Multi-Step Prediction of Occupancy Grid Maps with Recurrent Neural Networks

Nima Mohajerin and Mohsen Rohani*

Huawei Noah's Ark, Markham, ON, Canada
{nima.mohajerin, mohsen.rohani}@huawei.com

Abstract

*We investigate the multi-step prediction of the drivable space, represented by Occupancy Grid Maps (OGMs), for autonomous vehicles. Our motivation is that accurate multi-step prediction of the drivable space can efficiently improve path planning and navigation resulting in safe, comfortable and optimum paths in autonomous driving. We train a variety of Recurrent Neural Network (RNN) based architectures on the OGM sequences from the KITTI dataset. The results demonstrate significant improvement of the prediction accuracy using our proposed **difference learning** method, incorporating motion related features, over the state of the art. We remove the egomotion from the OGM sequences by transforming them into a common frame. Although in the transformed sequences the KITTI dataset is heavily biased toward static objects, by learning the difference between consecutive OGMs, our proposed method provides accurate prediction over both the static and moving objects. A video of the performance of our method on the KITTI dataset is available at* https://youtu.be/Bskd0Z7eLFE.

1. Introduction

Determining the environment state is a crucial ability for autonomous vehicles to have. Particularly for path planning and navigation, the state of the environment is required to determine safe areas to drive, that is, the drivable space. In a driving scenario, the classic approach is to detect and track objects, and based on the state of the tracked objects, determine the drivable space [21, 5]. Figure 1(a) illustrates the general pipeline of the classic approach. As the new data comes in, the objects in the environment are detected (and classified). To keep track of objects, each object is given an ID. The Data Association module assigns existing IDs to the detected objects, or initiates new IDs if there is no matched tracks. The states of the tracked objects are updated using

---

*The authors contributed equally.
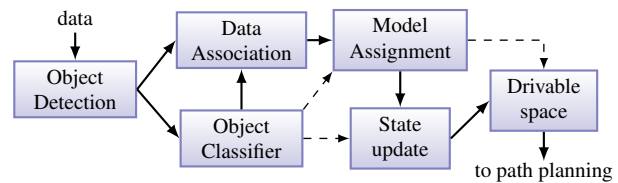
(a) Classic approach for drivable space detection.
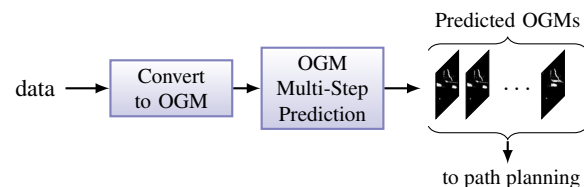
(b) OGM multi-step prediction.

Figure 1. Classic approach versus ours for providing the drivable space to a motion planner in autonomous driving.

models which are assigned to them based on their class. Finally, the updated states (possibly along with the assigned models) are used to determine the drivable space.

The various stages in the classic approach require hand-engineered features. Additionally, any error occurred at an early stage of the pipeline can propagate towards the next stages leading to an overall performance degradation. An alternative approach is to predict the drivable space at a level closer to the sensory information, requiring fewer abstraction levels. However, for such prediction to be useful, first a proper representation of the drivable space should be chosen which can be directly obtained from sensor measurements. One such proper means in this context is an Occupancy Grid Map (OGM) [3, 24]. An OGM divides the space around the ego-vehicle into equal cells which represent the occupancy state of surrounding regions, i.e., free or occupied. The state of the cells can be constructed using Laser range finders (and images [22]).

Given an OGM, methods such as Probabilistic Road-Maps and Rapidly exploring Random Trees can be employed to generate a collision-free path [24]. In the presence of dynamic objects, however, a single OGM is insufficient

for guaranteeing a safe passage in general, as it does not properly represent the moving objects. Planning based on the updated OGMs results in inefficient and uncomfortable paths requiring frequent steering corrections or even inadmissible paths. In such scenarios, one common approach is to "grow" the occupied regions by padding the occupied cells, which artificially creates a safe distance from obstacles. However, the amount of padding is difficult to determine a priori and depends on the objects' type, velocity, etc. Without proper object detection and classification, the padding size has to be chosen above a safe threshold which may result in severely restricting the ego-vehicle movement in crowded areas, such as urban environments, and failing to detect useful drivable spaces in highway scenarios where the objects may move relatively fast.

Alternatively, it is possible to provide multi-steps of OGM prediction into the future [2, 4, 10]. In a dynamic environment, such predictions can provide the drivable space to a planning algorithm without the need for the several stages required in the classic approach. The idea is to encode the observed states of the environment, represented as OGMs, into latent variables of a recursive model, from which the future OGMs can be produced recursively. Recurrent Neural Networks (RNNs) are, therefore, a potential model candidate, which have also been successfully applied to various other problems with dynamic nature, such as unsupervised video prediction [12, 7], multi-step prediction of mobile robots [15] and visual odometry [25]. Moreover, it is shown that RNNs can produce promising results in the context of OGM prediction [2, 4].

OGMs may also be looked upon as black-and-white images. However, predicting images in a video stream, i.e., video prediction [7, 13, 23], can be studied either as a regression problem, or multinomial classification (with at least 256 classes for gray-scale images), where OGM prediction is a binary classification problem. As a sanity check, we employed PredNet [12] and the CDNA model [7], but both methods failed to provide a reliable OGM prediction. Particularly, the occlusions that occur within OGMs, due to the nature of range finder sensors, are difficult to be dealt with in the video prediction task where the primary goal is to predict the future images as similar as possible to the ground truth. Therefore, in this work we do not consider the multi-step prediction of OGMs as a video prediction task.

