Deep Semantic Classification for 3D LiDAR Data

Pre-print of article that will appear at the 2017 IEEE International Conference on Intelligent Robots and Systems.

Please cite this paper as:
A. Dewan, G. Oliveira and W. Burgard, ”Deep Semantic Classification for 3D LiDAR Data” in Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE, 2017.

bibtex:

```
@inproceedings{ dewan17iros,
author = { Ayush Dewan, Gabriel L. Oliveira and Wolfram Burgard},
booktitle = { Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on},
organization = {IEEE},
title = { {Deep Semantic Classification for 3D LiDAR Data} },
year = {2017}
}
```

Deep Semantic Classification for 3D LiDAR Data

Ayush Dewan Gabriel L. Oliveira Wolfram Burgard

***Abstract*— Robots are expected to operate autonomously in dynamic environments. Understanding the underlying dynamic characteristics of objects is a key enabler for achieving this goal. In this paper, we propose a method for pointwise semantic classification of 3D LiDAR data into three classes: non-movable, movable and dynamic. We concentrate on understanding these specific semantics because they characterize important information required for an autonomous system. Non-movable points in the scene belong to unchanging segments of the environment, whereas the remaining classes corresponds to the changing parts of the scene. The difference between the movable and dynamic class is their motion state. The dynamic points can be perceived as moving, whereas movable objects can move, but are perceived as static. To learn the distinction between movable and non-movable points in the environment, we introduce an approach based on deep neural network and for detecting the dynamic points, we estimate pointwise motion. We propose a Bayes filter framework for combining the learned semantic cues with the motion cues to infer the required semantic classification. In extensive experiments, we compare our approach with other methods on a standard benchmark dataset and report competitive results in comparison to the existing state-of-the-art. Furthermore, we show an improvement in the classification of points by combining the semantic cues retrieved from the neural network with the motion cues.**

I. INTRODUCTION

One of the vital goals in mobile robotics is to develop a system that is aware of the dynamics of the environment. If the environment changes over time, the system should be capable of handling these changes. In this paper, we present an approach for pointwise semantic classification of a 3D LiDAR scan into three classes: *non-movable, movable and dynamic*. Segments in the environment having non-zero motion are considered dynamic, a region which is expected to remain unchanged for long periods of time is considered non-movable, whereas the frequently changing segments of the environment is considered movable. Each of these classes entail important information. Classifying the points as dynamic facilitates robust path planning and obstacle avoidance, whereas the information about the non-movable and movable points can allow uninterrupted navigation for long periods of time.

To achieve the desired objective, we use a Convolutional Neural Network (CNN) [23, 12, 13] for understanding the distinction between movable and non-movable points. For our approach, we employ a particular type of CNNs called up-convolutional networks [20]. They are fully convolutional architectures capable of producing dense predictions for a high-resolution input. The input to our network is a set of three channel 2D images generated by unwrapping 360° 3D LiDAR data onto a spherical 2D plane and the output is the *objectness* score, where a high score corresponds to the movable class. Similarly, we estimate the *dynamicity* score for a point by first calculating pointwise 6D motion using our previous method [6] and then comparing the estimated motion with the odometry to calculate the score. We combine the two scores in a Bayes filter framework for improving the classification especially for dynamic points. Furthermore, our filter incorporates previous measurements, which makes the classification more robust. In Fig. 1 we show the classification results of our method. Black points represent non-movable points, whereas movable and dynamic points are shown in green and blue color respectively.

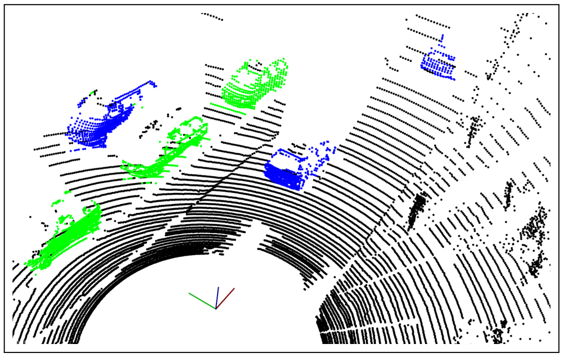

Fig. 1. Semantic classification of a 3D LiDAR scan. *Non-movable* points belonging to building, vegetation and road are colored black. Points on parked vehicles, correctly classified as *movable* are shown in green, whereas the points on moving vehicles are classified as *dynamic* and are shown in blue.

