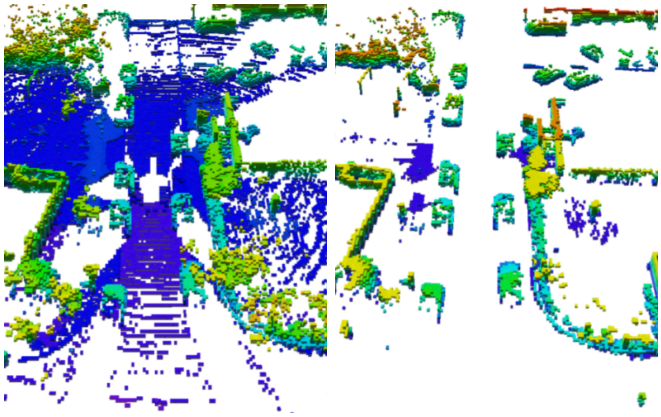
In the majority of scenarios we observed the ground surface to be flat. Therefore, we make a plane assumption in this work using either one scan or an accumulation of multiple registered scans as input. We perform nonlinear Least-Squares optimization [19] to find the optimal plane parameters

$$\mathbf{pl}^* = \arg\min_{\mathbf{pl}} \sum_{\mathbf{p}\in\mathcal{P}} \rho\left(\left\|\mathbf{e}\left(\mathbf{pl},\mathbf{p}\right)\right\|^2\right) \tag{1}$$

which minimize the accumulated point-to-plane error for all points $\mathbf{p}$ of the point set, where $\mathbf{e}\left(\mathbf{pl},\mathbf{p}\right)$ denotes the distance vector between $\mathbf{p}$ and its plane projection point. The loss function ρ is chosen to be the Cauchy loss with a small scale (5 cm) to strictly robustify against outliers. In addition, we remove all points far below the estimated ground plane as they are likely a result of multipath propagation.

C. Input Grid Map

Given a single range sensor scan, we perform ground surface estimation as described in Section III-B. We obtain a *ground* and a *non-ground* subset of points which are then used to compute a grid map. Each grid cell contains



(a) Complete (b) Driving corridor

Figure 4: Evidential voxel map before and after vehicle driving corridor segmentation. Color indicates the height above ground.

the number of *detections*, free-space *transmissions* and the average reflected energy, termed as *intensity*. We determine these values by casting rays to all detected end points from the sensor origin. Fig. 5a–5f show the resulting six grid map layers in an exemplary scenario.

D. Target Occupancy Grid Map

For each scan, we take all scan data in the time interval up to $\pm$2s to create an evidential 3D voxel map. Instead of using data in a time interval one could also use data within a range of poses. Compared to a single measurement, the scene has an extended field of view and is less subject to occlusions. Here, we implicitly make the assumption that the world is static which leads to artifacts when obstacles (e.g. cars, pedestrians) are moving. However, evidential combination of registered measurements can partially mitigate this problem, as moving objects generate higher per-cell uncertainties which is presented in the following.

Given a set of subsequent, registered scans, we estimate the ground surface the same way as for the input. The scans are inserted into a 3D voxel map similar to [4] by raycasting to the scans' endpoints from the sensor origins. We then reduce the 3D voxel map to the vehicle's driving corridor between 0.2 m and 3 m above ground by performing the ground surface segmentation described in Section III-B. Fig. 4 depicts the voxel maps before and after driving corridor segmentation for one scenario.

Each voxel includes the number of reflections and transmissions. For each voxel only two hypotheses, O for a cell being occupied, or F for a cell being free, are possible which results in the frame of discernment $\Theta = \{O, F\}$. The beliefs

$$\text{bel}(O) = 1 - \text{pl}(F) = e(\{O\}) \tag{2}$$
$$\text{bel}(F) = 1 - \text{pl}(O) = e(\{F\}) \tag{3}$$

of a cell being occupied or free can also be expressed by their plausibility counterparts $\text{pl}(\cdot)$ and depends on the elementary evidences reflections e_R and transmissions e_T, respectively.