In [2], a network of Gated Recurrent Units [1] (GRUs), with dilated convolutions, is used whose states are initialized as it receives and reconstructs observed OGMs, for a limited number of steps. Then, the initialized states are employed to generate next OGMs, while the input OGMs are blank. Inspired by [2], we propose an approach to learn the *difference* between consecutive OGMs as a compensation matrix. The current OGM, modified by the compensation matrix, provides the features for a predictive classifica-

tion. This choice stems from the fact that the current OGM provides a reliable prior for the drivable space. The major source of discrepancy between consecutive OGMs is the egomotion, which presumably is available (e.g., from the Drive-By-Wire system or GPS). Therefore, in this work, the observed OGMs are transferred to a common frame using egomotion information. The common frame corresponds to the ego-vehicle coordinate system at the present time, i.e., the frame at which we want to predict future OGMs.

The OGMs are also predicted into the future with respect to the common frame. In fact, in our approach the drivable space is predicted as OGMs in a frame which is absolute over the observed history and the prediction period, but is attached to the ego-vehicle. This is different to the approach taken in [2] where a Spatial Transformer Module (STM) [11] is employed to take account for the egomotion. Since our intention is to predict OGMs suitable for planning, it is reasonable to predict the OGMs regardless of the *future* egomotion. Providing the environment state in an absolute frame makes it possible for a planning algorithm to examine and plan based on virtually all ego trajectories that are safe, i.e., those which do not interfere with other objects' paths in the environment.

Training a network for OGM prediction is supervised as the target output is the OGM. However, the targets directly come from the sensors and do not need human supervision to be generated. Such datasets are easy to collect which significantly reduces the human labor and improves the time-to-market. One difficulty is that the number of static objects is quite often dominant in OGM datasets. Consequently, the network can easily trap in a local minimum where it assumes the surrounding environment is basically static. Learning the difference between OGMs partially prevents such local minima as the network only needs to compensate for the moving objects. Additionally, we propose to incorporate motion relevant information as inputs to the networks, either by using standard optical-flow extraction algorithms such as the Farneback method [6], or by feeding the OGM differences. Both methods are considered and evaluated for performance and computation time.

Our contributions are summarized as follows,

- A novel RNN-based architecture is proposed to predict OGMs over multi-steps into the future, based on learning the OGM differences between consecutive frames,
- Motion related features are employed which enhances OGM prediction over dynamic objects significantly,
- A conclusive comparison between various RNN-based architectures is presented for the problem at hand.

In the following, relevant works are reviewed in Section 2. Section 3 formulates the OGM prediction problem and describes our methods to address it. Section 4 presents the experimental results and discussions. We conclude the paper in Section 5.

2. Related Work

In [20], the authors propose a deep tracking scheme, where a simple RNN is leveraged to learn OGM-to-OGM mappings. The training data consists of OGMs that are built by synthesized laser readings collected from a static sensor. This work is improved and expanded in [2], where the egomotion is handled with STMs, and convolutional GRUs, with dilation, replace the vanilla RNN architecture. However, the two approaches in [20] and [2] are similar: a sequence of observed OGMs, followed by blank OGMs, are fed to an RNN, whose output is trained on the observed OGMs for the entire sequence. Therefore, the RNN acts as an autoencoder while it receives the observed OGM, to initialize its states. The accumulated information in the states is employed to predict the OGM evolution as the network receives blank OGMs. However, feeding blank inputs cause the output OGMs to fade out rapidly. Also, approaches in [20] and [2] become computationally inhibitive as the OGM size increases unless the OGMs are either resized or an encoder-decoder architectures is employed.

Authors in [10] and [4] employ Dynamic OGMs (DOGMa). DOGMa is the result of fusing a variety of sensor readings using Bayesian filtering, which associates dynamic information to each cell as well as the occupancy state [18]. The dynamic information contains the velocity and its uncertainty. An encoder-decoder structure with Convolutional Long-Short-Term-Memory network (ConvLSTM) [27] receives DOGMa and produces the occupancy probability of static regions alongside the anchor boxes for dynamic objects [4]. An automatic output label generation is also used which can have a "relatively high false negative rate" [10]. Additionally, the datasets in [10] and [4] are collected from a static sensor (parked car).

OGM prediction has already been used in path planning of autonomous robots. In [17], Variable Length Markov Model is employed to predict the OGM in an environment whose main occupants are humans. However, in [17], the experiments are restricted to cases where there is only one human exists in the vicinity of the autonomous robot. Similarly in [19], multi-steps of predicted OGMs are stacked in an XYT space (T stands for time) which is then used for path planning of rescue robots. The XYT stack of predicted OGMs is also used in other works, such as [9], however, the OGMs are either updated using object models (classic approach) or the main occupant are of the same object type, e.g. human in [17] and [19].

