Airborne Particle Classification in LiDAR Point Clouds Using Deep Learning

Leo Stanislas*, Julian Nubert*, Daniel Dugas, Julia Nitsch, Niko Sünderhauf, Roland Siegwart, Cesar Cadena, and Thierry Peynot

Abstract LiDAR sensors have been very popular in robotics due to their ability to provide accurate range measurements and their robustness to lighting conditions. However, their sensitivity to airborne particles such as dust or fog can lead to perception algorithm failures (e.g. the detection of false obstacles by field robots). In this work we address this problem by proposing methods to classify airborne particles in LiDAR data. We propose and compare two deep learning approaches, the first is based on voxel-wise classification, while the second is based on point-wise classification. We also study the impact of different combinations of input features extracted from LiDAR data, including the use of multi-echo returns as a classification feature. We evaluate the performance of the proposed methods on a realistic dataset with the presence of fog and dust particles in outdoor scenes. We achieve an F1 score of 94% for the classification of airborne particles in LiDAR point clouds, thereby significantly outperforming the state-of-the-art. We show the practical significance of this work on two real-world use cases: a relative pose estimation task using point cloud matching, and an obstacle detection task. The code and dataset used for this work are available online[1].

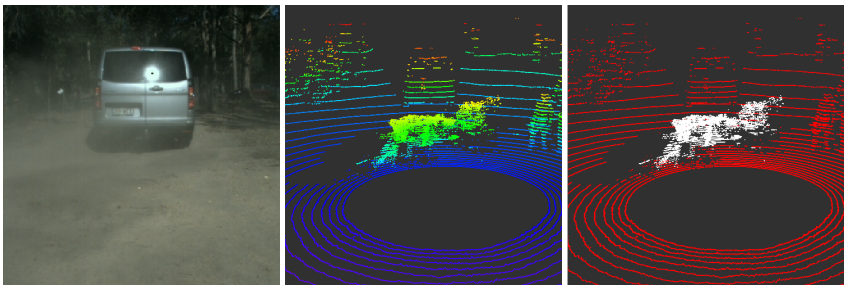

Fig. 1: Left: Image of an experimental scene with dust behind a car. Middle: LiDAR point cloud with dust corruption (colored by height). Right: Predicted particles (white) and non-particles (red).

Julian Nubert, Daniel Dugas, Julia Nitsch, Roland Siegwart, Cesar Cadena
ETH Zurich, Zurich, Switzerland e-mail: nubertj@ethz.ch

Julia Nitsch
Ibeo Automotive Systems, Hamburg, Germany

Leo Stanislas, Niko Sünderhauf, Thierry Peynot
Queensland University of Technology, Brisbane, Australia, e-mail: l.stanislas@qut.edu.au

* indicates equal contributions.

[1] https://leo-stan.github.io/particles_detection_fsr

1 Introduction

LiDAR sensors are used in many areas of robotics such as obstacle detection [2, 10], localization [3, 9], and semantic mapping [7, 8]. In cases of weak or changing illumination or when high spatial resolution is needed, LiDARs often have an advantage compared to RADAR sensors or cameras [11]. However, LiDAR sensors are sensitive to particles in the air such as dust, fog, or smoke [13, 14]. This impacts a given point cloud as depicted in Fig. 1. Present-day algorithms are often suffering from corruptions of this nature. For instance, *Boss* - the winner of the DARPA Urban Challenge - was led to a (temporary) stop because a dust cloud in the LiDAR data was perceived as an obstacle [24]. This work aims to detect LiDAR points generated by airborne particles to reduce their impact on robotics algorithms.

Limited work has been done in this field. Gerardo et al. [6] used a RADAR and LiDAR to perform sensor data fusion and discarded the LiDAR data whenever there was an inconsistency in the data between the sensors, exploiting the fact that RADAR data is not affected by airborne particles. Peynot and Kassir [12] used a similar approach by detecting discrepancies between the LiDAR range and camera intensity gradients. However, both of these approaches require multiple types of sensors. A classification approach [20] was proposed using a LiDAR sensor, by predicting individual LiDAR points generated by airborne particles, however, it was only tested on traditional classifiers (*Support Vector Machine* and *K-Nearest Neighbor*). In our previous work [21], we pursued a similar approach to [20] but evaluating more modern classifiers. We obtained promising results and showed that a voxel-based deep learning classifier outperformed traditional classifiers such as *Random Forest* or *Support Vector Machine*.

