A Large-Scale Mapping Method Based on Deep Neural Networks Applied to Self-Driving Car Localization

Vinicius B. Cardoso
Departamento de Informática
Universidade Federal do Espírito Santo
Vitória, Brazil
vinicius@lcad.inf.ufes.br

André Seidel Oliveira
Departamento de Engenharia Elétrica
Universidade Federal do Espírito Santo
Vitória, Brazil
aseideloliveira@gmail.com

Avelino Forechi
Coordenadoria de Engenharia Mecânica
Instituto Federal do Espírito Santo
Aracruz, Brazil
avelino.forechi@ifes.edu.br

Pedro Azevedo
Departamento de Informática
Universidade Federal do Espírito Santo
Vitória, Brazil
pedro@lcad.inf.ufes.br

Filipe Mutz
Coordenadoria de Informática
Instituto Federal do Espírito Santo
Serra, Brazil
filipe.mutz@ifes.edu.br

Thiago Oliveira-Santos
Departamento de Informática
Universidade Federal do Espírito Santo
Vitória, Brazil
todsantos@lcad.inf.ufes.br

Claudine Badue
Departamento de Informática
Universidade Federal do Espírito Santo
Vitória, Brazil
claudine@lcad.inf.ufes.br

Alberto F. De Souza, *Senior Member, IEEE*
Departamento de Informática
Universidade Federal do Espírito Santo
Vitória, Brazil
alberto@lcad.inf.ufes.br

***Abstract*— We propose a new approach for real time inference of occupancy maps for self-driving cars using deep neural networks (DNN) named NeuralMapper. NeuralMapper receives LiDAR sensor data as input and generates as output the occupancy grid map around the car. NeuralMapper infers the probability of each grid map cell from one of the three following classes: Occupied, Free and Unknown. The system was tested with two datasets and achieved an average accuracy of 76.48% and 73.81%. We also evaluated our approach for localization purposes in a self-driving car and most of the localization pose errors were less than 0.20m with an RMSE of 0.28 which are close to the results in the literature for methods using other grid mapping approaches.**

Keywords—self-driving cars, mapping, deep neural networks.

I. INTRODUCTION

Self-driving cars have advanced greatly over the last twenty years. In 2004, the USA Defense Advanced Research Projects Agency (DARPA) promoted the first self-driving car competition, with the goal of crossing the Mojave Desert without human intervention [1] – in this competition none of the participants achieved the goal. After this first attempt, two other competitions were promoted by DARPA, in 2005 and 2007, and five and six vehicles completed the proposed challenges, respectively. These events have ignited the self-driving cars' development and contributed to self-driving cars become reality nowadays. Companies like Google, Volvo and Uber are running trials in several cities around the world offering autonomous driving services to the public.

Despite the great advances already made in this area, today's self-driving cars still do not deal accurately with all situations and weather conditions encountered in everyday urban traffic. For this reason, universities and research centers worldwide are conducting research to improve this technology and increase the autonomy level of self-driving cars.

The IARA (Intelligent Autonomous Robotic Automobile) self-driving car, shown in Figure 2, has being developed since 2009 by the High Performance Computing Laboratory (*Laboratório de Computação de Alto Desempenho* - LCAD) of Federal University of Espírito Santo (*Universidade Federal do Espírito Santo* – UFES, Brazil). IARA is based on a Ford Escape Hybrid adapted with a variety of sensors and processing units. Its autonomy system follows the typical architecture of self-driving cars [2] and as such is composed of several subsystems, which includes a Mapper [3], a Localizer [4], a Moving Obstacle Tracker [5], a Traffic Sign Detector [6][7], a Route Planner, a Path Planner, a Behavior Selector, a Motion Planner [8], an Obstacle Avoider [9] and a Controller [10], among others. Maps used by IARA are built from LiDAR point clouds in online and offline mapping. Considering that LiDAR readings may be inaccurate due to environmental conditions (e.g. rain, falling leaves, uneven reflection surfaces) or due to measurement errors caused by noise from the sensor itself, a mapping system cannot completely rely on the distances and positions directly measured by LiDAR. Therefore, for mapping, it is necessary to use techniques that take into account all these uncertainties to try and filter the sensor errors as much as possible.

The algorithm currently used in IARA seeks to update the probability of occupancy of regions with each new reading of the LiDAR sensor. For this, an Occupancy Grid Mapping (OGM) algorithm [3], based on the Bayes filter, is applied for gradually increase or decrease the occupancy confidence of each cell of the grid-map. However, this approach is not very accurate for objects at long distances. Furthermore, it is necessary to use an algorithm to infer the occupancy of cells that were not reached by any sensor reading, since the OGM used does not take into account spatial discontinuities. In

This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001; Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) - grants 311654/2019-3, 200864/2019-0 and 311504/2017-5; and Fundação de Amparo à Pesquisa do Espírito Santo - Brasil (FAPES) – grant 84412844.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

addition, the mapping algorithm needs to treat several special cases along thousands of lines of code to find the best way to properly extract the features of sensor readings [3].

