2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Robust Modeling and Prediction in Dynamic Environments Using Recurrent Flow Networks

Sungjoon Choi, Kyungjae Lee, and Songhwai Oh

Abstract— **To enable safe motion planning in a dynamic environment, it is vital to anticipate and predict object movements. In practice, however, an accurate object identification among multiple moving objects is extremely challenging, making it infeasible to accurately track and predict individual objects. Furthermore, even for a single object, its appearance can vary significantly due to external effects, such as occlusions, varying perspectives, or illumination changes. In this paper, we propose a novel recurrent network architecture called a *recurrent flow network* that can infer the velocity of each cell and the probability of future occupancy from a sequence of occupancy grids which we refer to as an *occupancy flow*. The parameters of the recurrent flow network are optimized using Bayesian optimization. The proposed method outperforms three baseline optical flow methods, Lucas-Kanade, Lucas-Kanade with Tikhonov regularization, and HornSchunck methods, and a Bayesian occupancy grid filter in terms of both prediction accuracy and robustness to noise.**

I. Introduction

With the help of advances in computing and sensing technologies, robot applications have spread out from static and structured environments to dynamic real-world environments where autonomous driverless car and mobile service robots are representative examples. In this regard, accurate and robust future prediction in a dynamic environment should be properly addressed. Inevitable uncertainty in measurements as well as limited computation capability must be handled appropriately in order for dynamic prediction algorithms to be used in real-world applications. In this paper, we focus on the problem of predicting future occupancy information in a dynamic environment using a sequence of occupancy grids obtained from distance measuring sensors such as a laser rangefinder.

In static environments, occupancy grid mapping has been widely used for a number of mobile robot applications [1], [2]. An occupancy grid discretizes the environment into grids or cells and assigns the probability of occupancy to each cell [3] using distance measuring sensors such as laser rangefinders. Occupancy grid mapping has been successfully applied to a number of problems, including simultaneous localization and mapping (SLAM) [4].

This research was supported by a grant to Bio-Mimetic Robot Research Center funded by Defense Acquisition Program Administration and by Agency for Defense Development (UD130070ID) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2015R1A2A1A15052493).

S. Choi, K. Lee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 151-744, Korea (e-mail: {sungjoon.choi, kyungjae.lee, songhwai.oh}@cpslab.snu.ac.kr).

Some works have been focused on extending traditional occupancy grids framework to infer dynamic and temporal information such as a velocity of each cell. [5] presented a probabilistic grid-based mapping method for modeling dynamic environments by applying hidden Markov models. A Bayesian occupancy filter (BOF) was first proposed in [6] by adapting a 4-dimensional grid representation with 2-dimensional occupancy information and 2-dimensional velocity information in a Cartesian coordinate with Bayesian filtering techniques. The Bayesian occupancy grid filter (BOGF) [7] was presented for more accurate prediction of cell transitions by incorporating a prior knowledge. However, as both BOF and BOGF require $O(n^2)$ computational complexities, where n is the number of cells, this heavy computational complexities make such methods inappropriate for mobile robot applications.

In this paper, we propose a novel multilayer recurrent architecture, which we will refer to as a *recurrent flow network* and a unique update rule for inferring the velocity of each cell, an *occupancy flow*, and predicting future occupancy information. While the recurrent connections in the intermediate layer makes the proposed architecture a recurrent network, the network connections as well as an occupancy flow update rule vary greatly from existing recurrent neural networks. The connection weights as well as update rules of the recurrent flow network are governed by several parameters where these parameters are trained by Bayesian optimization [8].

The remainder of the paper is organized as follows. In Section II, the proposed recurrent flow network is introduced. A novel occupancy flow update rule for the recurrent flow network is presented in Section III. A comprehensive set of simulations and experiments using real-world LiDAR data is illustrated in Section IV.

II. Recurrent Flow Network

In this section, we propose a novel recurrent network architecture, which we will refer to as a recurrent flow network (RFN). The RFN is particularly designed to predict the future occupancy in a dynamic environment by inferring the velocity of each cell using the occupancy flow update method described in Section III. An egocentric occupancy grid is given as an input to the RFN and the intermediate context and the uppermost output layers estimate the velocity and the probability of future occupancy of each cell, respectively.