Other methods [22, 10] for similar semantic classification have been proposed for RGB images, however, a method solely relying on range data does not exist according to our best knowledge. For LiDAR data, separate methods exists for both object detection [4, 16, 9, 3] and for distinguishing between static and dynamic objects in the scene [5, 18, 21]. The two main differences between our method and the other object detection methods is that the output of our method is a pointwise *objectness* score, whereas other methods concentrate on calculating object proposals and predict a bounding box for the object. Since our objective is pointwise classification, the need for estimating a bounding box is alleviated as a pointwise score currently suffices. The second difference is that we utilize the complete 360° field of view (FOV) of LiDAR for training our network in contrast to other methods which only use the points that overlap with the FOV of the front camera.

The main contribution of our work is a method for seman-

All authors are with the Department of Computer Science at the University of Freiburg, Germany. This work has been supported by the European Commission under the grant numbers FP7-610532-SQUIRREL

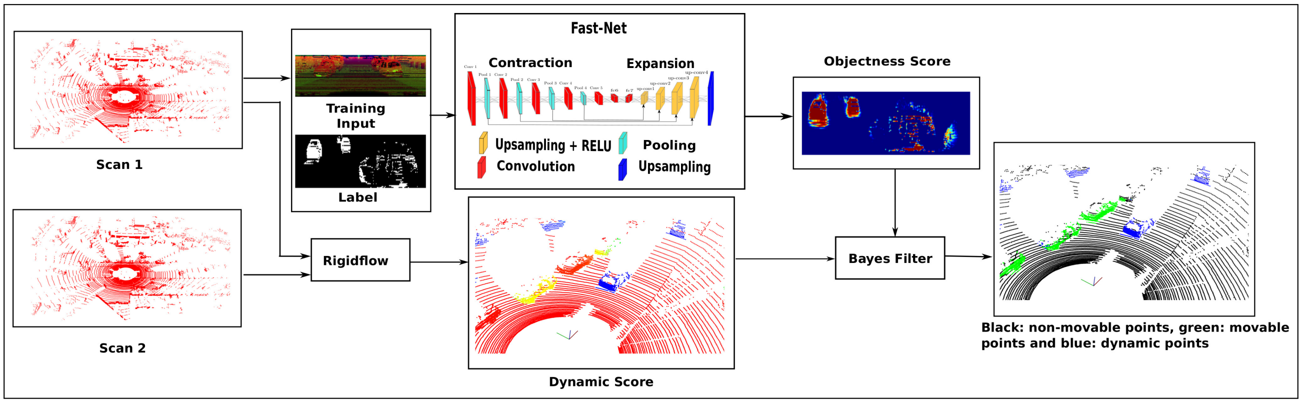

Fig. 2. An overview of the proposed method. The input is two LiDAR scans, where the first scan is converted to a 2D image and is processed by the neural network for estimating the *objectness* score. Our RigidFlow approach uses both scans and calculates the *dynamicity* score. The *objectness* and the *dynamicity* score is then processed by the Bayes filter approach for estimating the desired pointwise semantic classification of the scene.

tic classification of a LiDAR scan for learning the distinction between non-movable, movable and dynamic parts of the scene. As mentioned above, these three classes encapsulate information which is critical for robust autonomous robotic system. A method for learning the same classes in LiDAR scans has not been proposed before, even though different methods exists for learning other semantic level information. Unlike other existing methods, we use the complete range of the LiDAR data. For training the neural network we use the KITTI object benchmark [11] and compare our results on this benchmark with the other methods. We also test our approach on the dataset by Moosmann and Stiller [18].

II. Related Works

We first discuss methods which have been proposed for similar classification objectives. Then, we discuss other methods proposed especially for classification in LiDAR scans and briefly discuss the RGB image based methods.

For semantic motion segmentation in images, Reddy *et al.* [22] proposed a dense CRF based method, where they combine semantic, geometric and motion constraints for joint pixel-wise semantic and motion labeling. Similar to them Fan *et al.* [10] proposed a neural network based method. Their method is closest to our approach as they also combine motion cues with the object information. For retrieving the object level semantics, they use a deep neural network [1]. Both of these methods show results on the KITTI sceneflow benchmark for which ground truth is only provided for the images, thus making a direct comparison difficult. However, we compare the performance of our neural network with the network used by Fan *et al.* [10].

For LiDAR data, a method with similar classification objectives does not exist, however different methods for semantic segmentation [27, 7, 25], object detection [4, 16, 9] and moving object segmentation [5, 18, 21] have been proposed. Targeting semantic segmentation in 3D LiDAR data, Wang *et al.* proposed a method [25] for segmenting movable objects. More recently Zelener and Stamos proposed a method [27] for object segmentation, primarily concentrating on objects with missing points and Dohan *et al.* [7] discusses a method for hierarchical semantic segmentation of a LiDAR scan. These methods report results on different datasets and since we use the KITTI object benchmark for our approach we restrict our comparison to other recent methods that use the same benchmark.