Motivated by these limitations of the mapping method of IARA, as well as the advances of deep learning, in this work, we propose a new approach for real time inference of occupancy maps based on Deep Neural Networks (DNN), named NeuralMapper. NeuralMapper is a subsystem of IARA's autonomy software architecture that receives LiDAR sensor data as input and generates as output an occupancy grid map around the car. In NeuralMapper, every LiDAR point cloud is transformed from spherical to 2D Cartesian coordinates in the LiDAR reference frame. Then, five matrices are computed from the 2D LiDAR data using statistics of this data (the input data of NeuralMapper follows that employed by Caltagirone et al. [11]). Those five matrices are combined into a five-channel tensor that is used as input to the NeuralMapper DNN. The statistics considered are the number of laser rays that hits a cell of the grid map around the car, as well as the maximum, minimum, mean and standard deviation of the height of the obstacles hit by rays that falls into a cell.

The output of the network is an occupancy map with the corresponding probability associated with each of the three possible classes of each grid-map cell: unknown occupancy, free or occupied. This map is transformed to the car reference frame and published to the other IARA's subsystems (see Figure 2). A block diagram of NeuralMapper is shown in Figure 1.

We evaluated NeuralMapper with real world data using IARA. The system was tested with two datasets built with IARA's sensor data and achieved an average accuracy of 76.48% in the first and 73.81% in the second dataset. We also evaluated the use of NeuralMapper' maps for localization purposes using IARA and most of the IARA's localization errors were smaller than 0.2m, with an RMSE of 0.28m. These results are close to those in the literature for methods using other grid mapping approaches.

The following sections will be devoted to assess the capability of this method by both literature review and practical experimentation in IARA.

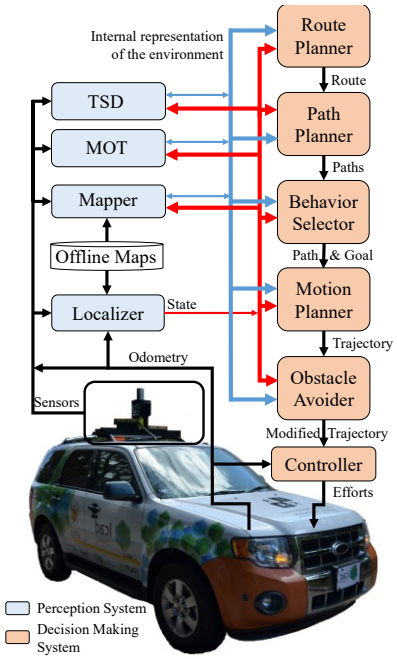

Figure 2- Overview of the typical architecture of self-driving cars [2]. TSD denotes Traffic Signalization Detection and MOT, Moving Objects Tracking.

II. Related Work

There are several techniques presented in literature that uses neural networks with point cloud data acquired from sensors such as, LiDAR, radar, sonar and others. Those sensor's data are used for several purposes such as, build occupancy grid maps, localization and 3D point cloud segmentation.

Santos et al. [12] propose a system that uses data from ultrasound sensors installed around a robot that serve as input to a feed-forward neural network that determines if the cells around the robot are free or occupied. This work only deals with the occupancy of the cells of a prior map skeleton of the environment.

Gupta et al. [13] propose the creation of occupancy grid maps using data provided by a sonar sensor. The measurement of the sensor is converted into a probabilistic representation of the environment that describes if a grid cell is empty, occupied

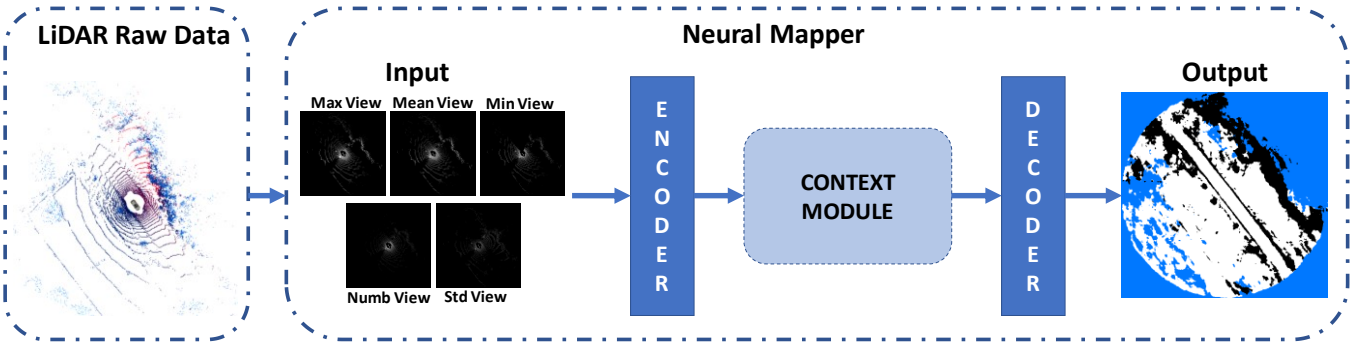

Figure 1 – Overview of NeuralMapper architecture. The input on the left shows the LiDAR point cloud raw data. NeuralMapper projects the raw data on the ground as a set of 2D statistics maps. The statistics maps are neurally processed by the encoder, the context module and the decoder to produce an occupancy grid map as output. In the map, unknown cells are blue, free cells are white and the occupied cells are black.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