Based on the recommendation for their inverse sensor model in [4], we choose the elementary evidences as

$$e_R(\{O\}) = 0.4, \qquad e_T(\{O\}) = 0 \tag{4}$$
$$e_R(\{F\}) = 0, \qquad e_T(\{F\}) = 0.1, \tag{5}$$
$$e_R(\{\Theta\}) = 0.6, \qquad e_T(\{\Theta\}) = 0.9, \tag{6}$$

so that they lead to similar behavior when combination rules are applied.

These evidences are combined using Yager's Rule of Combination, in the following indexed by the $\cup$ symbol. This results in the combined evidence of a voxel V

$$e_{\cup,V}(\{O\}) = (1 - e_R(\{\Theta\})^m) \cdot e_T(\{\Theta\})^n \tag{7}$$
$$e_{\cup,V}(\{F\}) = (1 - e_T(\{\Theta\})^n) \cdot e_R(\{\Theta\})^m \tag{8}$$
$$e_{\cup,V}(\{\Theta\}) = 1 - e_{\cup,V}(\{O\}) - e_{\cup,V}(\{F\}) \tag{9}$$
$$e_{\cup,V}(\{\emptyset\}) = 0 \tag{10}$$

with the number of reflections m and the number of transmissions n. As a result of reflections and transmissions within the same voxel, conflicting evidence masses are assigned to the entire frame of discernment. Thus grid cells temporarily covered by moving objects yield a high uncertainty which mitigates the assumption of a static environment for target data generation.

Finally, the evidential 3D voxel map is projected onto a plane in order to get an evidential 2D occupancy grid map. In the following, we summarize $k = 1 \dots K$ voxels $V^{(k)}$ on top of each other to a *pillar* P. However, the voxel evidences cannot be combined as shown previously as they describe different locations. As for the voxels, a pillar is assigned the two hypotheses O and F. Whereas evidence for an occupied voxel is also evidence that the corresponding pillar is occupied, evidence for a voxel being free is less an evidence for the whole pillar being free. The latter is only the case if all pillar voxels have high evidence for being free. On the one hand, this yields the belief

$$\text{bel}_{\cup,P}(F) = e_{\cup}(\{F\}) = \prod_{k=1}^{K} e^{(k)}_{\cup,V}(\{F\}) \tag{11}$$

for a pillar being free, similar to chaining all voxel evidences for being free by logical *and* functions. On the other hand, the belief

$$\text{bel}_{\cup,P}(O) = e_{\cup}(\{O\}) = 1 - \prod_{k=1}^{K} 1 - e^{(k)}_{\cup,V}(\{O\}), \tag{12}$$

for a pillar being occupied can be interpreted as chaining all voxel evidences for not being free by logical *or* functions.

As a result, the belief

$$\text{bel}_{\cup,P}(\Theta) = 1 = e_{\cup,P}(\Theta) + e_{\cup,P}(F) + e_{\cup,P}(O) \tag{13}$$

shows that the correct hypothesis is in Θ with evidence

$$e_{\cup,P}(\Theta) = 1 - e_{\cup,P}(F) - e_{\cup,P}(O). \tag{14}$$

Finally, we obtain a two-channel grid map used for training our inference networks containing the beliefs $\text{bel}_{\cup,P}(O)$ and $\text{bel}_{\cup,P}(F)$. Any additional channel would be redundant.

IV. Training

A. Data Set and Training Strategy

We created a dataset containing 7707 range sensor scans using a Velodyne HDL64E-S2 lidar. The dataset contains sequences from different traffic scenarios such as parking lots, highways, cities or rural roads. After preprocessing the data (see Section III), we split the dataset into 5995 training and 712 evaluation samples covering different driving environments. We created grid maps with an initial size of $100\,\mathrm{m} \times 100\,\mathrm{m}$ and quadratic cells with an edge length of 12.5 cm. To further increase the number of training examples, we applied random rotation and offset from the sensor origin and cropped areas of $64\,\mathrm{m} \times 64\,\mathrm{m}$ that were then used as training examples. Due to our limited computational resources we trained all networks using Minibatch-SGD with four samples per batch, layer normalization [20] and used the Adam optimizer with a learning rate of $1 \cdot 10^{-4}$ for Unets and $5 \cdot 10^{-4}$ for Resnets.