For object detection, Engelcke *et al.* [9] extends their previous work [24] and propose a CNN based method for detecting objects in 3D LiDAR data. Li *et al.* proposed a Fully Convolutional Network based method [16] for detecting objects, where they use two channel (depth + height) 2D images for training the network and estimate 3D object proposals. The most recent approach for detecting objects in LiDAR scans is proposed by Chen *et al.* [4]. Their method leverages over both multiple view point information (front camera view + bird eye view) and multiple modalities (LiDAR + RGB). They use a region based proposal network for fusing different sources of information and also estimate 3D object proposals. For RGB images, approaches by Chen *et al.* [3, 2] are the two recent methods for object detection. In All of these methods, the neural network is trained for estimating bounding boxes for object detection, whereas, our network is trained for estimating pointwise *objectness* score; the information necessary for pointwise classification. In the results section, we discuss these differences in detail and present comparative results.

Methods proposed for dynamic object detection include our previous work [5] and other methods [18, 21, 26]. Our previous method and [26] are model free methods for detection and tracking in 3D and 2D LiDAR scans respectively. For detecting dynamic points in a scene, Pomerleau *et al.* [21] proposed a method that relies on a visibility assumption, i.e., the scene behind the object is observed, if an object moves. To leverage over this information, they compare an incoming scan with a global map and detect dynamic points. For tracking and mapping of moving objects a method was proposed by Moosmann and Stiller [18]. The main difference between these methods and our approach is that we perform pointwise classification and these methods reason at object level.

III. Approach

In this paper, we propose a method for pointwise semantic classification of a 3D LiDAR scan. The points are classified into three classes: non-movable, movable and dynamic. In Fig. 2 we illustrate a detailed overview of our approach. The input to our approach consists of two consecutive 3D LiDAR scans. The first scan is converted into a three-channel 2D image, where the first channel holds the range values and the second and third channel holds the intensity and height values respectively. The image is processed by an up-convolutional network called Fast-Net [20]. The output of the network is the pointwise *objectness* score. Since, in our approach points on an object are considered movable, the term object is used synonymously for movable class. For calculating the *dynamicity* score, our approach requires two consecutive scans. As a first step, we estimate pointwise motion using our RigidFlow [6] approach. The estimated motion is then compared with the odometry to calculate the *dynamicity* score. These scores are provided to the Bayes filter framework for estimating the pointwise semantic classification.

A. Object Classification

Up-convolutional networks are becoming the foremost choice of architectures for semantic segmentation tasks based on their recent success [1, 17, 20]. These methods are capable of processing images of arbitrary size, are computationally efficient and provide the capability of end-to-end training. Up-convolutional networks have two main parts: contractive and expansive. The contractive part is a classification architecture, for example AlexNet [15] or VGG [23]. They are capable of producing a dense prediction for a high-resolution input. However, for a low-resolution output of the contractive part, the segmentation mask is not capable of providing the descriptiveness necessary for majority of semantic segmentation tasks. The expansive part subdues this limitation by producing an input size output through the multi-stage refinement process. Each refinement stage consists of an upsampling and a convolution operation of a low-resolution input, followed by the fusion of the up-convolved filters with the previous pooling layers. The motivation of this operation is to increase the finer details of the segmentation mask at each refinement stage by including the local information from pooling.

In our approach we use the architecture called Fast-Net [20] (see Fig. 2). It is an up-convolutional network designed for providing near real-time performance. More technical details and a detailed explanation of the architecture is described in [20].

B. Training Input

For training our network we use the KITTI object benchmark. The network is trained for classifying points on *cars* as movable. The input to our network are three channel 2D images and the corresponding ground truth labels. The 2D images are generated by projecting the 3D data onto a 2D point map. The resolution of the image is 64×870. Each channel in an image represents a different modality. First channel holds the range values, second channel holds the intensity values, corresponding to the surface reflectance and the third channel holds the height values for providing geometric information. The KITTI benchmark provides ground truth bounding boxes for the objects in front of the camera, even though the LiDAR scanner has 360° FOV. To utilize the complete LiDAR information we use our tracking approach [5] for labeling the objects that are behind the camera by propagating the bounding boxes from front of the camera.