or if is an unknown area. As changes in the environment affect conversions, a neural network was used to learn these conversions, making the sensor data more adaptable to changes.

Weston et al. [14] present a technique that uses raw radar data and LiDAR data to create an occupancy grid map using a DNN. Radar sensors are able to detect long range and partial occluded objects and are cheaper than LiDAR but radar scans are notoriously difficult to interpret due to the presence of several pertinent noise artefacts while LiDAR systems provide precise, fine-grained measurements. The DNN was trained with partial occupancy labels generated by LiDAR data and was evaluated on five hours of recorded data in a dynamic urban environment.

Wu et al. [15] propose the SqueezeSeg, that is a CNN (Convolution Neural Network) to perform semantic segmentation of road-objects from LiDAR point clouds, such as cars, pedestrians and cyclists. The CNN was trained with labelled LiDAR point-cloud from KITTI dataset [16] and with data from a simulated LiDAR of a high-fidelity game engine. The input of SqueezeSeg is a point cloud and the output is a point-wise label map that is refined by a conditional random field (CRF). Each point cloud is processed in 8.7ms. SalsaNet proposed by Aksoy et al. [17] is also a neural network that segments directly from LiDAR point clouds the drivable free-space and vehicles. The training process uses KITTI dataset [16] but it does not provide drivable free-space labels, so they propose an automatic system to annotate it in the point-cloud using the neural network MultiNet [18]. Then the point cloud is represented as a 4D Bird-Eye-View image that contains information about maximum and minimum elevation, reflectivity and number of projected points. This image is used as input to train the DNN. SalsaNet results presented a mean IoU of 79.74% while SqueezeSeg results presented a mean IoU of 69.80%.

Caltagirone et al. [11] present a neural network approach that detect roads using only LiDAR data. The input of the DNN is a top-view image of the LiDAR where the points are in grey-scale representing mean elevation and density. Then a fast fully convolutional neural network (FCN) was trained with KITTI dataset [16] and was able to detect roads with a precision of 94.15%. The DNN architecture adopted in this work was based on a fully convolutional neural network used in [11] but the purpose and dataset are different as will be shown next.

III. A LiDAR-based DNN for Occupancy Grid Mapping

First, it should be recalled that OGM is about inferring an occupation probability for each cell on the map. The occupation probability of each cell can assume a continuous value between zero and one, therefore it can be used to classify the cell as free or occupied cell.

A. The NeuralMapper Subsystem

The NeuralMapper (Figure 1) is the subsystem that receives IARA's LiDAR sensor data as input and generates as output the occupancy map around the car. The input data of NeuralMapper follows Caltagirone et al. [11]. First, every LiDAR point cloud is transformed from spherical to 2D Cartesian coordinates in the LiDAR reference frame. Also following Caltagirone et al. [11], five statistics matrices are computed from the 2D coordinate matrix and normalized between zero and one. Those five normalized statistics matrices are combined into a five-channel tensor used as input to the network. The statistics considered are the number of laser rays that hits the cell, max/min/mean/std height.

In order to adapt the FCN [11] output to IARA's map, it was established that the network should return a map of probabilities for three classes: Occupied, Free and Unknown. Thus, the neural network would make inferences compatible to the probabilistic methods widely adopted in IARA's subsystems (Figure 1).

In this way, the output of the network is an occupancy map with the corresponding probability associated with each of the three classes for each output neuron (cell). This map is transformed into the car reference frame and published to the other IARA's subsystems. The output neuron probability is converted into occupancy probability as follows. If the most likely class is the Unknown, the map cell receives a value of -1, otherwise the probability of the class Unknown is zeroed and the probabilities of the other two classes are normalized so that the sum of the two is equal to 1. The code is available as an IARA's module at https://github.com/LCAD-UFES/carmen_lcad .

B. DNN architecture

Figure 1 shows the general architecture of the FCN which is divided into three large groups of layers, called Encoder, Context Module and Decoder.

The **Encoder** receives the inputs and reduces the size of the inputs data trying and minimizing the loss of information. For this, two normal convolutional layers are used with 3x3 kernel, stride 1 and 32 features map with Exponential Linear Unit (ELU) activation function [19], followed by a max pooling layer of 2x2 with stride 2.