This paper expands on the LiDAR-based classification approach. Since the impact on LiDAR data across different types of particles is comparable, we do not make the distinction between types of airborne particle and define two classes for each LiDAR point: *particle* (e.g. dust or smoke) or *non-particle* (e.g. solid object). We present and compare two deep-learning classification approaches:
Voxel-wise classification: Each point cloud is discretized into 3D voxels. A Neural Network architecture, significantly improved from our previous work [21], is used to perform classification on a voxel level.
Point-wise classification: Each point cloud is formatted into a 2D LiDAR image representation with each pixel corresponding to an individual LiDAR return. A *Convolutional Neural Network (CNN)* based on the U-Net architecture [18] performs point-wise classification on the 2D LiDAR image.

We explore different classification features extracted from the LiDAR data including geometry, intensity and multi-echo information and evaluate the best combination of these features for each classification approach. Additionally, we give insights into the ability of our classifiers to distinguish between different types of particles for applications such as autonomous driving where decisions depend on environmental conditions (e.g snow or fog). We evaluate our work on a realistic outdoor dataset with the presence of fog and dust particles in LiDAR point clouds. The dataset is labeled semi-automatically using a background subtraction method

based on occupancy grids, which allowed for increasing the amount of labeled data from our previous work [21] by a factor of ten. The two approaches presented in this work significantly outperform the state of the art in the classification of airborne particles in LiDAR point clouds. This is also the first demonstration of a benefit in using LiDAR multi-echo returns as a classification feature. Additionally, our point-wise classification approach is a novel use of a *CNN* on 2D LiDAR images to detect airborne particles. Finally, we demonstrate the significance of this work on two robotics use cases; a relative pose estimation task and an obstacle detection task, both in the presence of airborne particles. We show a clear performance increase in each task thanks to our approaches.

The remainder of this paper is structured as follows. In Sec. 2 we discuss the related work. Sec. 3 introduces the methodologies of our classification approaches. In Sec. 4, we provide details on our experimental setup while in Sec. 5 we present our experimental results. Finally, in Sec. 6 we conclude and consider future work.

2 Related Work

The choice of input features is an important part of any classification approach and LiDAR data contains various types of information. The geometry information is contained in the 3D position of each LiDAR point. Lalonde et al. [8] used the geometry information in the form of a *Principal Component Analysis (PCA)* to differentiate ground surfaces from non-ground surfaces and unstructured elements. The 3D position information of each LiDAR point was also used in [27] as part of a deep learning approach. The intensity (also referred to as remittance) information is partly correlated to the type of material hit by a LiDAR return. This was used in multiple terrain classification applications [23, 25] as well as for object detection [1, 27]. Finally, most modern LiDAR sensors are capable of returning multiple echos for each LiDAR ray (multi-echo). This was used as a classification feature in [17] in a terrain classification task. However, the authors did not find this feature to be beneficial due to the low proportion of meaningful multi-echo returns in their experiments. In our previous work [21] we used a combination of geometry and intensity information to detect airborne particles in LiDAR point clouds. In this work we investigate various combinations of geometry, intensity and multi-echo information.

Multiple deep learning approaches have been used to classify LiDAR point clouds. A voxel-based approach was developed by Zhou et al.[27], achieving state-of-the-art performance on the object detection task in the KITTI dataset [5]. 3D voxels are used to aggregate classification features from LiDAR points. Using voxels provides the ability to change the resolution of the data representation. This approach was adapted in our previous work [21] to perform the voxel-wise classification of airborne particles in LiDAR point clouds. While the results were promising, the classifier could not exploit the geometry information. In this work, we improve this architecture by adding three sequential 3D convolutions to learn the spatial structure of the data. Alternatively, Qi et al. developed a point-based classification approach on unordered LiDAR point clouds for semantic segmentation of indoor scenes [15]. However, this approach does not make use of the local structure of the

data. Although this shortcoming was later addressed with a hierarchical recursive approach [16], this increased the computational cost of inference. Another point-based classification approach was proposed by Zhang et al. [26] using a *Convolutional Neural Network (CNN)* based on the *U-Net* architecture [18] on a 2D image representation of the LiDAR point clouds. Their image representation was based on a bird's eye view of a 3D voxel grid with the vertical axis as the feature channel. This approach outperformed the aforementioned point-based approach from Qi et al. on semantic segmentation tasks in two different datasets. In this work, we also use the *U-Net* architecture to perform point-wise classification, however, our LiDAR image representation is a 2D projection of the 3D point cloud into a 2D image where each pixel corresponds to a LiDAR return. This representation is advantageous with constant size and density regardless of the observed scene.

3 Airborne Particle Classification in LiDAR Point Clouds

We structure our binary classification approach as a four step process illustrated in Fig. 2, taking a LiDAR point cloud as input and returning the same point cloud with a class label associated to each point. First, the features are selected from the point cloud information returned by the sensor, followed by a formatting step to prepare the data for a particular representation depending on the classification approach. The neural network then predicts the class of the input. Finally, a post-processing step is applied to convert the data from the prediction representation back to the original point cloud format with predicted labels added.