3. OGM Prediction in Autonomous Driving

In practice, an OGM cell can be either fully occupied, partially occupied or completely free. However, in this work, it is assumed that a cell can be either fully occupied or free, with a probability of occupancy associated to it. Formally, the state of the cell located at i^{th} row and j^{th} column of the OGM, at time-step k, is represented by a binary random variable $c_k(i,j) \in \{0,1\}$. For now, assume that the OGM at time-step k, $\mathbf{O}_k$, is the probability matrix of $c_k(i,j)$, that is,

$$\mathbf{O}_k = \Big[p\big(c_k(i,j)\big) \Big] = \begin{bmatrix} p\big(c_k(1,1)\big) & \dots & p\big(c_k(1,X)\big) \\ \vdots & \ddots & \vdots \\ p\big(c_k(Y,1)\big) & \dots & p\big(c_k(Y,X)\big) \end{bmatrix},$$
(1)

where X and Y indicate the grid size. In order not to lose generality, we assume that at each time-step, $c_k(i,j)$s, where $i = 1,...,Y$ and $j = 1,...,X$, are independent with possibly different distributions. However, $c_k(i,j)$s are dependent over time. In general, the probability of $c_k(i,j)$ is conditioned on the set of random variables $C_{i,j}$,

$$\begin{aligned} C_{i,j} = \big\{ c_v(m,n) | v &= k-1, k-2, ... \\ m &= i - \alpha_i, ..., i + \beta_i, \\ n &= j - \alpha_j, ..., , j + \beta_j, \\ 0 &\le \alpha_i < i, 0 \le \beta_i \le X - i, \\ 0 &\le \alpha_j \le j, 0 \le \beta_j < Y - j \big\}, \end{aligned}$$
(2)

where α_i, β_i and α_j, β_j are integers and indicate some neighborhood around i and j. In general, the values α_i, β_i and α_j, β_j depend on a number of factors, such as the ego-vehicle's velocity and/or the speed of the surrounding objects. Therefore, we assign the extreme possible values to them, in which case, (m,n) covers the entire OGM. Therefore, the elements of OGM matrix in (1) are, in fact, conditional probabilities,

$$p\big(c_k(i,j)|\mathcal{C}_{k-1}, \mathcal{C}_{k-2}, ...\big),$$
(3)

$$\mathcal{C}_k = \big\{ c_k(m,n) | m = 1,...,Y; n = 1,...,X \big\},$$
(4)

and we define the OGM as the matrix of the *conditional* probabilities (3),

$$\mathbf{O}_k = \Big[p\big(c_k(i,j)|\mathcal{C}_{k-1}, \mathcal{C}_{k-2}, ...\big) \Big], i = 1,...,Y; j = 1,...X.$$
(5)

Our goal is to predict the OGMs, as depicted in (5), over multi-steps into the future. To this end, we employ and train a variety of RNN-based architectures. The feedback connections in an RNN are particularly useful to establish the recursive dependency represented in (5). The RNN-based models in this work are employed as sequence-to-sequence maps, i.e., they receive a sequence of input OGMs and produce a sequence of OGMs, of the same length. Therefore, given a fixed sequence length, T, a *prediction task* is defined as feeding the input OGM sequence, $\mathbb{O}$, to an RNN-based model which generates the output OGM sequence, $\hat{\mathbb{O}}$,

$$\mathbb{O} = \{\mathbf{O}_k\}, k = 1,...,T,$$
(6a)

$$\hat{\mathbb{O}} = \{\hat{\mathbf{O}}_k\}, k = 1,...,T,$$
(6b)

where $\hat{\mathbf{O}}_k$ is the model output at time-step k whose form depends on the model architecture. Based on [2] we initially form the input sequence as follows,

$$\mathbf{O}_k = \begin{cases} \mathbf{O}_k^*, k = 1, ..., \tau_I, \\ [\mathbf{0}], k = \tau_I + 1, ...T, \end{cases} \quad (7)$$

where $\mathbf{O}_k^*$ is the *observed* OGM at k and $[\mathbf{0}]$ represents a matrix of zeros whose size is the same as $\mathbf{O}_k^*$, i.e., a blank OGM. Later we will modify the input sequence. Equation (6) and (7) indicate that a prediction task consists of two phases. During the first phase, where $k = 1, ..., \tau_I$, the observed OGMs are given to the model. The first phase is mainly intended for RNN state initialization [16], and therefore, we will refer to it as the initialization phase, or init-phase for short. The second phase, namely the prediction phase, starts at $k = \tau + 1$ and is intended for multi-step prediction, as the input OGMs are blank. The length of the prediction phase is $\tau_P = T - \tau_I$.

It is worthwhile to mention that in [2], authors train their network to reconstruct the input during the init-phase, therefore, their architecture behaves similar to an autoencoder. From our point of view, such approach does not necessarily result in extracting features that are useful for prediction, rather, those features are useful for reconstructing the inputs. In contrary, our models are trained to generate one-step-ahead prediction during the init-phase. Therefore, we force the models to learn useful features for prediction.

3.1. Base Architecture

To handle large OGMs, an encoder/decoder architecture is employed, as depicted in Figure 2. The encoder module consists of a number of convolutional layers, each followed by a max pooling, to reduce the input OGM size and extract features. Then, the features are passed through an RNN, followed by a decoder module. The decoder module upsamples the RNN output using transposed convolutional layers. The number of layers in the encoder and the decoder is the same. Also, the decoder's output has the same XY dimension as the input OGM.
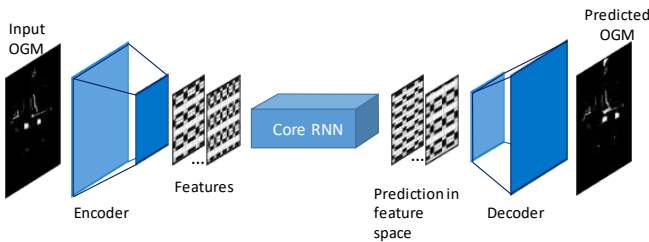
Figure 2. Base Architecture.