The **Context Module** is mainly formed of dilated convolution layers. Dilated convolutions have the ability to increase the receptive field of the convolutional filter, in order to increase the ability to infer continuity between contiguous map cells. For mapping obstacles, there are many discontinuities over long distances in sensor readings. So, it would be interesting to apply dilation for predicting the continuation of obstacles to more distant places, where the sensor is no longer very effective. There are eight Dilated convolutions layers with ELU activation function, 3x3 kernel, and 128 features maps. These layers have a progressive increase in dilation, so that in the last layer there is an expansion of 32x64. Unlike the work proposed by Caltagirone et al. [11], in this work, it was determined that the output map would be square, with a range of 60 meters around the car. Therefore, the dilation used in each layer was of the same magnitude (e.g. the last layer has expansion of 64x64).

The **Decoder** is the last part of the architecture, where the data processed on the network regains its original dimensions with unpooling operations. The implementation of the unpooling used [11] saves the positions of the maximum values extracted in the max pooling in order to insert the values again

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

in those same positions while setting the others to zero. The decoder continues with two more normal convolutional layers with 3x3 kernel, stride 1, with 32 features map and the last layer with 3 channels output. Finally, feature maps enter a log softmax layer, which returns a multidimensional map output, where each dimension is the logarithm probability for each class. Therefore, the occupancy map is built by comparing the probabilities of each of these three classes for each cell at the network output, as described in Section III.A.

IV. Experimental Methodology

This section presents the methodology employed to evaluate the proposed systems. First, the datasets used in the experiments are described and then the hyperparameters and procedure for training the neural network. After that, we present the methodology for generating the ground truth for the localization and the metrics for evaluating the neural network performance and the localization accuracy.

A. Datasets

Large datasets are required for training deep neural networks. At first, datasets could be created by manually labelling map cells or by using floor plans. A lower cost solution though is the use of IARA's offline mapping system to generate input and output pairs from logs of sensor data. All the data from the log are used for generating an OGM of the environment. Then, for each LiDAR point cloud, we crop a circle in the OGM centered at the car pose and with radius of 60m. This cropped circle is defined as the output for the LiDAR point cloud.

Three classes are considered for each cell: unknown, occupied and free. Since the OGM is probabilistic, we use soft labels instead of one-hot encoding. The probability of each class is given by the occupancy probability of the cell. Figure 4 illustrates a point cloud and its associated output. The advantages of using the offline map instead of the online OGM are the following. First, it is more accurate due to the integration of several instantaneous maps. Second, it presents more occupation information which allows the neural network to learn how to fill areas that are not observed by sensors.

Two datasets produced from logs of sensor data are used in the experiments. The datasets will be referred as Dataset 1 and Dataset 2. The datasets were recorded in the same environment, the 3.5km UFES beltway and split into three disjoint regions as presented in Figure 3.

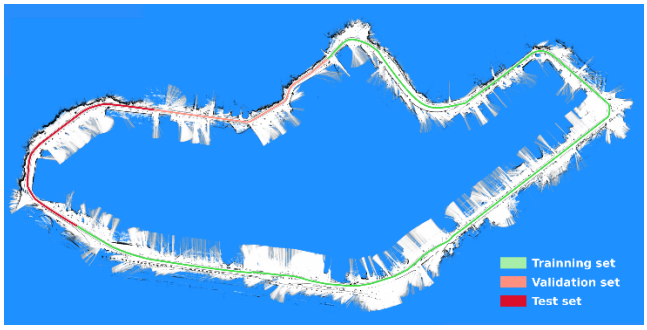

Figure 3 -UFES beltway map. In green, orange and red is shown the training, validation and test set region, respectively.

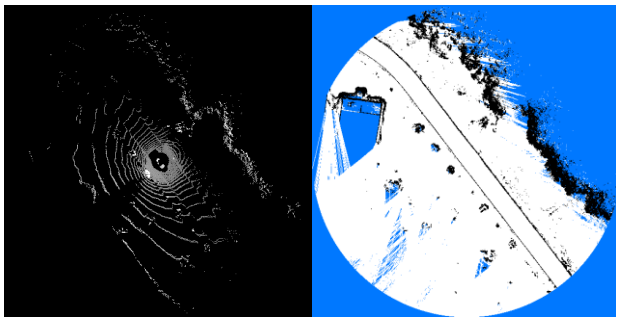

Figure 4 - The left image is the max high statistic map, one of the five inputs to NeuralMapper. The right image is the corresponding ground truth. Both images were normalized for visualization.

The regions from Dataset 1 are used for training, validating and testing the neural network, while all regions from Dataset 2 are used for test. Note that the same regions are selected in both datasets and that one of the regions from Dataset 2 corresponds to the same region used to train the neural network.