1) Training: Our approach is modeled as a binary segmentation problem and the goal is to predict the *objectness* score required for distinguishing between movable and non-movable points. We define a set of training images $\mathcal{T} = (X_n, Y_n), n = 1, \ldots, N$, where $X_n = \{x_k, k = 1, \ldots, |X_n|\}$ is a set of pixels in an example input image and $Y_n = \{y_k, k = 1, \ldots, |Y_n|\}$ is the corresponding ground truth, where $y_k = \{0,1\}$. The activation function of our model is defined as $f(x_k, \theta)$, where θ is our network model parameters. The network learns the features by minimizing the cross-entropy(*softmax*) loss in Eq. (1) and the final weights θ^* are estimated by minimizing the loss over all the pixels as shown in Eq. (2).

$$\mathcal{L}(p,q) = -\sum_{c\in\{0,1\}} p_c \log q_c \tag{1}$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{k=1}^{N\times|X_n|} \mathcal{L}\left(f\left(x_k, \theta\right), y_k\right) \tag{2}$$

We perform a multi-stage training, by using one single refinement at a time. Such technique is used based on the complexity of single stage training and on the gradient propagation problems of training deeper architectures. The process consists of initializing the contractive side with the VGG weights. After that the multi-stage training begins and each refinement is trained until we reach the final stage that uses the first pooling layer.

We use Stochastic Gradient Descent with momentum as the optimizer, a mini batch of size one and a fixed learning rate of $1e^{-6}$. Based on the mini batch size we set the momentum to 0.99, allowing us to use previous gradients as much as possible. Since the labels in our problem are unbalanced because the majority of the points belong to the non-movable class, we incorporate *class balancing* as explained by Eigen and Fergus [8].

The output of the network is a pixel wise score a_c^k for each class c. The required *objectness* score $\xi^k \in [0,1]$ for a point k is the posterior class probability for the *movable* class.

$$\xi^k = \frac{\exp(a_1^k)}{\exp(a_1^k) + \exp(a_0^k)} \tag{3}$$

C. RigidFlow

In our previous work [6], we proposed a method for estimating pointwise motion in LiDAR scans. The input to our method are two consecutive scans and the output is the complete 6D motion for every point in the scan. The two main advantages of this method is that it allows estimation of different arbitrary motions in the scene, which is of critical importance when there are multiple dynamic objects in the scene and secondly it works for both rigid and non-rigid bodies.

We represent the problem using a factor graph $G = (\Phi, \mathscr{T}, \mathscr{E})$ with two node types: factor nodes $\phi \in \Phi$ and state variables nodes $\tau_k \in \mathscr{T}$. Here, $\mathscr{E}$ is the set of edges connecting Φ and state variable nodes $\mathscr{T}$.

The factor graph describes the factorization of the function

$$\phi(\mathscr{T}) = \prod_{i \in I_d} \phi_d(\tau_i) \prod_{l \in N_p} \phi_p(\tau_i, \tau_j), \tag{4}$$

where $\mathscr{T}$ is the following rigid motion field:

$$\mathscr{T} = \{\tau_k \mid \tau_k \in SE(3), k = 1, \ldots, K\} \tag{5}$$

$\{\phi_d, \phi_p\} \in \phi$ are two types of factor nodes describing the energy potentials for the data term and regularization term respectively. The term I_d is the set indices corresponding to keypoints in the first frame and $N_p = \{\langle 1,2 \rangle, \langle 2,3 \rangle, \ldots, \langle i,j \rangle\}$ is the set containing indices of neighboring vertices. The data term, defined only for keypoints is used for estimating motion, whereas the regularization term asserts that the problem is well posed and spreads the estimated motion to the neighboring points. The output of our method is a dense rigid motion field $\mathscr{T}^*$, the solution of the following energy minimization problem:

$$\mathscr{T}^* = \arg\min_{\mathscr{T}} E(\mathscr{T}), \tag{6}$$

where the energy function is:

$$E(\mathscr{T}) = -\ln \phi(\mathscr{T}) \tag{7}$$

A more detailed explanation of the method is presented by Dewan *et al.* [6].

D. Bayes Filter for Semantic Classification

The rigid flow approach estimates pointwise motion, however it does not provide the semantic level information. To this end, we propose a Bayes filter method for combining the learned semantic cues from the neural network with the motion cues for classifying a point as non-movable, movable and dynamic. The input to our filter is the estimated 6D motion, odometry and the *objectness* score from the neural network. The *dynamicity* score is calculated within the framework by comparing the motion with the odometry.