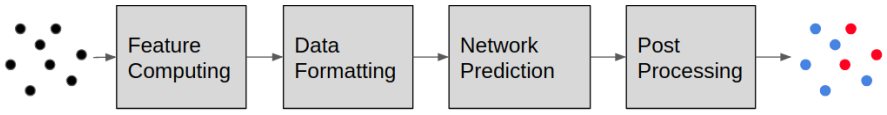

Fig. 2: The four stages of the point cloud classification process.

3.1 Classification Input Features

One of the main objectives of this work is to evaluate the best combination of LiDAR-based features for the classification of airborne particles. We investigate three different types of features:

Geometry: With the high density of the point clouds provided by modern 3D LiDAR sensors, the geometry (i.e. 3D position of points) provides information on the shape of elements in the scene. This information is especially useful when using convolutions in a neural network architecture. Airborne particles will generally produce an unstructured cloud while other elements will be more structured (e.g walls, cars).

Intensity: Intensity values depend on the amount of light measured by the LiDAR sensor. Airborne particles tend to return very low intensity values due to their small size, scattering most of the light emitted by the sensor. This helps to differentiate particles from surfaces made of a material with higher reflectivity such as grass,

concrete, and wood.

Multi-Echo: Different echoes can be measured when multiple peaks are detected in the light intensity of a single light ray at distinct ranges. This phenomenon is common for airborne particles where a first echo can be returned from a cloud of particles, while a subsequent one can go through that cloud and hit an object behind it [14]. Fig. 3 shows that most *non-particle* elements (red) have both echoes matching (green) whereas *particles* (white) tend to return a first echo (blue) while the obstacles behind return a second echo (purple).

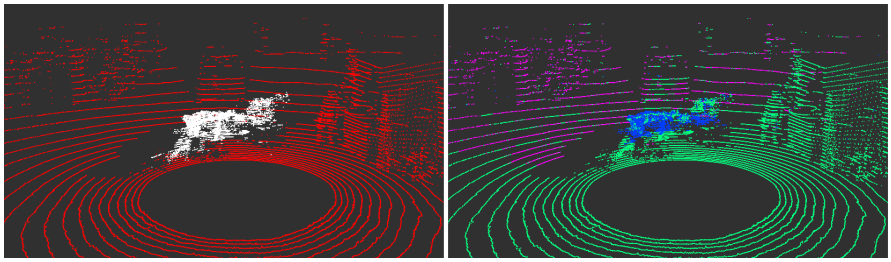

Fig. 3: LiDAR point cloud from the example scene from Fig. 1. Left: Colored by label with *particles* in white and *non-particles* in red. Right: Colored by echo feature with green when both echoes match, blue for the first echo return, and purple for the second echo return.

3.2 Voxel Classification Approach

Representing a LiDAR point cloud with voxels enables various levels of abstraction in the data, as well as concatenating features between neighboring points. This approach is an incremental improvement of the voxel-based approach presented in our previous work [21].

Data Representation: We discretize each LiDAR point cloud into equally spaced non-overlapping 3D voxels of equal size $[v_X, v_Y, v_Z]$. We only create a voxel if there is at least one LiDAR point inside it. This forms a sparse voxel grid containing N voxels for a given LiDAR point cloud, with a maximum number of voxels in each dimension defined by $[W, H, D]$. The voxel grid is centered around the LiDAR sensor. Each voxel contains a maximum of T LiDAR points $p = [x, y, z, a, e]$ with x, y, z being the absolute position, a the intensity with values in the range $[0; 100]$, and e the echo status with three possible values: 0 when both strongest and last echoes match, and when the echoes are different: 1 when the echo is the strongest and 2 when it is the last echo. The input feature vector of our voxel classification architecture is defined as

$$\mathbf{F_{in}} = \{\{[x_i - \hat{x}, y_i - \hat{y}, z_i - \hat{z}, a_i, e_{0,i}, e_{1,i}, e_{2,i}]\}_{i=1\ldots T_n}\}_{n=1\ldots N}, \tag{1}$$

where $\hat{x}, \hat{y}, \hat{z}$ is the average of the position of all points in the voxel. $x_i - \hat{x}, y_i - \hat{y}, z_i - \hat{z}$ is the position of each LiDAR point relative to the other points in the same voxel. a_i is the intensity return, and $e_{0,i}, e_{1,i}, e_{2,i}$ are the one-hot encoded values corresponding to the echo value each taking the binary value 0 or 1. This makes our input

feature tensor F_{in} of shape $[N, T, 7]$ corresponding to one scan of the LiDAR sensor where N (the number of voxels created from the scan) is used as batch size. Finally, the predicted label of each voxel is assigned to all LiDAR points inside to provide a label per LiDAR point for this evaluation.