The proposed RFN is a two-level architecture where a single level is depicted in Figure 1(a). Each level consists of three layers: an input layer, context layer, and output

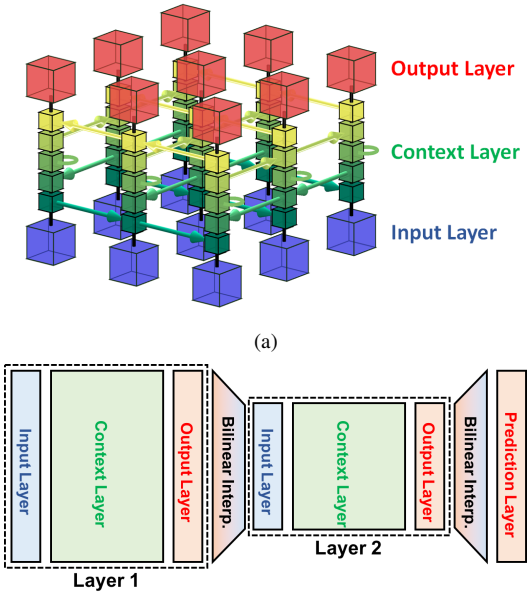


Fig. 1: (a) The structure of a simple single level RFN. The bottommost blue and uppermost red layers indicate input and output layers, respectively. The intermediate layer with smaller squares is a context layer with arrows indicating directed connections. (b) A two-level RFN with noisy occupancy grids are given as an input.

layer. The context layer consists of context columns and each context column connects one input cell to one output cell. Each context column consists of multiple context cells and these context cells have recurrent connections to its neighboring context cells including itself.

Figure 1(b) depicts the overall two-level structure of the RFN. We would like to note that while each layer shares the same structure, different optimized parameters of each level make the first layer to work as a denoising layer and the second layer to work as a temporal prediction layer.

A. Input Layer

Each cell in the input layer is connected to a cell in an occupancy grid map. The number of cells in the input layer is denoted by n. For notational simplicity $\mathbb{I}$ will be used as a set containing all n cell indices:

$$\mathbb{I} = \{1, 2, \ldots, n\}.$$

The possible states for an input cell are *occ*, *free*, and *unkn*, which refer to occupied, free or not occupied, and unknown, respectively. The i-th input cell at time t will be denoted by

$$o_t^i \in \{occ, free, unkn\}.$$

We also introduce a helper function $pos(i)$ that maps an index $i \in N$ to the position of o^i in the two dimensional grid world:

$$pos : \mathbb{I} \to \mathbb{Z} \times \mathbb{Z}.$$

B. Context Layer

The context layer is an intermediate recurrent layer between the input layer and the output layer, which plays a crucial role in inferring the *occupancy flow* The context layer consists of n context columns. Each context column consists of m context cells and has one-to-one correspondence with each input cell. Similar to $\mathbb{I}$, $\mathbb{M}$ will be used as a set containing all m context cell indices:

$$\mathbb{M} = \{1, 2, \ldots, m\}.$$

Every context cell in a context column has one directed connection to another context cell of a different context column or itself and context cells of the same level shares the same direction of connections. In other words, *each layer encodes a different velocity*. As m context cells exist within a context column, a context column is connected to itself and $m-1$ neighboring context columns. We will refer to these m connected context columns as a neighborhood. For example, for a simple single layer RFN shown in Figure 1(a), m is five, representing five neighbor cells: four cells in every four directions (velocities) and one cell itself.

Two helper functions are introduced for specifying neighbor connections: $nei(\cdot,\cdot)$ and $w(\cdot,\cdot)$ where $nei(i,j)$ returns the index of the connected context column of the j-th context cell of the i-th context column:

$$nei : \mathbb{I} \times \mathbb{M} \to \mathbb{I}$$

and $w(i,j)$ returns the weight of outward connection of the j-th context cell of the i-th context column

$$w : \mathbb{I} \times \mathbb{M} \to [0, 1].$$

The size of the neighborhood and connection weights encapsulate our assumption about how much the dynamic environment can vary within the sampling period and how likely objects will move, respectively. Weights of connections are governed by a single parameter ρ_n, where

$$w(i,j) = \exp\left(-\frac{\|pos(i) - pos(nei(i,j))\|_2^2}{\rho_n^2}\right).$$

For compact representations, we will denote the i-th context column, including its m context cells, at time t as

$$c_t^i \in \mathbb{R}_{\geq 0}^m$$

and all context columns at time t is referred as $c_t \in \mathbb{R}_{\geq 0}^{n \times m}$. The j-th context cell in c_t^i will be denoted as $c_t^{(i,j)}$.