The *objectness* score from the neural network is sufficient for classifying points as movable and non-movable, however, we still include this information in filter framework for the following two reasons:

- Adding object level information improves the results for dynamic classification because a point belonging to a non-movable object has infinitesimal chance of being dynamic, in comparison to a movable object.
- The current neural network architecture does not account for the sequential nature of the data. Therefore, having a filter over the classification from the network, allows filtering of wrong classification results by using the information from the previous frames. The same holds for classification of dynamic points as well.

For every point $P_t^k \in \mathbb{R}^3$ in the scan, we define a state variable $x_t = \{\texttt{dynamic, movable, non-movable}\}$. The objective is to estimate the belief of the current state for a point P_t^k.

$$Bel(x_t^k) = p(x_t^k \mid x_{1:t-1}^k, \tau_{1:t}^k, \xi_{1:t}^k, o_t^k) \tag{8}$$

The current belief depends on the previous states $x_{1:t-1}^k$, motion measurements $\tau_{1:t}^k$, object measurements $\xi_{1:t}^k$ and a Bernoulli distributed random variable o_t^k. This variable models the object information, where $o_t^k = 1$ means that a point belongs to an object and therefore it is movable. For the next set of equations we skip the superscript k that represents the index of a point.

$$\begin{aligned} Bel(x_t) &= p(x_t \mid x_{1:t-1}, \tau_{1:t}, o_t, \xi_{1:t}) && (9) \\ &= \eta\, p(\tau_t, o_t \mid x_t, \xi_{1:t}) \int p(x_t \mid x_{t-1}) Bel(x_{t-1}) dx_{t-1} && (10) \\ &= \eta\, p(\tau_t \mid x_t) p(o_t \mid x_t, \xi_{1:t}) \int p(x_t \mid x_{t-1}) Bel(x_{t-1}) dx_{t-1} && (11) \end{aligned}$$

In Eq. (10) we show the simplification of the Eq. (8) using the Bayes rule and the Markov assumption. The likelihood for the motion measurement is defined in Eq. (12).

$$p(\tau_t \mid x_t) = \mathscr{N}(\tau_t; \hat{\tau}_t, \Sigma) \tag{12}$$

It compares the expected measurement $\hat{\tau}_t$ with the observed motion. In our case the expected motion is the odometry measurement. The output of the likelihood function is the required *dynamicity* score.

In Eq. (11) we assume the independence between the estimated motion and the object information. To calculate the object likelihood we first update the value of the random variable o_t by combining the current *objectness* score ξ_t with the previous measurements in a log-odds formulation (Eq. (13)).

$$l(o_t \mid \xi_{1:t}) = l(\xi_t) + l(o_t \mid \xi_{1:t-1}) - l(o_0) \tag{13}$$

The first term on the right side incorporates the current measurement, the second term is the recursive term which depends on the previous measurements and the last term is the initial prior. In our experiments, we set $o_0 = 0.2$ because we assume that the scene predominately contains non-moving objects.

$$p(o_t \mid x_t, \xi_{1:t}) = \begin{cases} p(\neg o_t \mid \xi_{1:t}) & \text{if } x_t = \texttt{non-movable} \\ p(o_t \mid \xi_{1:t}) & \text{if } x_t = \texttt{movable} \\ s \cdot p(o_t \mid \xi_{1:t}) & \text{if } x_t = \texttt{dynamic} \end{cases} \tag{14}$$

The object likelihood model is shown in Eq. (14). As the neural network is trained to predict the non-movable and movable class, the first two cases in Eq. (14) are straightforward. For the case of dynamic object, we scale the prediction of movable class by a factor $s \in [0, 1]$ since all the dynamic objects are movable, however, not all movable object are dynamic. This scaling factor approximates the ratio of number of dynamic objects in the scene to the number of movable objects. This ratio is environment dependent for instance on a *highway*, value of s will be close to 1, since most of movable objects will be dynamic. For our experiments, through empirical evaluation, we chose the value of $s = 0.6$.

IV. Results

To evaluate our approach we use the dataset from the KITTI object benchmark and the dataset provided by Moosmann and Stiller [18]. The first dataset provides object annotations but does not provide the labels for moving objects and for the second dataset we have the annotations for moving objects [5]. Therefore to analyze the classification of movable and non-movable points we use the KITTI object benchmark and use the second dataset for examining the classification of dynamic points. For all the experiments, Precision and Recall are calculated by varying the confidence measure of the prediction. For object classification the confidence measure is the *objectness* score and for dynamic classification the confidence measure is the output of the Bayes filter approach. The reported F1-score [19] is always the maximum F1-score for the estimated Precision Recall curves and the reported precision and recall corresponds to the maximum F1-score.