Network Architecture: Our network architecture is based on the *VoxelNet* architecture from Zhou et al. [27] and is depicted in Fig. 4. Each voxel in the input tensor

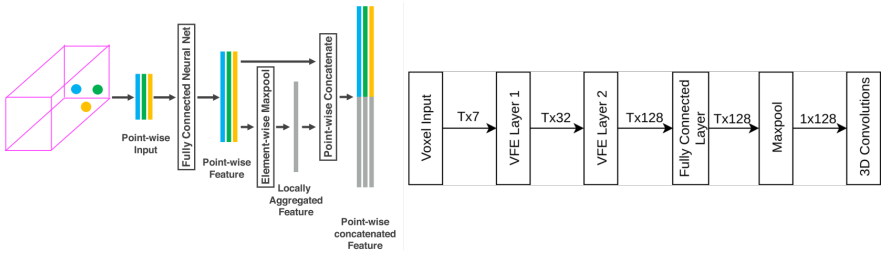

Fig. 4: Left: Voxel Feature Encoding (VFE) layer [27]. Right: Voxel classification architecture.

F_{in} is passed to two *Voxel Feature Encoding (VFE)* modules and a fully-connected layer (see Fig. 4) to learn features within each voxel. We apply a *maxpool* operation on the output of the fully-connected layer to concatenate the features over all T points. Batch-normalization and *Relu*-activation is used after each fully-connected layer inside the *VFE* modules. We apply 3D convolution to the sparse tensor of shape $128 \times W \times H \times D$ formed with the computed features of each voxel. We apply three 3D convolutions sequentially with *Conv1*(128,64,k,s,p), *Conv2*(64,64,k,s,p), and *Conv3*(64,2,k,s,p), where k=(1,1,1) is the kernel size, s=(1,1,1) is the stride, and p=(1,1,1) is the padding. The output of the network is a sparse tensor of shape $2 \times W \times H \times D$ to which we apply a *Softmax* operation on the prediction score. The predicted label for all N voxels in each scan is obtained from this output. We apply a standard scaling operation by removing the mean and scaling to unit variance on the input values of each voxel. We reduce the imbalance in the training data by using a small voxel map around the zone with particles with $W = H = 100$ and $D = 15$ and a voxel size of $v_X = v_Y = v_Z = 0.2m$ during training.

3.3 Point Classification Approach

Data Representation: We render each 3D LiDAR scan in a 2D LiDAR image with each pixel corresponding to an emitted ray and the rows and columns of this image correspond to the vertical and horizontal angular resolution of the LiDAR sensor respectively. This is possible thanks to the inherent LiDAR property of sequentially scanning the environment. Using the angular resolution of the 3D LiDAR sensor, an image-width C can be computed while the height R is dependant on the number of vertical rings of the LiDAR sensor. The horizontal pixel coordinate is obtained by mapping the polar angle of the ring to the image space. Each pixel contains up to four channels corresponding to the range and intensity of both echo returns for a given LiDAR beam. We define this LiDAR image representation as:

$$\mathbf{F_{in}} = \{p_{ij} = [r_{echo1,ij}, a_{echo1,ij}, r_{echo2,ij}, a_{echo2,ij}]\}_{i=1...R, j=1...C}, \quad (2)$$

where $\mathbf{F_{in}}$ is the LiDAR image used as input tensor to the network, p_{ij} is the pixel corresponding to the vertical angle index i and horizontal angle index j. $r_{echo1,ij}$ and $a_{echo1,ij}$ are the range and intensity values for the first echo, while $r_{echo2,ij}$ and $a_{echo2,ij}$ are the range and intensity values for the last echo. R and C are the number of vertical and horizontal angle increments respectively. An example of the mask for the presented representation as an outcome of the labeling can be seen in Fig. 5.

Fig. 5: Mask of LiDAR image (class for each of the pixels). White: *particle*, black: *non-particle*.

Network Architecture: Due to spatial correlations of neighboring pixels, we use the *U-Net* architecture [18] to perform 2D convolutions on the input data. Fig. 6 depicts the structure of this network. *Maxpool* layers are used in the encoder to create smaller features with high expressiveness. In the decoder, the network extracts information from these meaningful features to classify each of the pixels. *Maxpool* and *up-sampling* operations are only performed along the input columns whereas convolutions are applied on both rows and columns with 3x3 filters.