The Dataset 1 was extracted from a log collected at dawn on 23/03/2016. Dataset 1 has 1445 scans and with data augmentation (details are presented in the next section) a total of 7228 samples, that were separated in 74% for the training set, 10% for the validation set, and 16% for the test set. The log was recorded at dawn to minimize the presence of moving objects. Dataset 2 is extracted from a log captured at 03/10/2019. Since the log was captured three years after the one used for building the Dataset 1, it presents different features due to time passing such as vegetation growth, new buildings and changes in roads. Dataset 2 also has 1445 scans and 7228 samples with dataset augmentation.

The input and output pairs were generated every two meters traveled by the vehicle to prevent the network from having unbalanced data for different regions. By doing so, the database no longer has information in regions where the car moved more slowly in the simulated log.

B. Training

The DNN input consists of five maps of statistics regarding the height of points around the car. These statistics maps are represented by a float tensor, where each cell represents a 20x20cm square of the environment. The car is always in the center of the map. Each cell observed by the laser will store the maximum height, minimum height, average height, standard deviation of height and number of points. To do so, first the points in the cloud undergo a transformation of coordinates, from spherical to Cartesian. Then, the points that hit the same cell are grouped and used for computing the statistics. Finally, these five dimensions of statistics (maximum, minimum height, standard deviation and number of points) are then normalized between zero and one and used as input for the network.

To normalize the maps, in case of statistic maps that use height, the upper bound value was the LIDAR's height position (1.86 meters in our case), for the number of points, was calculated the max possible number of points inside one cell (64 points in our case). The map's cells are initialized as -1, and this remains as the min value.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

As can be seen in Figure 4, the number of pixels of obstacle cells in each image is much smaller than free and unknown cells. This phenomenon generates a tendency for the network to opt for the last two classes over the first. To avoid this, weights inversely proportional to the number of pixels of the classes in each batch were used, so that the loss function multiplies the weight when computing the error. This "valuation" of errors in relation to less frequent classes compensates for the imbalance between them [20].

For data augmentation [21], we applied horizontal flips and 90 degrees rotation in each direction for all training image which increased the dataset 3 times. The cross-entropy loss function and ELU activation function were chosen in accordance with the work of Caltagirone et al. [11], besides a dropout of 25%. The neural network was trained for 50 epochs with an initial learning rate of 0.0005 and learning rate of decay of 0.5 at each 15 epochs. The training was performed on a 12GB Nvidia Tesla K40 GPU.

C. *Localization Ground Truth Generation*

The localization is an important task for self-driving car navigation. Generating a ground truth for evaluating the localization is quite challenging. A straightforward idea is to use GPS for comparison with localization estimates. However, although globally consistent, GPS data has significant amount of noise and GPS measurements are not necessarily consistent with the map.

Our approach for generating the localization ground truth is similar to the ones employed in [4][22] and [23]. We use the GraphSLAM technique described in [3] for estimating the poses of the vehicle using data from GPS, odometry and the LiDAR (for handling loop closures). Then, these poses are used for building an offline map with the NeuralMapper and, after that, the localization module is used for estimating the localization of the vehicle in relation to the NeuralMapper. The ground truth poses are obtained in a second step of optimization using the GraphSLAM. In this second step of optimization, besides the data from GPS, odometry and LiDAR (loop closures), the localization is also used as input for the method. By doing so, we encourage consistency with the map while using the sensors data to correct local localization errors.

D. *Metrics*

A metric for evaluating the trained models is the average accuracy of the classes in the DNN output. The classes predicted for each cell are compared with the ground truth and are defined in Equation (1):

$$A = \frac{VP}{TOTAL} \tag{1}$$

where VP is the total number of cells whose classes were correctly predicted and TOTAL is the total number of cells in the database, VP and TOTAL consider all images in the database.

Another metric used to analyze the results during test was the confusion matrix. Each matrix line represents the classes present in the ground truth, while the columns represent the inferences of the network. Thus, it is possible to visually inspect for what classes the prediction is better or worse. The values of the confusion matrix are all percentages of the total of each class and all lines add up to 100% for each class. The diagonal of the matrix represents the correctness of the network regarding the ground truth.

Additionally, we performed a qualitative evaluation of the offline map generated with the DNN with the IARA self-driving car in a real-world environment in autonomous navigation mode.

The metric for evaluating the localization was the same used in [22] and [23]. The poses ($\boldsymbol{x}_{1:n}$) estimated using the localization were compared to the ground truth poses ($\boldsymbol{g}_{1:n}$) obtained as described in Section IV.C. The metric chosen was the root mean squared error (RMSE) given by the Equation (2):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\boldsymbol{x}_i^x - \boldsymbol{g}_i^x)^2 + (\boldsymbol{x}_i^y - \boldsymbol{g}_i^y)^2}{n}} \tag{2}$$

where n is the number of estimated poses associated with sensor data, and the subscripted x and y represent the respective coordinates of the poses.