A. Object Classification

Our classification method is trained to classify points on *cars* as movable. The KITTI object benchmark provides 7481 annotated scans. Out of these scans we chose 1985 scans and created a dataset of 3789 scans by tracking the labeled objects. The implementation of Fast-Net is based on a deep learning toolbox Caffe [14]. The network was trained and tested on a system containing an NVIDIA Titan X GPU. For testing, we use the same validation set as mentioned by Chen *et al.* [4].

We provide quantitative analysis of our method for both pointwise prediction and object-wise prediction. For object-wise prediction we compare with these methods [3, 2, 16, 4]. Output for all of these methods is bounding boxes for the detected objects. A direct comparison with these methods is difficult since output of our method is pointwise prediction, however, we still make an attempt by creating bounding boxes out of our pointwise prediction as a post-processing step. We project the predictions from 2D image space to a 3D point cloud and then estimate 3D bounding boxes by clustering points belonging the to same surface as one object. The clustering process is described in our previous method [6].

For object-wise precision, we follow the KITTI benchmark guidelines and report average precision for easy, moderate and hard cases. The level of difficulty depends on the height of the ground truth bounding box, occlusion level and the truncation level. We compare the average precision for 3D bounding boxes AP_{3D} and the computational time with the other methods in Tab. I. The first two methods are based on RGB image, third method is solely LiDAR based, and the last method combines multiple view points of LiDAR data with RGB data. Our method outperforms the first three methods and an instance of the last method (front view) in terms of AP_{3D}. The computational time for our method includes the pointwise prediction on a GPU and object-wise prediction on CPU. The time reported for all the methods in Tab. I is the processing time on GPU. The CPU processing time for object-wise prediction of our method is $0.30s$. Even though performance of our method is not comparable with the two cases where LiDAR front view (FV) data is combined with bird eye view (BV) and RGB data, the computational time for our method is nearly 10× faster.

TABLE I

Object Classification AP_{3D}

Method	Data	IoU=0.5			time
		Easy	Moderate	Hard	
Mono3D [3]	Mono	25.19	18.2	15.52	4.2s
3DOP [2]	Stereo	46.04	34.63	30.09	3s
VeloFCN [16]	LiDAR	67.92	57.57	52.56	1s
MV3D [4]	LiDAR (FV)	74.02	62.18	57.61	-
MV3D [4]	LiDAR (FV+BV)	95.19	87.65	80.11	0.3s
MV3D [4]	LiDAR (FV+BV+Mono)	**96.02**	**89.05**	**88.38**	0.7s
Ours	LiDAR	95.31	71.87	70.01	**0.07s**

TABLE II

Pointwise versus Object-wise Prediction

Method	Recall	Recall (easy)	Recall (moderate)	Recall (hard)
pointwise	**81.29**	**84.79**	**71.07**	**68.10**
object-wise	60.47	87.91	48.44	46.80

TABLE III

Object Classification F1-score

Method	F1 Score	Precision	Recall
Ours(Fast-Net)	**80.16**	**79.06**	**81.29**
Seg-Net	69.83	85.27	59.12
Without *Class Balancing*	78.14	76.73	79.60

In Tab. II we report the pointwise and object-wise recall for the complete test data and for the three difficulty levels. The object level recall correspond to the AP_{3D} results in Tab. I. The reported pointwise recall is the actual evaluation of our method. The decrease in recall from pointwise prediction to object-wise is predominantly for moderate and hard case because objects belonging to these difficulty levels are often far and occluded therefore discarded during object clustering. The removal of small clusters is necessary because minimal over segmentation in image space potentially results in multiple bounding boxes in 3D space as neighboring pixels in 2D projected image can have large difference in depth, this is especially true for pixels on the boundary of an object. The decrease in performance from pointwise to object-wise prediction should not be seem as a drawback of our approach since our main focus is to estimate precise and robust pointwise

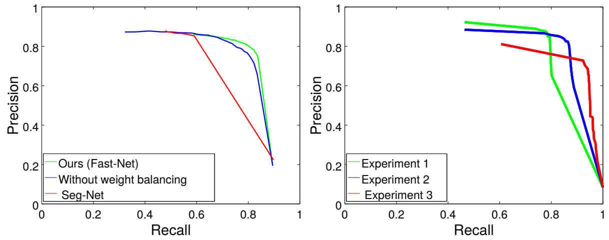

Fig. 3. Left: Precision Recall curves for object classification. Right: Precision recall curves for dynamic classification. Experiment 1 involves using the proposed Bayes filter approach, in experiment 2, the object information is not updated recursively and for the experiment 3, only motion cues are used for classification.

prediction required for the semantic classification.