By having four pooling layers, the horizontal resolution is reduced by a factor of $2^4 = 16$ in the encoder. Therefore, the width of the input LiDAR image has to be a multiple of 16 for the output to have the same shape as the input. Since the network is *fully convolutional*, it is able to process inputs of variable sizes. To train

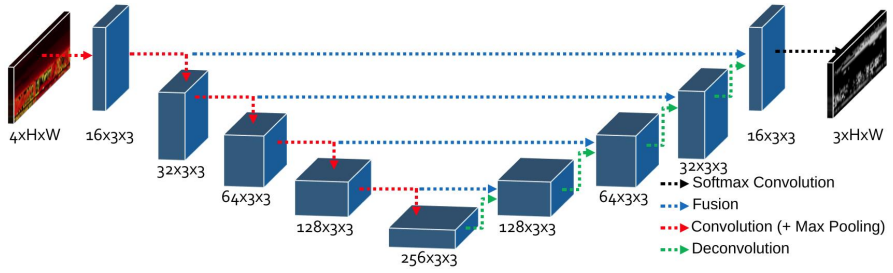

Fig. 6: The U-Net Architecture used for point classification.

the network, we take a sample of size $R = 32$ and $C = 512$ as an input vector to the network chosen randomly from the original LiDAR image. Additionally, the image is horizontally flipped with a probability of 50%. These two data augmentation methods prevent the network from overfitting the training data. To mitigate dataset imbalance, we only use frames with a high ratio of particles for training and a combination of the *binary cross entropy* and the *Dice Loss* [22] as cost function. The latter has shown advantages for training neural networks on imbalanced data.

4 Experimental Setup

4.1 Dataset

To evaluate our methods, we use the dataset from our previous work [21] containing dust and fog particles in 19 different outdoor scenes with elements such as cars, humans, buildings, shrubs, and trees. The sensor data were collected using a Velodyne HDL-32E LiDAR sensor mounted on a Clearpath Husky platform and contains position, intensity, and multi-echo information in the point clouds. Each scene was recorded statically with and without particles to allow for semi-automated labeling using background subtraction. The noise in the LiDAR point 3D position makes a simple point-to-point background subtraction inappropriate. Therefore, we used an occupancy grid representation where a particle-free reference grid was subtracted to the grid generated for each scan to be labeled. We use a Polar representation with logarithmic resolution on the radial axis for the occupancy grid to accommodate for the circular structure of the LiDAR scans and the lowering point density in the far field. This labeling implementation is included with the code for this work. Overall, our labeled dataset contains 16,247 fully-labeled LiDAR scans, which is an increase by a factor of ten from our previous work. Images of the scenes in the dataset are provided with the code for this work.

To evaluate our approaches, we used nine scenes as a training set, two scenes as a validation set, and the remaining eight scenes as a test set. We made sure the scenes were sufficiently different configurations from training to testing to prevent overfitting. All results were computed on the entire test set unless specified otherwise.

4.2 Performance Metrics

We evaluate the performance as a binary classification problem with a positive prediction (P) corresponding to a *particle* and a negative prediction (N) corresponding to a *non-particle*. We compute the *precision* and *recall* scores as $\mathit{precision} = \frac{TP}{TP+FP}$ and $\mathit{recall} = \frac{TP}{TP+FN}$. TP and TN denote the number of true positives and true negative predictions respectively and FP and FN the number of false positive and false negative predictions. Finally, we compute the F1 score as $F1 = 2\frac{\mathit{precision}\cdot\mathit{recall}}{\mathit{precision}+\mathit{recall}}$.

4.3 Evaluation Parameters

We evaluate the voxel-based approach with a map size of $W = H = 400$ and $D = 15$ and a voxel size of $v_X = v_Y = v_Z = 0.2m$. An overview of the parameters for both neural network architectures are noted in Table 1.

5 Experimental Results

Each evaluation is computed by training multiple networks on our training and validation sets and averaging their performance on the prediction of our test set. This

Table 1: Parameters configurations for both classification approaches.

	Voxel Classification	Point Classification
Optimizer	Adam	Adam
Learning Rate	0.001	0.001
Optimizer Parameters	$\beta_{1/2}$=(0.9, 0.9), ε=10^{-08}, weight decay=0	$\beta_{1/2}$=(0.9, 0.999), ε=0, weight decay=0
Training	10,000 iterations, batch size=N	271,150 iterations, batch size=32
Loss Function	Binary Cross Entropy	Combination of BCE and DL
Trainable Parameters	355,052	1,857,859
Input size	$T = 35$	$R = 32$, $C_{pred} = 2160$, $C_{train} = 512$

provides a more meaningful performance interpretation by reducing the influence of statistical training outliers.

5.1 Voxel Classification Evaluation

We evaluate the performance of this approach for six different configurations of input features. For each combination, we train five models and average their performance on our test set. The results are presented in Table 2. We compare the performance to the voxel classification approach from our previous work evaluated on the dataset used in this paper. Most combinations of features outperform the architecture from our previous work. The intensity and geometry features perform better on their own than combined. Adding multi-echo to the intensity feature decreases performance, however, it increases performance when added to the geometry feature. Finally, this approach performs best by combining all three features achieving a maximum F1 score of 91%.

Table 2: Averaged voxel classification performance for all six input configurations and our previous work.