C. Output Layer

The uppermost layer is the output layer, which consists of n output cells representing a predicted occupancy grid map. Each cell of the output layer has one-to-one correspondence with each context column and has a positive value between 0 and 1, representing the probability of corresponding cell being occupied in the next time step. We will denote the output layer at time t as

$$p_t \in [0, 1]^n.$$

We will also denote the i-th output cell at time t as p_t^i.

III. Occupancy Flow Update Rule

In this section, we propose the occupancy flow update rule for the recurrent flow network (RFN), which consists of three steps: *correction*, *context propagation*, and *prediction*. In the *correction* stage, we intensify correctly predicted context cells whereas attenuate incorrectly predicted ones. The updated information in the context layer is propagated to its neighborhood in the *context propagation* stage and a linear motion assumption is compensated using the concept of the *uncertainty in acceleration*. Finally, the occupancy prediction occurs in the *prediction* stage by passing the max-pooled context cells through a sigmoid function in each context column. Details are illustrated in forthcoming sections. Parameters used in the occupancy flow update rule are given in Table I.

M_n	Size of the neighborhood
ρ_n	Parameter for controlling connection weights
M_u	Size of the uncertainty smoothing
ρ_u	Parameter for controlling weights of uncertainty smoothing
α	Intensify rate for occupied context cells
β	Attenuate rate for free context cells
γ	Attenuate rate for unknown context cells
ϵ_{min}	Minimum threshold for context cells
ϵ_{max}	Maximum threshold for context cells
ϵ_{init}	Reinitialization value for context cells
θ_{pred}	Prediction threshold
θ_{bin}	Binary threshold
ν	Sigmoid function parameter
μ	Resize ratio between levels

TABLE I: Parameters of the recurrent flow network.

A. Update Rule

1) Correction: The *correction* step in the occupancy flow algorithm updates $\{c_t^i | i \in \mathbb{I}\}$, the values of the context layer at time t, using current input measurements, o_t, and previous values of the context layer, c_{t-1}. Specifically, a context column of which the corresponding input cell is newly occupied, which we will refer to as an occupied column, gets strengthened with a factor of α (line 9). If the maximum value of the context column is less than ϵ_{min} the context cells are reinitialized with ϵ_{init} (line 7). The context cells of which the corresponding input cells are not occupied or unknown get attenuated by the factor of β and γ, respectively (line 12).

2) Context Propagation: Once the context values are updated from the *correction* step, we propagate the context cell information to its neighbors taking the account of the connection weights (line 17-21). This process is equivalent to predicting the future flow information by assuming that the objects in a dynamic environment will follow a constant velocity model. The updated context cell values are further used to predict the future occupancy of the environment in the *prediction* step.

In practice, however, there is no guarantee that this assumption will properly hold in real world. Hence, we adapted a concept of *uncertainty in acceleration* which is similar to [7]. Particularly, the actual implementation resembles with image smoothing where the value of each context cell is updated by a weighted sum of context cells within its smoothing field. Two helper functions, $nei_u(i,j,k)$ and $w_u(i,j,k)$, are introduced to specify the connections and weights of a smoothing field. Suppose M_u is the size of a smoothing field. Then $nei_u(i,j,k)$ returns the index of the k-th connected cell in the smoothing field from the jth context cell of ith context column and $w_u(i,j,k)$ returns the corresponding weight (line 22-26).

3) Prediction: The final occupancy prediction is performed by first finding the maximum value in each context column and passing the maximum value through an activations function, a sigmoid function $\sigma(t;\nu)$ (line 28-30). This routine resembles to the max-pooling operating in a convolutional neural network (CNN) in that each context cell can be regarded as a filtered response of its neighbor information.