We show the Precision Recall curves for pointwise object classification in Fig. 3 (right). Our method outperforms Seg-Net and we report an increase in F1-score by 12% (see Tab. III). This network architecture was used by Fan *et al.* [10] in their approach. To highlight the significance of *class balancing*, we trained a neural network without *class balancing*. Inclusion of this information increases the recall predominantly at high confidence values (see Fig. 3).

B. Semantic Classification

For the evaluation of semantic classification we use a publicly available dataset [18]. In our previous work [5] we annotated the dataset for evaluating moving object detection. The dataset consists of two sequences: Scenario-A and Scenario-B, of 380 and 500 frames of 3D LiDAR scans respectively.

TABLE IV

DYNAMIC CLASSIFICATION F1-SCORE

Method	Scenario A			Scenario B		
	F1-score	Precision	Recall	F1-score	Precision	Recall
Experiment 1	82.43	87.48	77.9	**72.24**	**76.29**	68.60
Experiment 2	**84.44**	**83.85**	84.9	71.48	75.83	67.66
Experiment 3	81.40	72.77	**92.34**	63.89	59.46	**69.62**

We report the results for the dynamic classification for three different experiments. For first experiment we use the approach discussed in Sec. III-D. In the second experiment, we skip the step of updating the object information (see Eq. (13)) and only use the current *objectness* score within the filter framework. For the final experiment, object information is not included in the filter framework and the classification of dynamic points rely solely on motion cues.

We show the Precision Recall curves for classification of dynamic points for all the three experiments for Scenario-A in Fig. 3 (right). The PR curves illustrates that the object information affects the sensitivity (recall) of the dynamic classification, for instance when the classification is based only on motion cues (red curve), recall is better among all the three cases. With the increase in object information sensitivity decreases, thereby causing a decrease in recall. In Tab. IV we report the F1-score for all the three experiments on both the datasets. For both the scenarios, F1-score increases after adding the object information which shows the significance of leveraging the object cues in our framework. In Fig. 5, we show a visual illustration for this case.

For the Scenario-A, the highest score is for the second experiment. However, we would like to emphasize that the affect of including the predictions from the neural network in the filter is not only restricted to classification of dynamic points. In Fig. 6, we show the impact of our proposed filter framework on the classification of movable points.

V. CONCLUSION

In this paper, we present an approach for pointwise semantic classification of a 3D LiDAR scan. Our approach uses an up-convolutional neural network for understanding the difference between movable and non-movable points and estimates pointwise motion for inferring the dynamics of the scene. In our proposed Bayes filter framework, we combine the information retrieved from the neural network with the motion cues to estimate the required pointwise semantic classification. We analyze our approach on a standard benchmark and report competitive results in terms for both, average precision and the computational time. Furthermore, through our Bayes filter framework we show the benefits of combining learned semantic information with the motion cues for robust and precise classification. For both the datasets we achieve a better F1-score. We also show that introducing the object cues in the filter improves the classification of movable points.

REFERENCES

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv: 1511.00561*, 2015. URL `http://arxiv.org/abs/1511.00561`.

[2] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015.

[3] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. *arXiv preprint arXiv:1611.07759*, 2016.

[5] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Motion-based detection and tracking in 3d lidar scans. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[6] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Rigid scene flow for 3d lidar scans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[7] David Dohan, Brian Matejek, and Thomas Funkhouser. Learning hierarchical semantic segmentations of lidar data. In *3D Vision (3DV), 2015 International Conference on*, pages 273–281. IEEE, 2015.

[8] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[9] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. *arXiv preprint arXiv:1609.06666*, 2016.

[10] Qiu Fan, Yang Yi, Li Hao, Fu Mengyin, and Wang Shunting. Semantic motion segmentation for urban dynamic scene understanding. In *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, pages 497–502. IEEE, 2016.