	Previous Work [21]	Geometry	Intensity	Geometry Intensity	Geometry Multi-Echo	Intensity Multi-Echo	Geometry Intensity Multi-Echo
Precision	0.37	0.53	0.64	0.42	0.77	0.31	**0.89**
Recall	**0.97**	0.90	0.91	**0.97**	0.95	**0.97**	0.93
F1	0.54	0.65	0.75	0.58	0.85	0.44	**0.91**

5.2 Point Classification Evaluation

The averaged performance of our point classification approach for different input feature combinations is presented in Table 3. The multi-echo feature improves performance when added to the geometry feature. Finally, this approach performs best by combining all three features achieving a maximum F1 score of 91%.

Table 3: Averaged point classification performance for all six input configurations.

	Geometry	Intensity	Geometry Intensity	Geometry Multi-Echo	Intensity Multi-Echo	Geometry Intensity Multi-Echo
Precision	0.65	0.87	0.87	**0.94**	0.82	0.92
Recall	0.96	0.90	**0.96**	0.95	0.93	**0.96**
F1	0.73	0.89	0.91	**0.94**	0.87	0.93

formance when combined with geometry. The geometry is needed to make the best of this architecture. We obtain the highest scores for our point classification approach using *Geometry* and *Multi-Echo* achieving an F1 score of 94%. We observe F1 scores of more than 85% for using intensity values only.

5.3 Performance Comparison

In Table 4, we present the classification performance using the best input feature combination for each approach and compare them to the state-of-the-art. The performance of the voxel-based classification and the point-based classification is comparable while both are clearly outperforming the state of the art. The input features used to obtain these results will be used in the rest of this paper.

Table 4: Comparison of classification results between our approaches with the best configurations of input features and the state of the art.

	Previous Work [21]	Voxel Classification	Point Classification
Precision	0.37	0.89	**0.94**
Recall	**0.97**	0.93	0.95
F1 score	0.54	0.91	**0.94**

5.4 Differentiate Dust from Fog Particles

For some applications, it is beneficial to differentiate between different types of particles. For instance, firefighters might only be interested in smoke clouds when assessing a bush fire while removing other types of particles. We evaluate whether our approaches are able to differentiate the type of particle by defining three classes *dust*, *fog*, and *non-particle*. Our voxel-based approach was not able to successfully learn the difference between the fog and dust particles. This is likely due to the simpler structure of the network. However, our point classification architecture was successfully detecting the difference between the two types of particles with F1 scores of 96% for the *fog* class and 92% for the *dust* class.

5.5 Use Cases

5.5.1 Pose Estimation Task

LiDAR sensors are widely used for robot localization tasks [3]. A well-known approach to perform pose estimation using LiDAR data is using Point Cloud Matching to estimate the transformation between two point clouds taken from different positions. However, this matching process can be altered by the presence of moving airborne particles in the point cloud. We demonstrate the significance of our work by showing a decrease in pose estimation error and faster convergence when removing airborne particles detected in LiDAR point clouds prior to performing the matching step using a point-to-point *ICP* algorithm [19].

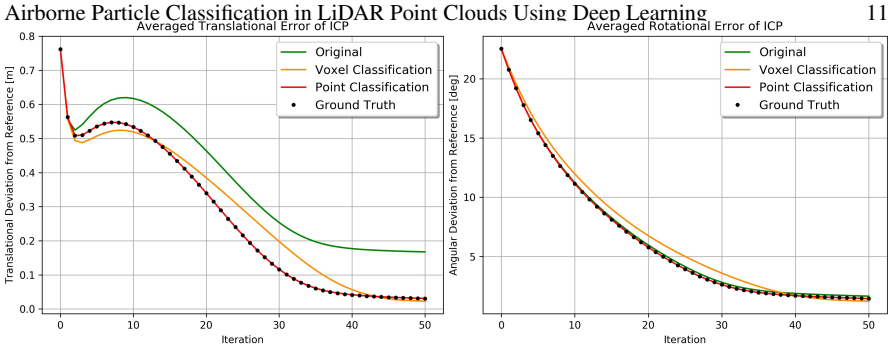

Fig. 8: Left: averaged translational error. Right: averaged rotational error. Different configurations: original point cloud, and filtered point cloud using ground truth, voxel classification, point classification.