To account for the egomotion, the OGMs are transferred to the frame at which the prediction starts, i.e., $k = \tau$. We use standard geometric image transformation using the dis-

placement in x, y and the heading directions over consecutive frames. This transformation is applied to all of the OGMs in each prediction task. As a result, the OGMs (observed and predicted per each prediction task) are treated as if they are seen from a stationary sensor whose location is the same as the sensor location at $k = \tau$. To avoid notation clutter, we use the same notation for the OGMs as before, keeping in mind that they are all transformed accordingly.

3.2. Extended Architectures

We extend the base architecture by feeding the output back to the network. Therefore, (7) becomes,

$$\mathbf{O}_k = \left[1 - u(k - \tau - 1)\right]\mathbf{O}_k^* + u(k - \tau - 1)\hat{\mathbf{O}}_k, \quad (8)$$

where $\hat{\mathbf{O}}_k$, is the predicted OGM at time-step k and $u(.)$ is the standard step function[1]. The output feedback connection incorporates the information, that are coded inside the decoder module, back into the network.

The second extension is to include motion-related features using Motion-Flow (MF) extraction algorithms (e.g., Farneback algorithm [6]), or a *two-channel difference* method. An MF extraction algorithm, receives two consecutive OGMs and provides a tensor of the same size and depth of two. Each of the two channels correspond to the movement in X and Y directions. The two-channel difference method takes the difference between two past consecutive OGMs and places the result into two channels; one channel for the positive values and the other for (the absolute value of) the negative values. The resulting two-channel tensor, from either of the methods, is then passed through an encoder module, whose output is stacked with the output of the OGM encoder. The extended architectures is illustrated in Figures 3.
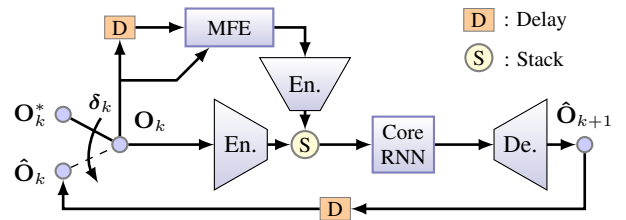
Figure 3. Extensions over the base architecture. "MFE" stands for Motion-related Feature Extraction. The switch δ_k changes position at $k = \tau_I + 1$. "En." and "De." stand for Encoder and Decoder, respectively.

3.3. Difference Learning Architecture

With the current technology, high resolution OGMs can be constructed at 10 Hz rate and faster. In a normal driving scenario, it is quite unlikely that the consecutive OGMs at

---

[1] $u(x)$ returns 1 for $x \geq 0$ and 0 otherwise.

such rates differ dramatically. In other words, current OGM provides a reasonable prior for predicting the next OGM.

We embed the above idea into a *difference learning* architecture, which is based on a two-stage process as illustrated in Figure 4. The first stage, depicted by **Diff. Learn**, generates a *compensation matrix*, $\Delta\mathbf{O}_k$, whose size is the same as the input OGM. The **Diff. Learn** module employs the extended architecture depicted in Figure 3. The result of adding $\Delta\mathbf{O}_k$ to the input OGM provides the basis on which the **Classifier** module produces the next OGM prediction. Therefore, the **Diff. Learn** module should implicitly distinguish between the static and dynamic objects which is then reflected in $\Delta\mathbf{O}_k$. The elements of $\Delta\mathbf{O}_k$ are real values in $[-1, 1]$. A value near zero corresponds to a cell whose occupancy should not be altered, i.e., a free cell or a cell occupied by a static object). Similarly, a value closer to 1 (or -1) attempts to add (or clear) occupancy to (from) the corresponding cells, indicating the cell is being occupied by (or freed from) a dynamic object. The **Classifier** module refines the modified OGM further to predict the next OGM, i.e., $\hat{\mathbf{O}}_{k+1}$. Figure 5 illustrated the detailed difference-learning model. Note that the **Classifier** module can be a simple feed-forward or a recurrent network.
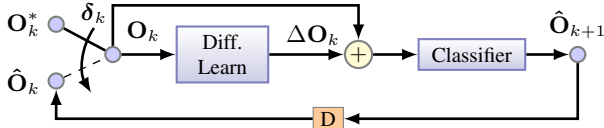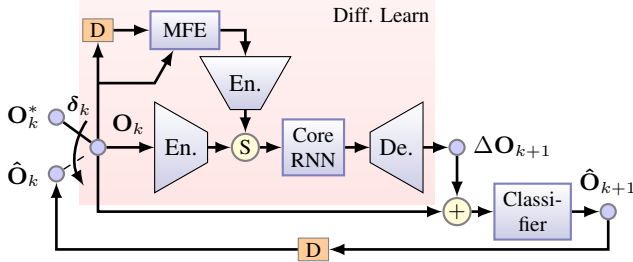

Figure 4. Difference OGM learning model.

Figure 5. Detailed architecture for the difference learning model.

3.4. Training Cost

As the problem at hand is classification (occlusion vs. free), we employ pixel-wise Cross-Entropy cost between the output, $\hat{\mathbf{O}}_k$, and the target, $\mathbf{O}_k^*$. Because the majority of the cells are normally unoccupied, the cost is normalized with respect to the number of occupied/free cells. Also, the pixel-wise cost is multiplied by the visibility matrix as in [2], to let the network handle occlusions.