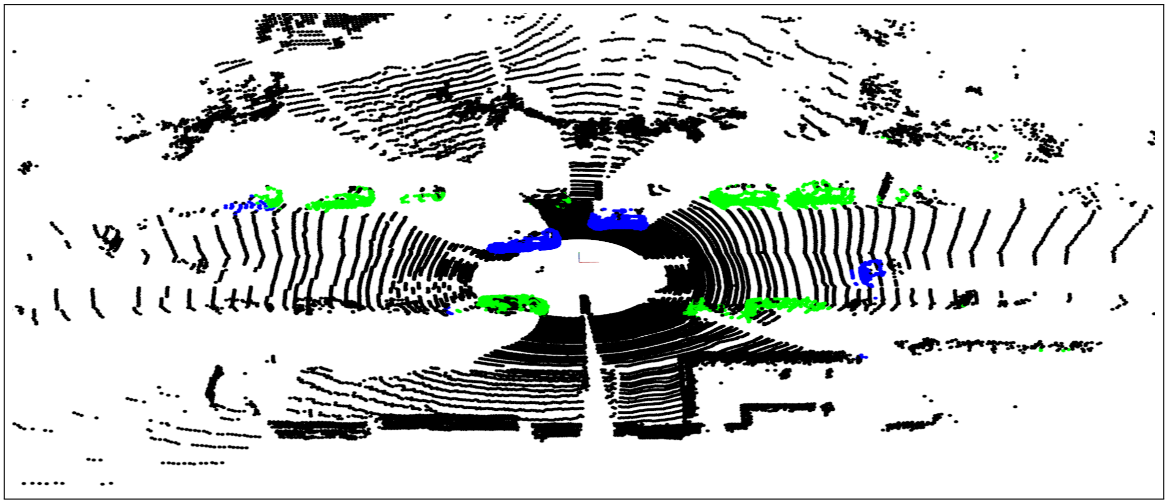

Fig. 4. A LiDAR scan from Scenario-B. Points classified as non-movable are shown in black, points classified as movable are shown in green and points classified as dynamic are shown in blue.

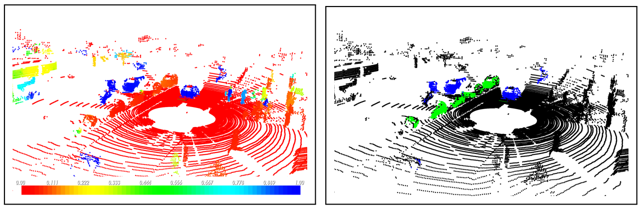

Fig. 5. Bayes filter correction for dynamic class. The image on the left is a visual representation of the *dynamicity score* and right image is the classification results, where non-movable points are shown black, movable points are shown in green and dynamic points are shown in blue. The color bar shows the range of colors for the score ranging from 0 to 1. The misclassified points in the left image (left corner) are correctly classified as non-movable after the addition of the object information.

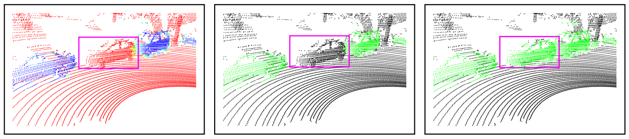

Fig. 6. Bayes filter correction for movable class. The image on the left is a visualization of *objectness* score, where blue points corresponds to movable class. The purple rectangle highlights movable points with low score. The image in the middle and right shows classification of Bayes filter, where green points correspond to movable class. For the middle image, the object information was not recursively updated (second experiment) in the filter framework. For the right image we use the proposed Bayes filter approach (experiment 1). The points having low score is correctly classified as movable since the filter considers the current measurement and the previous measurements, which shows the significance of updating the predictions from the neural network within the filter.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[13] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[16] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] Frank Moosmann and Christoph Stiller. Joint self-localization and tracking of generic objects in 3d range data. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[19] Tayyab Naseer, Benjamin Suger, Michael Ruhnke, and Wolfram Burgard. Vision-based markov localization across large perceptual changes. In *Proc. of the IEEE European Conference on Mobile Robots (ECMR)*, 2015.

[20] G. L. Oliveira, W. Burgard, and T. Brox. Efficient deep models for monocular road segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[21] François Pomerleau, Philipp Krusi, Francis Colas, Paul Furgale, and Roland Siegwart. Long-term 3d map maintenance in dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[22] N Dinesh Reddy, Prateek Singhal, and K Madhava Krishna. Semantic motion segmentation using dense crf formulation. In *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing*, page 56. ACM, 2014.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[24] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, 2015.

[25] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. What could move? finding cars, pedestrians and bicyclists in 3d laser data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4038–4044. IEEE, 2012.

[26] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. Model-free detection and tracking of dynamic objects with 2d lidar. *The International Journal of Robotics Research (IJRR)*, 34(7), 2015.

[27] Allan Zelener and Ioannis Stamos. Cnn-based object segmentation in urban lidar with missing points. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 417–425. IEEE, 2016.