We introduce an erroneous translation and rotation to our statically recorded dataset to produce corrupted scans and compare them to a reference scan of the scene free of particles. We apply the ICP algorithm to match the corrupted scan to the reference and measure the remaining translation and rotation error. The error introduced in each scan is a random rotation in the interval of $[0°, 45°]$ around the vertical axis followed by random translations in the interval of $[0m, 1m]$ along the vertical and horizontal axes. We perform this evaluation on 10% of the total scans in our test set and compute the average error in translation and rotation. This evaluation is performed on the original point cloud containing all particles and three filtered point clouds for which particles have been removed according to: 1) the ground truth, 2) the point classification approach, and 3) the voxel classification approach. The ground truth was obtained by performing ICP on a point cloud from which particles have been removed (filtered) using the ground truth labeled data. The results are presented in Fig. 8 where the ideal value is a nil translation and rotation since the sensor is static between scans. The impact of airborne particles in LiDAR point clouds on the translational matching of the ICP algorithm generates an error of $17cm$ after convergence, compared to $2.5cm$ when using the particle-free point clouds from the ground truth. The impact of airborne particles on the rotational error is less important than for the translational error with the original point cloud providing comparable performance to the particle-free point clouds. For both translational and rotational errors, removing particles in the LiDAR point cloud using our approaches provides a gain in performance comparable to the ground truth.

5.5.2 Obstacle Detection Task

LiDAR sensors are widely used to perform obstacle detection in mobile robotics applications [24]. However, airborne particles in LiDAR point clouds can lead to false obstacles being generated, inducing loss of performance. We evaluate how removing airborne particles in LiDAR point clouds can help obstacle detection in outdoor scenarios using an occupancy grid [4]. The occupancy grid we compute is a 2D array of $30m \times 30m$ and $0.4m$ resolution, with each cell either *unobserved*, *occupied*,

or *free*. All cells are initially *unobserved*. A cell is considered *occupied* if a LiDAR point in its 2D boundary is higher than $2m$ or if the height difference between the lowest and highest point is greater than $0.3m$, otherwise it is *free*. Fig. 9 shows the difference between an occupancy grid with the original LiDAR points in the presence of airborne particles and an occupancy grid for which the points detected as particles have been removed. Most false obstacles generated by the particles in the original point cloud are removed in the occupancy grid generated from the filtered point cloud. The cells behind the particles are still considered *unobserved*, leaving the same opportunity to the system to later detect an obstacle that might be hidden behind the particles as before the removal of LiDAR points. We evaluate two as-

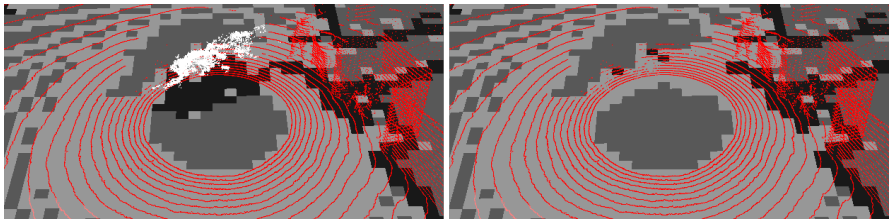

Fig. 9: Occupancy grids (occupied: black cells, free: light grey cells, unobserved: dark grey) and predicted point cloud with *particles* in white and *non-particles* in red for the scene in Fig 1. Left: Original point cloud. Right: Point cloud with predicted *particles* removed.

Table 5: Performance of our classification approaches on the obstacle detection task.

	Original Point Cloud	Voxel Classification	Point Classification
False Obstacle Prevented (higher is better)	0%	89%	96%
True Obstacle Removed (lower is better)	0%	0.1%	0.4%

pects when filtering LiDAR point clouds for obstacle detection: the percentage of false obstacles removed by correctly identifying the particles, and the percentage of true obstacles removed due to misclassification of a *non-particle* for a *particle*. Table 5 shows the performance of each approach. Both approaches are able to prevent the majority of false obstacles by detecting and removing airborne particles, with the point classification approach preventing up to 96% of false obstacle cells in the occupancy grid. Only a small percentage of true obstacles cells are removed from the occupancy grid, with the voxel classification approach only removing 0.1% of true obstacles.

6 Conclusion and Future Work

In this paper we proposed two deep-learning approaches to classify airborne particles in LiDAR point clouds, based on voxels and on points (projected into a LIDAR image), respectively. We also presented a thorough validation on realistic experimental dataset containing fog and airborne dust. First we evaluated the best combination of input features for each approach. The point classification approach

performed best using geometry and multi-echo features with an F1 score of 94%, while the voxel classification approach performed best when combining geometry, intensity, and multi-echo features with an F1 score of 91%. Both approaches significantly outperformed the state of the art. These results indicate that the multi-echo returns of the LiDAR sensor is beneficial as an input feature for classifying airborne particles. Both approaches provide comparable classification performance overall; however, the choice between the two strategies depends on the data representation, size of the network. We also show that it may depend on the task to perform. We can recommend the use of the voxel-based approach for a system already using a voxel representation or requiring a smaller network size due to hardware limitation. This approach seems to be well suited for obstacle detection tasks: in this paper the proposed method was able to clear a significant number of false obstacles (created by airborne particles), while removing only a very small number of true obstacles cells in an occupancy grid. We recommend the point classification approach to get the best possible classification performance at the cost of a bigger network. We also showed that this was the only approach of the two capable of differentiating between types of particles. When filtering a point cloud prior to a pose estimation task, both approaches provide a similar performance gain. Limits of the two approaches are symptomatic of deep learning and given as generalization to other particles, sensors and environments.