To make sure the **Diff. Learn** module is effectively employed, two extra terms are added for the difference learning models. First, based on the idea that most of the objects are static in a driving scenario, an L_2 norm on the compensation matrix, $\Delta\mathbf{O}_{k+1}$, is added to the total cost. This

term ensures that the non-zero elements inside $\Delta\mathbf{O}_{k+1}$ correspond to the cells whose occupancy should be changed during the prediction task, i.e., the dynamic objects. To force the classifier inputs towards the target, a multi-scale Structural Similarity Index Metric [26] (SSIM) between the summation result and the target OGM is added to the cost.

4. Experimental Results

The OGM sequences are obtained from the KITTI raw dataset [8]. The raw Lidar point-clouds, collected at 10Hz, are converted to bird's eye view OGMs (with ground removal) [14]. Then, the OGM sequences are partitioned into segments of 20 frames, preserving the order within each segment. Every segment corresponds to 2 seconds of OGM measurement and is regarded as one sample. The init-phase length is set to 10 steps, i.e., $\tau = 10$. Since the models' ability to handle occlusion is important to us, in addition to the natural occlusions that exist in the raw data, we impose a constant occlusion pattern over the OGMs by filtering out occupancies outside the front camera field of view (FOV). With a cell size equal to $20(\text{cm})\times20(\text{cm})$, each OGM frame consists of 256×256 cells, arranged in a square matrix, corresponding to occupancies over 50 meters in front of the car. The OGMs are visualized as 256×256 pixel, black and white or gray-scale, images, where the former shows the actual occupancy and the latter illustrates the discretized occupancy probabilities. Each pixel represents one OGM cell.

Table 1 lists the models, and their configurations, trained for this work. The ED (Encoder-Decoder) models provide a basis for evaluation. The ED-Di (Encoder-Decoder with Dilated convolutions) models correspond to the model proposed in [2] along with two improvements based on using the motion-related features. The Diff.1 and Diff.2 models represent our proposed difference learning architecture.

The encoders' architecture is identical across the models; they produce $64 \times 64 \times 32$ features from $256 \times 256 \times n$ inputs, where $n = 1$ for the OGM and $n = 2$ for motion-flow related features. The decoders are also similar across the models; they convert back the $64 \times 64 \times 32$ tensors to $256 \times 256 \times n$, where $n = 1$ for Diff.1 and Diff.2 models, and $n = 2$ for ED and ED-Di models. Also, the decoders in Diff.1 and Diff.2 models employ a $\tanh()$ activation function on their last layer to produce compensation matrix values in $[-1, 1]$ while the decoders in ED and ED-Di models employ a linear activation to produce logits. The Core RNN module consists of 4 ConvLSTM layers, where dilation at the kth ConvLSTM layer is equal to k for the ED-Di model and one for the rest. Therefore, in our settings, the ED-Di model at #4 row in Table 1 is an attempt to replicate the model proposed in [2]. Models #5 and #6 represent our suggested improvements on #4. All of the models are trained to predict one-step-ahead during the init-phase.

Table 2 lists the results. We evaluate the prediction accu-

#	Model	Cfg.	Arch.	Input	MFE	Di.	Diff. Learn	Classifier
1		Base	Fig. 2	Eq. (7)	N/A	No	N/A	N/A
2	ED	Ext1.	Fig. 3	Eq. (8)	Farenback	No	N/A	N/A
3		Ext2.	Fig. 3	Eq. (8)	Two-channel difference	No	N/A	N/A
4		Base	Fig. 2	Eq. (7)	N/A	Yes	N/A	N/A
5	ED-Di	Ext1.	Fig. 3	Eq. (8)	Farenback	Yes	N/A	N/A
6		Ext2.	Fig. 3	Eq. (8)	Two-channel difference	Yes	N/A	N/A
7		Base	Fig. 4	Eq. (8)	N/A	No	Base (Fig. 2)	2 Layer Conv.
8	Diff.1	Ext1.	Fig. 5	Eq. (8)	Farenback	No	Shaded area in Fig. 5	2 Layer Conv.
9		Ext2.	Fig. 5	Eq. (8)	Two-channel difference	No	Shaded area in Fig. 5	2 Layer Conv.
10		Base	Fig. 4	Eq. (8)	N/A	No	Base (Fig. 2)	1 ConvLSTM
11	Diff.2	Ext1.	Fig. 5	Eq. (8)	Farenback	No	Shaded area in Fig. 5	1 ConvLSTM
12		Ext2.	Fig. 5	Eq. (8)	Two-channel difference	No	Shaded area in Fig. 5	1 ConvLSTM

Table 1. The models trained in this work. ED models constitute the base line over each configuration (Cfg.), ED-Di models are the closest replica of the model proposed in [2], in our framework, Diff.1 and Diff.2 models represent our proposed models.