B. Metrics

To train the networks, we use cell-wise metrics. Given a grid with I cells, we define the loss

$$L = \left(\sum_{i=1}^{I} w^{(i)} \right)^{-1} \sum_{i=1}^{I} w^{(i)} l^{(i)} \tag{15}$$

as the average weighted per-cell loss with the belief $\mathrm{bel}(O)$ for a cell being occupied, $\mathrm{bel}(F)$ for the cell being free and the estimates $\mathrm{bel}'(O)$ and $\mathrm{bel}'(F)$, respectively.

Using the above notation, we define the per-cell residuals

$$\epsilon_O = \mathrm{bel}(O) - \mathrm{bel}'(O), \tag{16}$$

$$\epsilon_F = \mathrm{bel}(F) - \mathrm{bel}'(F) \tag{17}$$

and per-cell L_1 and L_2 losses

$$l_1 = |\epsilon_O| + |\epsilon_F|, \tag{18}$$

$$l_2 = \epsilon_O^2 + \epsilon_F^2. \tag{19}$$

As the target data includes inaccuracies due to registration errors, sensor noise or moving obstacles, we want to scale the per-cell loss depending on the target data uncertainty. The loss definitions (Eq. 18, 19) with $w = 1$ would therefore yield to an approximation of the target data uncertainty. To make the inference result more independent to this uncertainty, we propose two modifications of the above cost terms.

First, we propose to scale the per-cell loss depending on the weight

$$w_k = 1 + k \cdot (C - 1) \qquad k \in [0, 1] \tag{20}$$

which depends on the target data certainty

$$C = \mathrm{bel}(O) + \mathrm{bel}(F). \tag{21}$$

Second, we suggest to adapt the cost asymetrically for false free predictions, e.g. in the L_1 loss function such that

$$l_{1,k} = |\epsilon_O| + |\epsilon_F| + k \cdot \epsilon_F \qquad k \in [0, 1] \tag{22}$$

yields to an underestimation of $\mathrm{bel}(F)$.

Net	Architecture and ID (U: UNet, Res: Resnet)
HP Explicit	Explicitly set hyper parameters
Loss	Loss function definition
F	Number of filters in the first layer
S	Stack size (Convolutions + ReLUs)
D	Network depth
HP Implicit	Implicitly set hyper parameters
CL	Maximum number of convolution layers
Rec	Receptive field size in multiples of 0.125 m
Para	Total number of parameters
Training	Training parameters
Ep	Number of epochs
Step	Number of training steps
Metrics	Evaluation metrics
L1	Standard L_1 loss (Eq. 18, $w = 1$)
L2	Standard L_2 loss (Eq. 19, $w = 1$)
RelUnc	Relative uncertainty (Eq. 23)
False O	False occupied (Eq. 24)
False F	False-free (Eq. 25)
Time	Inference time

Table I: Abbreviations used in training and evaluation.

C. Networks and Hyperparameters

Our Unets are a generalized modification of [13]. The Resnets used are similar to [15] but use dilated convolutions [21]. As depicted in Table I, we explicitly set the network hyperparameters by varying the loss function type, the initial number of filters, the stack size and the network depth. A stack is defined as consecutive convolutions and ReLUs with identical filter size. After each stack, pooling is performed (Unet) or the dilation rate gets doubled (Resnet) and for each the filter size gets doubled. The Network depth describes the number of stacks of the encoder part. A bottom stack follows as well as a decoder part with the same number of stacks as the encoder. Thus, a network with Depth D has 2D+1 Stacks. The maximum number of convolutions CL of a network path is greater or equal (2D+1)S. The number of filters cannot be altered in residual blocks so extra 1x1 convolutions are added between the stacks.