Future work will evaluate the proposed methods on other types of particles such as smoke or snow. Evaluating the uncertainty of the prediction would also be beneficial, as in any robotics application. Finally, adding a notion of time in the network prediction is likely to be useful to learn and predict the movement aspects of airborne particles.

Acknowledgments
Parts of the computational resources used in this work were provided by the High Performance Computing Facility of the Queensland University of Technology, Brisbane, Australia. The authors would like to thank Ankit Dhall and Benjamin Le Bigot for their help with data labeling.

References

1. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. CoRR (2016)
2. Darms, M.S., Rybski, P.E., Baker, C., Urmson, C.: Obstacle detection and tracking for the Urban challenge. IEEE Transactions on Intelligent Transportation Systems (2009)
3. Dube, R., Cramariuc, A., Dugas, D., Nieto, J., Siegwart, R., Cadena, C.: SegMap: 3d segment mapping using data-driven descriptors. In: Robotics: Science and Systems (RSS) (2018)
4. Elfes, A.: Using Occupancy Grids for Mobile Robot Perception and Navigation. IEEE Computer (1989)
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
6. Gerardo-Castro, M.P., Peynot, T., Ramos, F., Fitch, R.: Non-Parametric Consistency Test for Multiple-Sensing-Modality Data Fusion. IEEE International Conference on Information Fusion (2015)

7. Laible, S., Khan, Y.N., Bohlmann, K., Zell, A.: 3D LIDAR- and Camera-Based Terrain Classification Under Different Lighting Conditions. Autonomous Mobile Systems (2012)
8. Lalonde, J.F., Vandapel, N., Huber, D.F., Hebert, M.: Natural Terrain Classification Using Three-Dimensional Ladar Data for Ground Robot Mobility. Journal of Field Robotics (2006)
9. Levinson, J., Thrun, S.: Robust vehicle localization in urban environments using probabilistic maps. IEEE International Conference on Robotics and Automation (2010)
10. Montemerlo, M., Jan, B., Suhrid, B., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D.: Junior: The Stanford Entry in the Urban Challenge. Journal of Field Robotics (2008)
11. Motaz Khader, S.C.: An introduction to automotive lidar. Tech. rep., Texas Instruments (2018)
12. Peynot, T., Kassir, A.: Laser-camera data discrepancies and reliable perception in outdoor robotics. IEEE International Conference on Intelligent Robots and Systems (2010)
13. Peynot, T., Scheding, S., Terho, S.: The Marulan Data Sets:Multi-Sensor Perception in Natural Environment With Challenging Conditions. The International Journal of Robotics Research (2010)
14. Phillips, T.G., Guenther, N., McAree, P.R.: When the dust settles: The four behaviors of lidar in the presence of fine airborne particulates. Journal of Field Robotics (2017)
15. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
16. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems (2017)
17. Reymann, C., Lacroix, S.: Improving lidar point cloud classification using intensities and multiple echoes. International Conference on Intelligent Robots and Systems (IROS) (2015)
18. Ronneberger, O., P.Fischer, Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Springer Medical Image Computing and Computer-Assisted Intervention (MICCAI) (2015)
19. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA) (2011)
20. Shamsudin, A.U., Ohno, K., Westfechtel, T., Takahiro, S., Okada, Y., Tadokoro, S.: Fog removal using laser beam penetration, laser intensity, and geometrical features for 3d measurements in fog-filled room. Advanced Robotics
21. Stanislas, L., Sünderhauf, N., Peynot, T.: Lidar-based detection of airborne particles for robust robot perception. Australasian Conference on Robotics and Automation (ACRA) (2018)
22. Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Cardoso, M.J.: Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. CoRR (2017)
23. Suger, B., Steder, B., Burgard, W.: Terrain-adaptive obstacle detection. IEEE International Conference on Intelligent Robots and Systems (2016)
24. Urmson, C., et al.: Autonomous Driving in Urban Environments: Boss and the Urban Challenge. Journal of Field Robotics (2008)
25. Wurm, K.M., Kümmerle, R., Stachniss, C., Burgard, W.: Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. IEEE International Conference on Intelligent Robots and Systems (2009)
26. Zhang, C., Luo, W., Urtasun, R.: Efficient convolutions for real-time semantic segmentation of 3d point clouds. 2018 International Conference on 3D Vision (3DV) (2018)
27. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. CoRR (2017)