Architectures		Whole OGM						Objects only						t/f (ms)	Ord.
Model	Cfg.	Whole seq.			Prediction seq.			Whole seq.			Prediction seq.				
		TP	TN	S_{100}	TP	TN	S_{100}	TP	TN	S_{100}	TP	TN	S_{100}		
	Base	87.51	97.85	93.91	85.37	97.38	92.95	77.07	97.88	95.71	70.39	97.56	95.01	**3.8**	11
ED	Ext.1	87.87	87.29	93.75	85.25	99.07	95.49	77.19	98.94	96.79	70.28	98.59	96.02	9.2	12
	Ext2.	89.48	98.69	94.85	87.50	98.21	94.31	78.95	98.77	96.55	72.76	98.39	95.77	4.2	5
	Base	86.16	97.41	93.89	86.01	97.68	93.10	77.83	98.01	95.91	71.84	97.89	95.14	4.0	9
ED-Di	Ext.1	88.09	89.81	94.02	85.89	99.10	96.92	78.24	98.55	97.01	71.25	98.88	**97.03**	9.8	7
	Ext.2	88.60	97.90	94.57	87.15	97.52	95.52	76.52	96.26	95.25	69.42	97.93	93.87	4.7	10
	Base	85.39	99.35	95.35	84.01	98.89	94.26	78.90	**99.34**	**97.44**	72.06	98.92	96.39	4.0	6
Diff.1	Ext.1	88.58	99.07	97.11	85.03	98.29	97.85	79.80	99.12	97.12	77.02	98.56	97.00	10.2	3
	Ext.2	88.14	97.86	98.10	86.67	98.33	97.12	79.96	98.89	95.99	74.41	97.92	96.28	5.7	4
	Base	85.37	99.39	95.55	81.60	99.09	94.45	73.37	99.30	97.39	69.91	**99.00**	96.57	4.3	8
Diff.2	Ext.1	**91.02**	**99.69**	**99.47**	88.36	**99.28**	**98.41**	**82.97**	98.81	97.37	**79.91**	98.37	96.73	12.1	1
	Ext.2	*90.31*	*98.53*	*98.97*	***88.89***	*98.91*	*97.83*	*81.09*	*99.02*	*97.01*	*77.43*	*98.11*	*96.97*	*6.1*	2

Table 2. The OGM prediction accuracy measured by %True-Positive (%TP), %True-Negative (TN) and S_{100} ($100\times$[standard SSIM]) measures. The decreasing order of prediction accuracy is listed under "Ord.", where 1 corresponds to the best overall accuracy. The bold-faced numbers correspond to the best performance on each column. Within each 3×3 block, the shaded row corresponds to the best performance over the block. The "t/f" column lists the time it takes to generate one frame of prediction using GTX 1080Ti GPU.

racy using %True-Positive (TP), %True-Negative (TN) and S_{100} measures. The S_{100} is the standard SSIM measure multiplied by 100, in order to be on the same scale as TP and TN. The evaluation horizon is either the entire sequence (20 frames) listed under "Whole seq.", or only over the prediction length (the second 10 frames) listed under "Prediction seq.". Also, to evaluate the prediction accuracy over the areas occupied by the objects only, we employ the tracklets in the KITTI dataset to mask the target and the predicted OGMs and then compute the measures. The obtained results are listed under "Objects only" and partially reflect the prediction accuracy of dynamic objects. Recall that the labels and/or tracklets are not used for training.

Over each 3-by-3 block the shaded row highlights the result with the highest sum (TP+TN+S_{100}), reflecting the best performance among the three corresponding configurations. The last column (titled Ord.) lists the order at which the prediction accuracy decreases, that is, number 1 corresponds to the best and 12 to the worst performance, respectively.

Based on the results, our proposed difference learning model, using Farenback algorithm to extract motion-flow features, (model #11) outperforms the other models, however, it is also computationally the most expensive one. If

we switch to the two-channel difference method (model #12), we gain a considerable improvement over the computation time, while losing a small amount of accuracy. Considering online deployment, model #12 can be readily employed onboard of an autonomous vehicle as it takes about 60ms to predict OGM over 1 second into the future. Additionally, it is clear that inclusion of the motion-related features improves the prediction accuracy for all of the models we consider in this work.
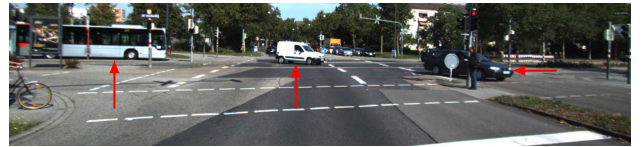

Figure 6. One image from the ego-vehicle POV corresponding to the results shown in Figures 7 and 8. The arrows indicate the moving objects.