An overall occupancy flow update method for a single level recurrent flow network is shown in Algorithm 1. The output layer of the first RFN level is downsized with bilinear interpolation with a factor of μ and fed into the input layer of the second level and the final prediction of the second RFN

Algorithm 1 Occupancy Flow Algorithm (Single level)

1: **Input:** Previous and current occupancy grid map o_{t-1} and o_t, parameters in Table I, and context layer c_t.
2: **Output:** Predicted occupancy grid map p_{t+1}.
3: // 1. Correction step
4: **for** $i \in \mathbb{N}$ **do**
5: **if** o_t^i is occ AND o_{t-1}^i is $free$ **then**
6: **if** $\max(c_t^i) \leq \epsilon_{min}$ **then**
7: $c_t^i \leftarrow \epsilon_{init} \cdot \mathbf{1}_m$
8: **else**
9: $c_t^i \leftarrow \alpha \cdot c_t^i$
10: **end if**
11: **else**
12: $c_t^i \leftarrow \beta^{\mathbf{I}(o_t^i = free)} \gamma^{\mathbf{I}(o_t^i = unkn)} c_t^i$
13: **end if**
14: **end for**
15: $c_t \leftarrow \min(c_t, \epsilon_{max})$
16: // 2. Context-propagation step
17: **for** $i \in \mathbb{N}$ **do**
18: **if** o_t^i is occ **then**
19: $c_{t+1}^{nei(i,:)} \leftarrow w(i,:) \odot c_t^i$
20: **end if**
21: **end for**
22: **for** $i \in \mathbb{N}$ **do**
23: **for** $j \in \mathbb{M}$ **do**
24: $c_{t+1}^{(i,j)} \leftarrow \sum_{k=1}^{M_u} w_u(i,j,k) c_{t+1}^{(nei_u(i,j,k),j)}$
25: **end for**
26: **end for**
27: // 3. Prediction step
28: **for** $i \in \mathbb{N}$ **do**
29: $p_{t+1}^i \leftarrow \sigma\left(\max(c_{t+1}^i)\right)$
30: **end for**

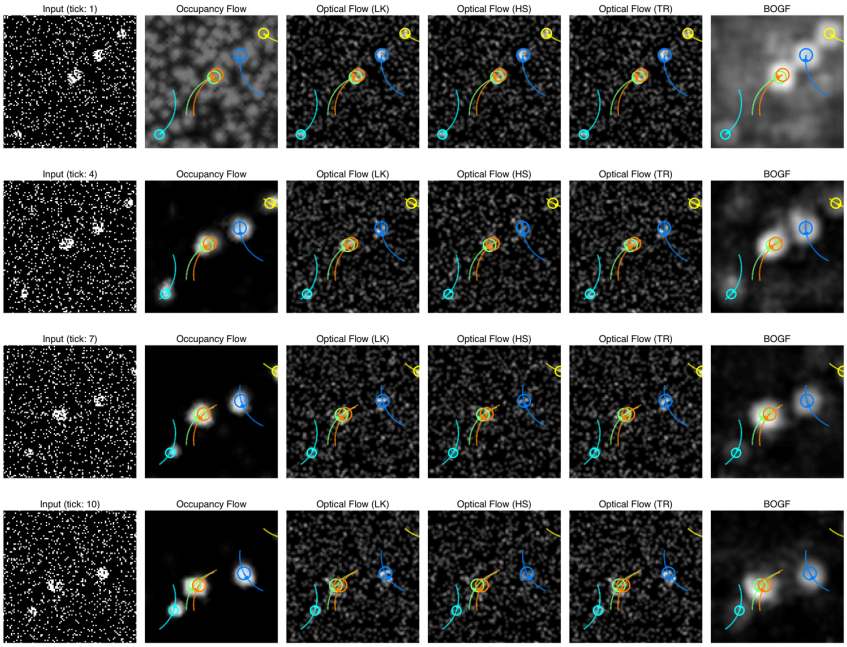


Fig. 2: A sequence of noisy occupancy grid maps and predicted occupancy grids with different methods. Each moving object and future trajectory are shown with a circle and dotted line with different colors, respectively.