We trained Unets and Resnets with several configurations. One Unet (U5, *NoSplit*) was trained for a three channel grid map input without splitting the range sensor scans into ground and non-ground points. Besides the standard loss (Eq. 15) with $w = 1$, we also train one Unet (U6, *W0.9*) with the weight definition according to Eq. 20 with $k = 0.9$ depending on the target data certainty. Unet U7 (*F0.8*) was trained with the asymetric loss in Eq. 22 with $k = 0.8$.

V. Evaluation

Table II depicts the best network configurations we found in our evaluation. For the abbreviations used, please refer to Table I.

A. Quality Metrics

As we infer scalar beliefs of cells being free or occupied, common metrics used in the evaluation of binary classifiers, e.g. precision or recall [22] cannot be applied. Therefore, we define further metrics to evaluate the inference quality in

	HP Explicit				HP Implicit			Training		Metrics					
Net	**Loss**	**F**	**S**	**D**	**CL**	**Rec**	**Para**	**Ep**	**Step**	**L1**	**L2**	**False O**	**False F**	**RelUnc**	**Time**
							10^3		10^3	10^{-4}	10^{-5}		10^{-2}	10^{-2}	ms
U1	L1	8	3	3	25	152	123	40	30	8.52	5.18	5.03	44.80	1.02	**38**
U2	L1	16	3	3	25	152	487	40	60	8.14	4.97	5.62	36.60	1.03	44
U3	L1	8	3	4	32	313	492	40	60	8.44	5.19	5.13	49.30	1.01	39
U4	L2	8	3	3	25	152	123	40	60	10.40	**4.35**	68.10	59.50	1.00	**38**
U5	L1 NoSplit	8	3	3	25	152	123	40	60	8.93	5.64	6.53	55.30	1.03	**38**
U6	L1 W0.9	8	3	3	25	152	123	40	60	11.30	6.57	42.20	83.40	**0.92**	**38**
U7	L1 F0.8	8	3	3	25	152	123	40	60	11.20	7.74	**5.26**	**3.42**	1.13	**38**
Res1	L1	8	2	3	26	93	73	120	181	8.18	4.97	4.21	7.11	1.02	72
Res2	L1	8	2	4	36	193	295	87	130	8.13	4.89	6.22	10.10	1.01	119
Res3	L1	8	1	4	22	105	172	120	180	8.16	5.03	3.23	6.20	1.02	82
Res4	L1	16	2	3	26	93	288	103	150	**8.08**	4.93	3.47	7.21	1.02	107

Table II: Selection of network parameters and evaluation results. We evaluated Unets and dilated Resnets with different hyperparameters. Networks for qualitative analysis and best performance results highlighted.

addition to the metrics presented in Section IV-B on which the networks are trained on.

We define the relative uncertainty

$$m_{\text{RelUnc}} = \frac{\hat{U}}{U} = \frac{1 - \text{bel}'(F) - \text{bel}'(O)}{1 - \text{bel}(F) - \text{bel}(O)} \tag{23}$$

as the ratio of the predicted uncertainty $\hat{U}$ and the target uncertainty U. In the default case, $m_{\text{RelUnc}} \approx 1$. However, m_{RelUnc} might vary a lot for some configurations and helps us describe their influence.

Whereas L_1 and L_2 norm are assembled by per-channel distances, e.g. $\text{bel}'(F)$ and $\text{bel}(F)$, we aim to penalize contradictory inference w.r.t. the target data. Therefore, we define the false occupied / free metrics

$$m_{\text{FalseO}} = \max(0, \text{bel}'(O) + \text{bel}(F) - 1), \tag{24}$$

$$m_{\text{FalseF}} = \max(0, \text{bel}(O) + \text{bel}'(F) - 1) \tag{25}$$

that penalize areas of inferred high belief in contradiction to the target data. Compared to m_{FalseO}, m_{FalseF} is usually more critical because an acute collision might be possible if obstacles are detected too small or not detected at all. For both of the metrics above, areas of high uncertainty, either labeled or predicted, will induce only little error.