We also present the standard deviation of the error along with the percentage of samples in which the localization error is smaller than a threshold. The thresholds considered are 0.2m, 0.5m, 1.0m, and 2.0m.

V. Results and Discussions

To evaluate the performance of the NeuralMapper, we trained the DNN using the Dataset 1 for 50 epochs. Our model achieved an accuracy of 76.5% on the validation set and a loss of 0.25 on the training set. Figure 5 shows the training loss and the validation accuracy for each training epoch.

Using the test set from the Dataset 1, the model achieved an average accuracy of 76.48%. TABLE I shows the confusion matrix for this test set. It shows that the network is less precise in classifying Occupied cells, since it correctly classifies just 62.20% of the samples. The network achieved an accuracy of 80% for the unknown cells and 73.45% for the free cells.

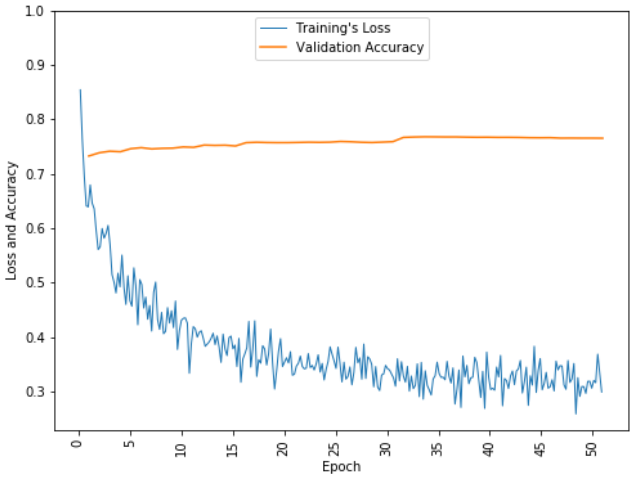

Figure 5 - *L*oss of training (in blue) and accuracy of the validation set (in orange) after each trained epoch.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE II presents the confusion matrix for the test region of Dataset 2. Observe that the test region is the same of Dataset 1, but in a different date. The results show an increase of occupied cells error, but the total average accuracy at this dataset was 73.81%. The test region is challenging due to its features such as different lanes, asphalt and cobbles, in addition to different elevations. That can be seen in the results presented at TABLE III which shows the model confusion matrix when considering all regions from the Dataset 2 (i.e., the training, validation and test regions from Dataset 1). The accuracy achieved was 76.90%. This also shows the model generalization.

Despite the tests showing a low accuracy rate in the occupied class, a qualitative analysis of the predictions show that the network is accurate on the road and that most errors are in external regions. Figure 6 illustrate this fact by presenting a comparison between the neural network prediction and the ground truth. The errors in external regions are expected at some level since most of the laser readings are concentrated in the center.

TABLE I CONFUSION MATRIX OF TEST SET DATASET 1

		Ground Truth		
		Unknown	***Free***	***Occupied***
Predictions	***Unknown***	80.05%	12.87%	8.66%
	Free	13.63%	73.45%	29.15%
	Occupied	6.33%	13.68%	62.20%

TABLE II CONFUSION MATRIX OF TEST SET DATASET 2

		Ground Truth		
		Unknown	***Free***	***Occupied***
Predictions	***Unknown***	79.86%	16.01%	6.51%
	Free	15.01%	68.21%	41.76%
	Occupied	5.13%	15.78%	51.72%

TABLE III CONFUSION MATRIX OF TEST WITH ALL DATASET 2

		Ground Truth		
		Unknown	***Free***	***Occupied***
Predictions	***Unknown***	82.05%	13.57%	7.88%
	Free	12.78%	72.24%	32.52%
	Occupied	5.17%	14.19%	59.60%

It is important to notice that the ground truth considers various LIDAR's scans because it is generated using the OGM. Which also makes it not trivial to select a good metric for the segmentation problem. Figure 6 and Figure 7 compare the output of NeuralMapper and probabilistic occupancy grid mapping from a single LIDAR scan.

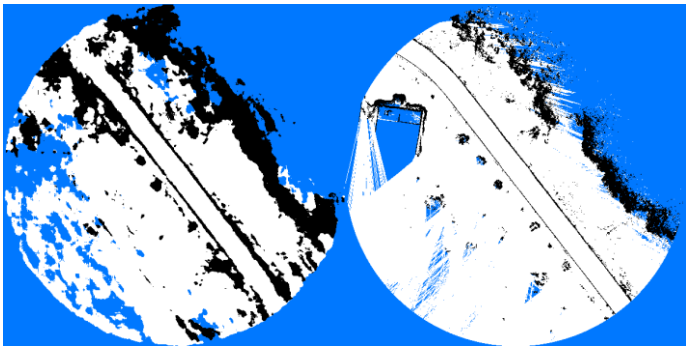

Figure 6 - The left image is the NeuralMapper's Output while the right image is the corresponding ground truth.