	Occupancy Flow	LK	HS	TR	BOGF
Time [ms]	33.4	120.2	2.3	1.6	963.9

TABLE II: Running Times of tested algorithms.

layer is upsized with a factor of $1/\mu$ to match the input occupancy grids. Thus the total number of parameters of a two-level RFN is 27, 13 for parameters of each layers and 1 for the resize ratio μ.

B. Estimating Velocities

While the output cell of the RFN represents the probability of future occupancy, one can infer the velocity of each grid, the occupancy flow, using the corresponding context column. As each layer encodes different velocities, let us introduce a helper function $vel(j)$ that maps an index of a context cell to the corresponding two dimensional velocity:

$$vel : \mathbb{M} \rightarrow \mathbb{R} \times \mathbb{R}.$$

Then, the velocity of i-th cell at time t can be estimated by

$$v_t^i = \frac{\sum_{j=1}^{m} vel(j) \cdot c_t^{(i,j)}}{\sum_{j=1}^{m} c_t^{(i,j)}}.$$

C. Learning Parameters

Bayesian optimization [8] is used to train the parameters of the proposed recurrent flow network. We used the expected improvement (EI) criterion with the ARD kernel function with unit gain which is identical to the setting in [8]. Bayesian optimization is appropriate for optimizing the parameters of the recurrent flow network as some parameters are discrete, e.g., the size of the neighborhood M_n or uncertainty smoothing M_u.

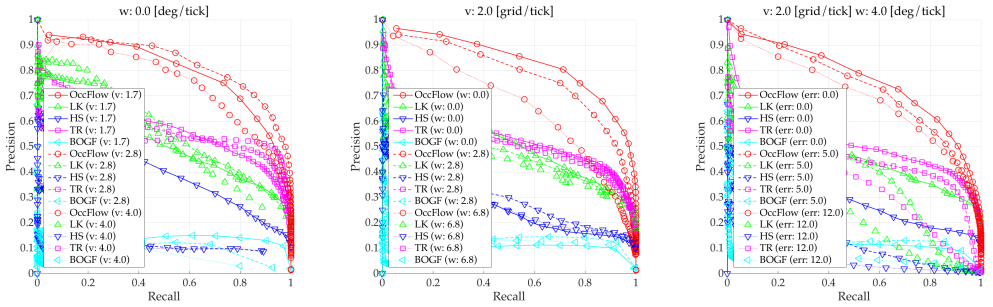
As shown in Section III-A, the total number of parameters is 27, and the score of a set of parameters is computed by the prediction performance. In particular, we simulated the future occupancy prediction experiments and the score is evaluated using the area under curve of the precision recall graph. Parameters are then fine-tuned via a greedy search. The final parameters of the recurrent flow network are shown in Table III and the same set of parameters are used for experiments in Section IV.

IV. Experiments

A. Future Occupancy Prediction

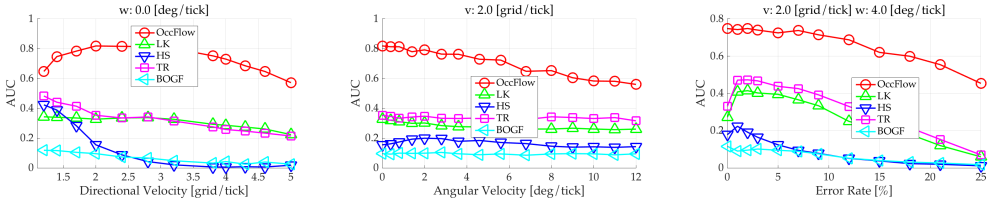
We tested prediction accuracies of the occupancy flow (OccFlow) inferred by the proposed recurrent flow network, Bayesian occupancy grid filter (BOGF) [7] and three different optical flow methods, Lucas-Kanade (LK) method [9], Lucas-Kanade with Tikhonov regularization (TR) [10], and HornSchunck method [11]. Bayesian occupancy filter (BOF) [6] was not compared as BOGF extends BOF by incorporating prior information [7].

The size of the occupancy grid map is restricted to 100×100 due to high computational and memory complexities of the BOGF method whose complexities are $O(n^2)$, where n is the number cells in the grid. In order to fairly compare these methods with ours, the occupancy flow, we carefully implemented an additional prediction stage for the BOGF and optical flow methods. The occupancy prediction is a straightforward process of going one step forward in time using the inferred velocity of the occupied cells. Image smoothing with a 3×3 unit Gaussian filter is applied as a post processing for all compared methods.