To study the predicted OGMs qualitatively, Figure 7 illustrates 10 frames of prediction generated by the four models in their best configuration. A dark red grid is overlaid on every OGM as a reference for the inertial coordinate system. Also, the dashed yellow lines show the border of the FOV. The initial OGMs (under the column "Init-phase") are bi-
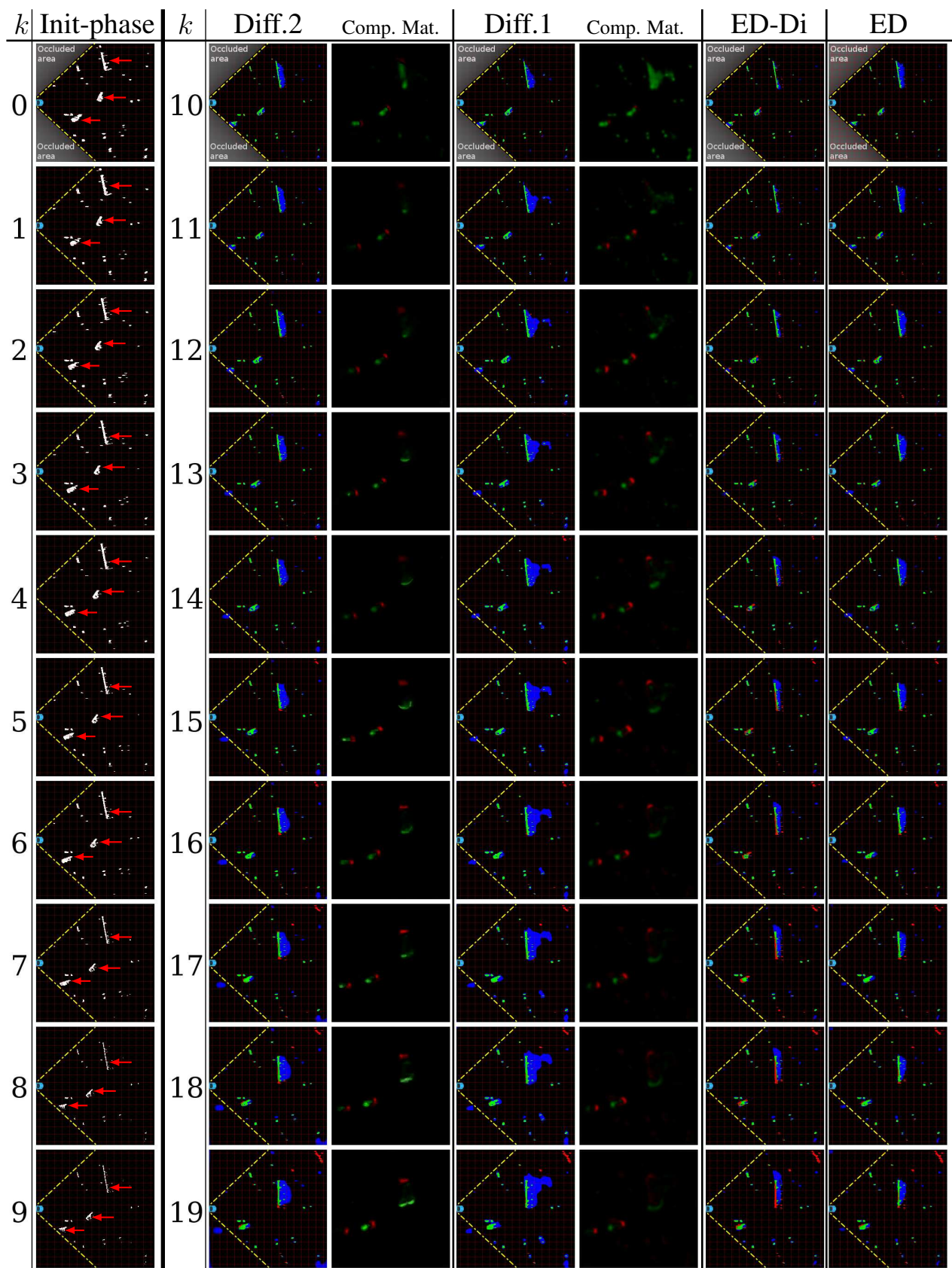
Figure 7. Qualitative results of the four models in their best configuration. The numbers below k indicate time-step.

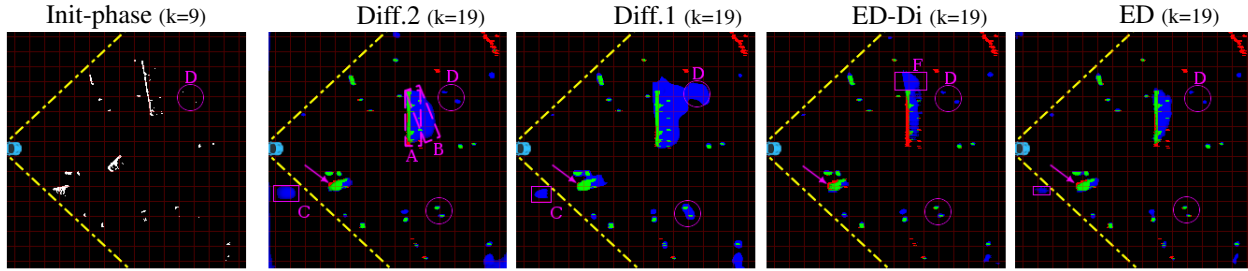| Init-phase (k=9) | Diff.2 (k=19) | Diff.1 (k=19) | ED-Di (k=19) | ED (k=19) |

Figure 8. Last frame of OGM prediction extracted from Figure 7 along with the last frame of initial OGMs.

nary matrices. Therefore, they are illustrated as black and white images. The Init-phase OGMs indicate that there are three moving objects, two cars followed by a bus, highlighted by red arrows. One frame of the actual scene is illustrated in Figure 6. The two cars are turning to their right. However, by the end of the Init-phase, it is difficult to determine the turning direction for the bus.

The generated compensation matrix from Diff.1 and Diff.2 models are illustrated under "Comp. Mat." for each of the two models. In the compensation matrices, green and red pixels correspond to numbers in $(0, 1]$ and $[-1, 0)$, respectively, and black pixels correspond to 0. To assess the prediction quality, the pixels of predicted OGMs are color coded as green, blue and red which correspond to true positive, false positive and false negative results.

The qualitative results confirm the superiority of Diff.2 model over the rest; the moving objects are tracked precisely and occlusions are handled properly. Figure 8 shows the last predicted frame extracted from Figure 7. An important factor to notice is the inadequacy of the ground truth as a target which the prediction should be compared against. In fact, an OGM only shows the *visible* border of the object. Therefore, a false positive is not undesirable everywhere on the OGMs. Over the predicted OGM by Diff.2 in Figure 8, the areas highlighted by the dashed pink rectangles (A and B) show a relatively large area of false positive. Rectangle A shows the border of the bus, grabbed from the tracklets. The ground truth OGM (green and red pixels together) only provides the visible border of the bus, however, intuitively we can say the area inside the rectangle A is fully occupied. In this case, the false positive over A is desirable. The B rectangle shows the other possible direction of the bus.