B. Processing Time

We evaluated the processing times on a 2.5 GHz six core Intel Xeon E5-2640 CPU with 15 MB cache and an NVIDIA GeForce GTX 1080 Ti GPU with 11 GB graphics memory. The input grid map estimation including ground plane segmentation takes 18 ms on average. Given this input, we evaluated the inference times for different networks and hyperparameter configurations. The results are depicted in Table II. We observe the Unets to be faster than Resnets as they might take advantage of higher parallelization. However, even for Resnets we achieve real-time performance (processing time less than 100 ms) for one configuration.

C. Discussion

We observed that our networks trained with L_2 loss yield higher false occupied errors compared to networks trained with L_1 loss. The qualitative inference results depicted in Fig. 5 show that our networks generalize well when moving obstacles are present in the target data. Compared with input grid map layers split as ground and non-ground, *U5* achieved only a slightly worse performance which shows that the network filters ground surface points to some extent. Network *U6* achieved comparably high false free/occupied errors but is well suited for grid map augmentation due to its low relative uncertainty. Network *U7* achieved small false free/occupied errors but therefore has a higher relative uncertainty which might be a gain for accurate grid map filtering instead of augmentation.

VI. Conclusion

We presented a framework for evidential grid map augmentation using Deep Learning techniques. By performing quantitative and qualitative evaluation for different configurations, we show that Resnets and Unets infer evidences accurately from domain-specific training data and can be tuned towards grid map augmentation or reliable filtering w.r.t. safety metrics. Whereas Resnets achieve more accurate results, Unets have a significantly smaller inference time.

In future works we are going to improve the label data generation, especially the registration of range sensor measurements. This step should decrease target data uncertainty and thus speed up the training process. Also, we are going to extend our approach to time series of occupancy grid maps in order to augment and predict moving obstacles.

References

[1] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3061–3070, 2015.

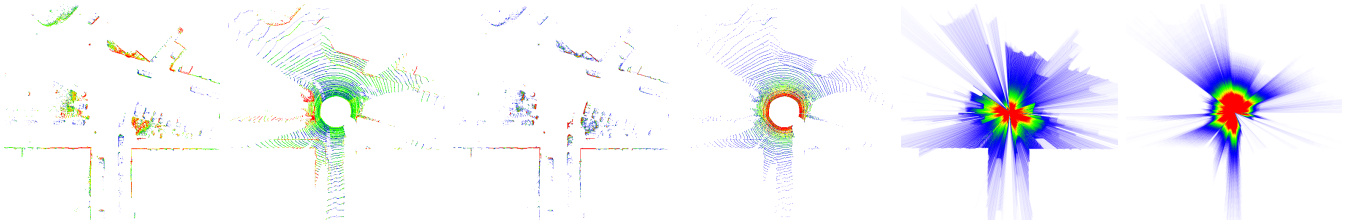
[2] A. Elfes, "Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception," *Autonomous Mobile Robots: Perception, Mapping, and Navigation*, vol. 1, pp. 136–146, 1991.

[3] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5.