(a) (b) (c)

Fig. 3: Precision recall graphs of the prediction methods under different scenario, different directional velocities in (a), different angular velocities in (b), and different error rates in (c).



(a) (b) (c)

Fig. 4: AUC values of the prediction methods under different scenario, different directional velocities in (a), different angular velocities in (b), and different error rates in (c).

	M_n	ρ_n	M_u	ρ_u	α	β	γ	ϵ_{min}	ϵ_{max}	ϵ_{init}	θ_{pred}	θ_{bin}	ν	μ
$Level1$	3	4.23	3	1.12	1.53	0.05	0.85	0.81	14.6	2.89	0.81	0.81	1.42	0.5
$Level2$	5	1.72	3	0.8	5	0.3	0.79	0.23	27.8	1.73	0.79	0.66	0.15	2

TABLE III: Optimized parameters of the two-level recurrent flow network.

We tested the prediction performance under three different scenarios: different directional velocities (one to five grids per period), different angular velocities (0 to 12 degrees per period), and different salt and pepper error rates (0 to 40 percentages) on occupancy grids, to extensively evaluate the prediction performance of the recurrent flow network. In the different salt and pepper error cases, median filtering [12] is applied to optical flow methods and BOGF. The salt-pepper-noise emulates noises in rangefinder sensors from external environmental influences, such as rains or heavy sunlight. In each case, three to five obstacles are deployed at random locations with random speeds and 10 independent experiments are conducted to compute the average performance. The overall prediction performance of each method is measured by the area under curve (AUC) of the precision-recall graph.

Sequences of snapshots of different prediction methods are shown in Figure 2. One can clearly see that the prediction methods of the occupancy flow (second column) outperforms compared method even with a considerable amount of errors (15% salt-and-pepper noise). This strong robustness comes from to the unique *correction* and *context propagation* stages of the occupancy flow and it makes the occupancy flow appropriate for real-world applications which will be shown in the forthcoming sections.

Precision-recall graphs with different prediction methods and different noise levels are shown in Figure 3. The resulting areas under curves (AUC) of the precision recall graphs as a function of salt and pepper nose ratio are shown in Figure 4. In all scenarios, the proposed occupancy flow method outperforms the compared prediction methods. The performance gap between the proposed method and the others are maximized in the *different speeds* scenario. This is mainly due to the fact that the compared optical flow methods do not work properly with large displacements [13]. The BOGF, on the other hand, tends to over-predict the future occupancy, which is shown in the precision-recall graph in that it achieves high recall while showing poor precision.

B. Real World Driving LiDAR Dataset

We also applied the occupancy flow to the Ford campus vision LiDAR dataset [17] to show its applicability to advanced driver assistance systems (ADAS). For converting the 3 dimensional point cloud data (PCD) to the 2 dimensional occupancy grids, we first divide the planar space into 100×100 grids and compute the maximum height value and variance of the height information of the PCD per each cell in the grids. If the maximum height value or variance exceed predefined thresholds, we assign the cell to be occupied and