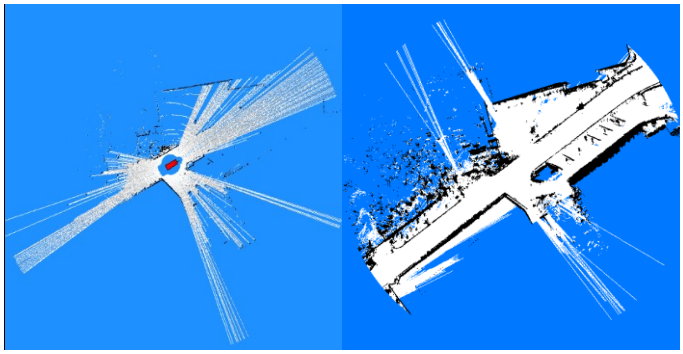

Figure 7 - The left image is the probabilistic Occupancy grid map from one LIDAR scan while the right image is the corresponding offline map also used as ground truth for the NeuralMapper.

The main use of the offline map is for localization purpose. Therefore, we also evaluate if it is possible to use the NeuralMapper for estimating the localization of the self-driving car. For this evaluation, the NeuralMapper and Dataset 1 are used to build the OGM and the Dataset 2 is used to test the localization. Figure 8 shows the Cumulative Distribution Function (CDF) chart achieved by IARA's localization technique [4]. For 92.8% of the samples, the pose error was smaller than 0.5m and in 40.26% of the samples the error was smaller than 0.2m. TABLE IV presents the metrics summarized. The localization achieved a RMSE of 0.28m with a standard deviation of 0.017m. These results are equivalent to the literature using other types of grid maps [4][22][23] and they show that the NeuralMapper can be successfully used for estimating the localization of a self-driving car.

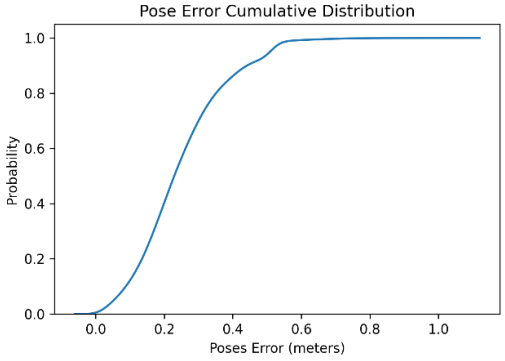

Figure 8 - The CDF from the localization experiment using Dataset 2.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE IV METRICS OF LOCALIZATION USING DATASET 2

RMSE (m)	*STD (m)*	*% < 2m*	*% < 1m*	*% < 0.5m*	*% < 0.2m*
0.28	0.017	100	99.99	92.80	40.26

We validate these results by using the IARA's platform for navigation in autonomous mode. The video (http://tiny.cc/9gqejz) shows the IARA's systems operating with the NeuralMapper being used by the localization and the planning modules. The planning module receives as input a map that is obtained by merging the information from the NeuralMapper with instantaneous data captured by the LiDAR. The car managed to navigate and maintain itself inside the correct lane during all the experiment. It also coped successfully with moving obstacles. This result show that the NeuralMapper can be used for both localization and planning.

Some minor oscillations in the car's trajectory can be observed in parts of the video. These oscillations can be explained by the inner workings of the motion planner module [8]. It depends on a precise localization to keep the car in the planned path it acts to compensate inconsistencies in the planned and executed trajectories. As shown in TABLE IV, in some cases the localization error is larger than a map cell (i.e., 0.2m x 0.2m). Therefore, due to these errors, the planning module may act to compensate incorrectly identified inconsistencies in the trajectory which results in oscillations in the path followed by IARA.

VI. CONCLUSIONS AND FUTURE WORK

The main motivation of this work was to replace the occupancy grid mapping (OGM) algorithm with neural networks given their capability to learn how to handle non-linearities direct from data, and due to its potential to reduce hundreds of lines of code.

The spatial discontinuity generally presented in OGMs, shows the potential of the network to perform even better results than the probabilistic mapping. This is because the Bayesian mapping used today, despite temporarily filtering noise in the cells, does not use information from neighboring spaces to estimate occupation. However, it is reasonable to believe that for common objects, if all the cells around the one is occupied, that one will be occupied as well. In the other hand, this spatial continuity is naturally embedded in convolutional neural networks.

As the results show, even though the semantic mapping results appear less accurate than the necessary, they are sufficient for localization given the preserve of local structure around the car and that can also be confirmed by the localization RMSE metric. In addition, the qualitative results show IARA running in autonomous mode with the NeuralMapper. In this way, the current OGM algorithm could be replaced by a deep neural network, which uses examples to learn the task.