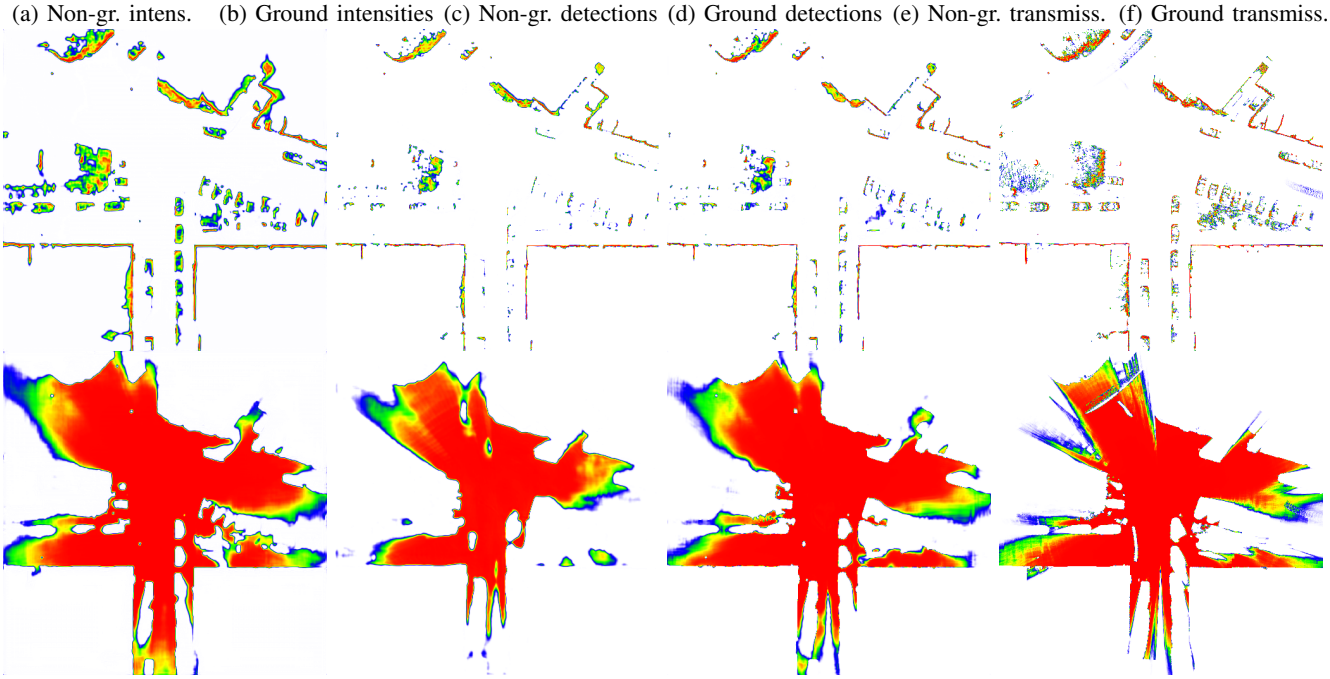
[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework based on Octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[5] J. Borenstein and Y. Koren, "The Vector Field Histogram: Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[6] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt, "Map Learning and High-Speed Navigation in RHINO," 1998.



(a) Non-gr. intens. (b) Ground intensities (c) Non-gr. detections (d) Ground detections (e) Non-gr. transmiss. (f) Ground transmiss.



(g) U6 (h) U7 (i) Res4 (j) Target data

Figure 5: Top row: Grid map layers used as inference network inputs. Fig. 5i, 5g and 5h depict the inference output for different networks, where bel(O) is shown in the middle row and bel(F) in the bottom row. Fig. 5j depicts the target occupancy grid map used for training the networks. Low values indicated by white color, high values by red color. Due to the mapping process, moving objects (e.g. a moving car in the upper left corner) yield high uncertainty in the target data. However, for a given sensor data frame the network generalizes well and augments the map correctly. Best viewed digitally with zoom.

[7] C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous Localization, Mapping and Moving Object Tracking," *International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[8] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time Loop Closure in 2D LIDAR SLAM," *IEEE International Conference on Robotics and Automation*, pp. 1271–1278, May 2016.

[9] P. Biber and W. Strasser, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," *IEEE International Conference on Intelligent Robots and System*, vol. 3, pp. 2743–2748, 2003.

[10] A. Schaefer, L. Luft, and W. Burgard, "An Analytical Lidar Sensor Model Based on Ray Path Information," *IEEE International Conference on Robotics and Automation*, vol. 2, no. 3, pp. 1405–1412, 2017.

[11] F. Engelmann and B. Leibe, "Joint Object Pose Estimation and Shape Reconstruction in Urban Street Scenes Using 3D Shape Priors," *Lecture Notes in Computer Science*, vol. 663, pp. 219–230, 2016.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," pp. 234–241, 2015.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015.

[16] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," 2016.

[17] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Proc. of Robotics: Science and Systems*, vol. 2, p. 4, 2009.

[18] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A Tutorial on Graph-Based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[19] S. Agarwal, K. Mierle, and Others, "Ceres Solver," http://ceres-solver.org.

[20] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," jul 2016.

[21] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," 2015.

[22] D. M. Powers, "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness & Correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.