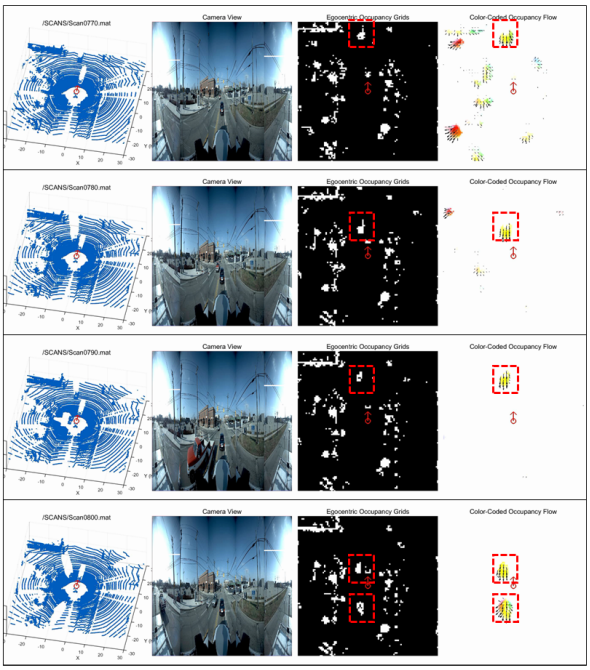


Fig. 5: A sequence of snapshots of real world driving LiDAR experiments. The first and second columns indicate 3 dimensional point cloud data and stitched RGB camera images, respectively. The converted egocentric occupancy grids are shown in the third column and the color-coded occupancy flow with arrows are shown in the forth column. Car positions in occupancy grids are shown with red dotted rectangles.

free otherwise where similar method had been used in [18], [19].

A sequence of snapshots is illustrated in Figure 5. In this sequence, a car with LiDAR sensors stops at the crossroads while two cars are approaching in order at the opposite lane. These phenomena are appropriately captured by the occupancy flow. At first, several *occupancy flows* are detected, mostly on downward directions, since the car was moving upward. As the car remains still, however, only the moving cars at the opposite lane are detected (see the second to forth rows at the last column) along with their directions shown with yellow color indicating downward direction. Note that even though the car stays idle, the occupancy grid map varies constantly due to external effects such as sunlights or internal sensor configurations.

V. Discussion and Conclusion

In this paper, we focused on the problem of predicting future occupancies by estimating the velocity of each cell using the recurrent flow network. Unlike existing occupancy filtering methods which require $O(n^2)$ space and time complexities, where n is the number of cells in a grids, the complexities of the proposed method are $O(nm)$, where m is the number of neighbors and $m \ll n$. The average runtime of the occupancy flow was 34 ms with 100×100 grids in MATLAB making it suitable for many real-time applications.

We first tested the occupancy prediction performance of the occupancy flow on synthetic examples and achieved a favorable prediction performance compared to a Bayesian occupancy grid filter and three widely used optical flow methods. Furthermore, the proposed method was tested using a Ford campus LiDAR dataset. In all cases, correct flow information was captured while ignoring noises in the input occupancy grids.

References

[1] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI magazine*, vol. 9, no. 2, p. 61, 1988.

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

[3] S. Thrun, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, pp. 1–35, 2002.

[4] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.

[5] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Proc. of the AAAI Conference on Artifical Intelligence (AAAI)*, July 2012.

[6] C. Chen, C. Tay, C. Laugier, and K. Mekhnacha, "Dynamic environment modeling with gridmap: a multiple-object tracking application," in *International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, December 2006.

[7] T. Gindele, S. Brechtel, J. Schröder, and R. Dillmann, "Bayesian occupancy grid filter for dynamic environments using prior map knowledge," in *Intelligent Vehicles Symposium*. IEEE, June 2009.

[8] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems (NIPS)*, December 2012.

[9] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision." in *Internaltion Joint Conference on Artificial Intelligence (IJCAI)*, August 1981.

[10] A. N. Tikhonov and V. A. Arsenin, *Solutions of ill-posed problems*. Vh Winston, 1977.

[11] B. K. Horn and B. G. Schunck, "Determining optical flow," in *1981 Technical symposium east*. International Society for Optics and Photonics, 1981, pp. 319–331.

[12] J. S. Lim, *Two-dimensional signal and image processing*. Prentice Hall, 1990, vol. 1.

[13] T. Brox, C. Bregler, and J. Malik, "Large displacement optical flow," in *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2009.

[14] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[15] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," in *Proc. of the International Conference of Robotics and Automation (ICRA)*. IEEE, May 1992.

[16] K. P. Valavanis, T. Hebert, R. Kolluru, and N. Tsourveloudis, "Mobile robot navigation in 2-d dynamic environments using an electrostatic potential field," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 30, no. 2, pp. 187–196, 2000.

[17] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and lidar data set," *International Journal of Robotics Research*, vol. 30, no. 13, pp. 1543–1552, November 2011.

[18] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[19] F. Von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche, "Driving with tentacles: Integral structures for sensing and motion," *Journal of Field Robotics*, vol. 25, no. 9, pp. 640–673, 2008.