The box, labeled C, illustrates another area where a false positive is desirable. In fact, this box is out of the FOV where no visible ground truth is provided. However, the network is managed to track the moving object out of the FOV and into the occluded area. In Diff.1 predicted OGM, however, some of the false positive areas are likely undesirable. For instance, the circle D on Diff.1 OGM shows that the areas between two static objects (notice the circle D on the Init-phase OGM) and the bus are joined and the network has failed to properly predict occluded parts of the scene.

The predicted OGM from ED and ED-Di models are

considerably less accurate than Diff.2 and Diff.1. Particularly, the area under the rectangle F (on ED-Di) is wrongly predicted occupied. Also, the occupancies corresponding to the moving bus is not accurately predicted as indicated by the red areas. The ED result is slightly better than ED-Di, however, the occluded areas are not handled as well as Diff.2 model. Diff.1 model is a slightly more "cautious" predictor, among others, due to the more false positive it generated. One interesting difference is highlighted by the arrow, where the object's border is only properly determined by Diff.1 model.

Interesting patterns emerge from the compensation matrices. For instance, looking at the compensation matrices, on Figure 7, for the Diff.2 model over $k > 11$, we may infer that there are three moving objects in the scene, two of which are smaller in length. In fact, when the compensation matrix is applied to an OGM, a green area followed by a red area is a pattern which encourages the occupancies in those areas to move from the red area towards the green area. Therefore, both the dynamic objects and their approximate headings may be interpreted from the compensation matrix. However, since the compensation matrix is a byproduct of our approach, in this work we opt not to employ any inference on the compensation matrix. Illustrating the compensation matrix, however, is a supporting evidence for the difference learning idea.

5. Conclusion

Accurate multi-step prediction of OGMs, as representations of the future drivable space, is useful for path planning algorithms and does not require multi stages of engineered features in the classic approach of object detection and tracking. In this work, a difference learning architecture, based on RNNs, is proposed to predict OGMs multi-steps into the future. We have shown that our proposed architecture outperforms the state of the art in OGM multi-step prediction and is accurate in predicting the static and moving objects as well as handling occlusions. Furthermore, as a future work, the generated features by the compensation matrix from our proposed scheme may provide interesting features for label-less detection and tracking of dynamic objects.

References

[1] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 2

[2] J. Dequaire, P. Ondrúška, D. Rao, D. Wang, and I. Posner. Deep tracking in the wild: End-to-end tracking using recurrent neural networks. *The International Journal of Robotics Research*, 37(4-5):492–512, 2018. 2, 3, 4, 5, 6

[3] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference on Uncertainty in AI*, volume 2929, page 6, 1990. 1

[4] N. Engel, S. Hoermann, P. Henzler, and K. Dietmayer. Deep object tracking on dynamic occupancy grid maps using rnns. *arXiv preprint arXiv:1805.08986*, 2018. 2, 3

[5] A. Ess, K. Schindler, B. Leibe, and L. Van Gool. Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research*, 29(14):1707–1725, 2010. 1

[6] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003. 2, 4

[7] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016. 2

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 5

[9] O. K. Gupta and R. A. Jarvis. Optimal global path planning in time varying environments based on a cost evaluation function. In *Australasian Joint Conference on Artificial Intelligence*, pages 150–156. Springer, 2008. 3

[10] S. Hoermann, M. Bach, and K. Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. *arXiv preprint arXiv:1705.08781*, 2017. 2, 3

[11] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 2

[12] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016. 2

[13] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 2

[14] I. Miller and M. Campbell. A mixture-model based algorithm for real-time terrain estimation. In *The 2005 DARPA Grand Challenge*, pages 407–436. Springer, 2007. 5

[15] N. Mohajerin, M. Mozifian, and S. L. Waslander. Deep learning a quadrotor dynamic model for multi-step prediction. In *Robotics and Automation (ICRA), IEEE International Conference on*. IEEE, 2018. 2

[16] N. Mohajerin and S. L. Waslander. State initialization for recurrent neural network modeling of time-series data. In

[17] H. Noguchi, T. Yamada, T. Mori, and T. Sato. Mobile robot path planning using human prediction model based on massive trajectories. In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pages 1–7. IEEE, 2012. 3

[18] D. Nuss, S. Reuter, M. Thom, T. Yuan, G. Krehl, M. Maile, A. Gern, and K. Dietmayer. A random finite set approach for dynamic occupancy grid maps with real-time application. *The International Journal of Robotics Research*, 37(8):841–866, 2018. 3

[19] T. Ohki, K. Nagatani, and K. Yoshida. Collision avoidance method for mobile robot considering motion and personal spaces of evacuees. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1819–1824. IEEE, 2010. 3

[20] P. Ondruska and I. Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. *arXiv preprint arXiv:1602.00991*, 2016. 3

[21] A. Petrovskaya and S. Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009. 1

[22] A. M. Santana, K. R. Aires, R. M. Veras, and A. A. Medeiros. An approach for 2d visual occupancy grid map using monocular vision. *Electronic Notes in Theoretical Computer Science*, 281:175–191, 2011. 1

[23] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015. 2

[24] E. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou. A review of global path planning methods for occupancy grid maps regardless of obstacle density. *Journal of Intelligent & Robotic Systems*, 84(1-4):829–858, 2016. 1

[25] S. Wang, R. Clark, H. Wen, and N. Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542, 2018. 2

[26] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 5

[27] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015. 3