Furthermore, our approach can be used with other kinds of grid maps, for instance, reflectivity, color, and multi-object semantic grid maps, allowing, in those cases, the possibility to include more information on the DNN input.

In future works, it is essential to experiment with larger datasets and different architectures that use geometric transformation and minimizes preprocessing.

ACKNOWLEDGMENT

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

REFERENCES

[1] C. D. Crane, The 2005 DARPA Grand Challenge, vol. 36. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[2] C. Badue et al., "Self-Driving Cars: A Survey," arXiv preprint arXiv:1901.04407v2. 2019.

[3] F. Mutz, L. P. Veronese, T. Oliveira-Santos, E. de Aguiar, F. A. Auat Cheein, and A. Ferreira De Souza, "Large-Scale Mapping in Complex Field Scenarios Using an Autonomous Car," Expert System and Applications, vol. 46, pp. 439–462, Mar. 2016.

[4] L. de P. Veronese et al., "A light-weight yet accurate localization system for autonomous cars in large-scale and complex environments," in *2016* IEEE 19th International Conference on Intelligent Transportation Systems *(ITSC)*, pp. 520–525, 2016.

[5] R. Sarcinelli et al., "Handling pedestrians in self-driving cars using image tracking and alternative path generation with Frenét frames," Comput. Graph., vol. 84, pp. 173–184, Nov. 2019.

[6] L. C. Possatti, R. Guidolini, V. B. Cardoso, R. F. Berriel, T. M. Paixão, C. Badue, A. F. De Souza, and T. Oliveira-Santos, "Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars", in IEEE International Joint Conference on Neural Networks (IJCNN), 2019.

[7] L. T. Torres, T. M. Paixão, R. F. Berriel, A. F. De Souza, C. Badue, N. Sebe, and T. Oliveira-Santos, "Effortless Deep Training for Traffic Sign Detection Using Templates and Arbitrary Natural Images", in IEEE International Joint Conference on Neural Networks (IJCNN), 2019.

[8] V. Cardoso et al., "A Model-Predictive Motion Planner for the IARA Autonomous Car," in 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 225–230.

[9] R. Guidolini, C. Badue, M. Berger, L. de P. Veronese, and A. F. De Souza, "A Simple Yet Effective Obstacle Avoider for the IARA Autonomous Car," in 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 1914–1919.

[10] R. Guidolini, A. F. De Souza, F. Mutz, and C. Badue, "Neural-based model predictive control for tackling steering delays of autonomous cars," in 2017 International Joint Conference on Neural Networks (IJCNN), 2017, vol. 2017-May, pp. 4324–4331.

[11] L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde, "Fast LIDAR-based road detection using fully convolutional neural networks," in 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, no. Iv, pp. 1019–1024.

[12] V. Santos, J. G. M. Goncalves, and F. Vaz, "Perception maps for the local navigation of a mobile robot: a neural network approach," in Proceedings of the 1994 IEEE International Conference on Robotics and Automation, 1994, no. pt 3, pp. 2193–2198.

[13] V. Gupta, G. Singh, A. Gupta, and A. Singh, "Occupancy grid mapping using artificial neural networks," in 2010 International Conference on Industrial Electronics, Control and Robotics, 2010, pp. 247–250.

[14] R. Weston, S. Cen, P. Newman, and I. Posner, "Probably Unknown: Deep Inverse Sensor Modelling Radar," in 2019 International Conference on Robotics and Automation (ICRA), 2019, vol. 2019-May, pp. 5446–5452.

[15] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud," in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 1887–1893.

[16] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354–3361, 2012.

[17] E. E. Aksoy, S. Baci, and S. Cavdar, “SalsaNet: Fast Road and Vehicle Segmentation in LiDAR Point Clouds for Autonomous Driving,” Sep. 2019.

[18] M. Teichmann, M. Weber, M. Zollner, R. Cipolla, and R. Urtasun, “MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving,” in 2018 IEEE Intelligent Vehicles Symposium (IV), vol. 2018-June, no. Iv, pp. 1013–1020, 2018.

[19] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” in 4th International Conference on Learning Representations (ICLR), arXiv preprint arXiv:1511.07289, 2016.

[20] P. Y. Simard, D. Steinkraus, and J. C. Platt, “Best practices for convolutional neural networks applied to visual document analysis.” In Seventh International Conference on Document Analysis and Recognition, pp. 958, 2003.

[21] J. Wang, and L. Perez, “The effectiveness of data augmentation in image classification using deep learning”. in Convolutional Neural Networks Vis. Recognit, pp. 11, 2017.

[22] J. Levinson, and S. Thrun, “Robust vehicle localization in urban environments using probabilistic maps”. In 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 4372-4378, 2010.

[23] R. W. Wolcott, and R. M. Eustice, “Robust LIDAR localization using multiresolution Gaussian mixture maps for autonomous driving”. The International Journal of Robotics Research, Vol. 36, Num. 3, pp. 292-319. 2017.