AutoGrAN: Autonomous Vehicle LiDAR Contaminant Detection using Graph Attention Networks

Grafika Jati
DEI Department, University of Bologna
Italy
grafika.jati2@unibo.it

Martin Molan
DEI Department, University of Bologna
Italy
martin.molan2@unibo.it

Junaid Ahmed Khan
DEI Department, University of Bologna
Italy
junaidahmed.khan@unibo.it

Francesco Barchi
DEI Department, University of Bologna
Italy
francesco.barchi@unibo.it

Andrea Bartolini
DEI Department, University of Bologna
Italy
a.bartolini@unibo.it

Giuseppe Mercurio
FEV Italia s.r.l.
Italy
mercurio_g@fev.com

Andrea Acquaviva
DEI Department, University of Bologna
Italy
andrea.acquaviva@unibo.it

ABSTRACT

Extreme conditions and the integrity of LiDAR sensors influence AI perception models in autonomous vehicles. Lens contamination caused by external particles can compromise LiDAR object detection performance. Automatic contaminant detection is important to improve reliability of sensor information propagated to the user or to object detection algorithms. However, dynamic conditions such as variations in location, distance, and types of objects around the autonomous vehicle make robust and fast contaminant detection significantly challenging.

We propose a method for contaminant detection using voxel-based graph transformation to address the challenge of sparse LiDAR data. This method considers LiDAR points as graph nodes and employs a graph attention layer to enhance the accuracy of contaminant detection. Additionally, we introduce cross-environment training and testing on real-world contaminant LiDAR data to ensure high generalization across different environments. Compared with the current state-of-the-art approaches in contaminant detection, our proposed method significantly improves the performance by as much as 0.1575 in F1-score. Consistently achieving F1 scores of 0.936, 0.902, and 0.920 across various testing scenarios, our method demonstrates robustness and adaptability. Requiring 128 milliseconds on a AMD EPYC 74F3 CPU for the end-to-end process, our method is well-suited for an early warning system, outperforming human reaction times, which require at least 390 milliseconds to detect hazards. This significantly contributes to enhancing safety and reliability in the operations of autonomous vehicles.

ICPE '24 Companion, May 7–11, 2024, London, United Kingdom

ACM ISBN 979-8-4007-0445-1/24/05.
https://doi.org/10.1145/3629527.3652896

CCS CONCEPTS

• **Computing methodologies** → **Scene anomaly detection**; **Vision for robotics**; **3D imaging**.

KEYWORDS

Advanced driver assistance systems (ADAS), autonomous vehicle perception systems, LiDAR Point Cloud reliability, LiDAR contamination and Noise detection, Graph Processing, Graph Attention Networks

ACM Reference Format:
Grafika Jati, Martin Molan, Junaid Ahmed Khan, Francesco Barchi, Andrea Bartolini, Giuseppe Mercurio, and Andrea Acquaviva. 2024. AutoGrAN: Autonomous Vehicle LiDAR Contaminant Detection using Graph Attention Networks. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3629527.3652896

1 INTRODUCTION

The LiDAR becomes one of the main sensors for the perception model in autonomous vehicles. LiDAR point clouds provide 3D information about the surrounding environment, offering an in-depth picture of the objects around it [12, 16]. Perception models such as object detection, object recognition, scene reconstruction, motion estimation, and path planning are essential. The integrity of LiDAR data influences the performance of perception models in autonomous vehicles. Failure of integrity can result in catastrophic errors or failures.

LiDAR data integrity is essential in edge case conditions, such as sensor cover contamination. While the sensor may be in good condition, its perception can be limited by unexpected situations affecting its surface or cover. Beyond severe weather conditions, the sensor cover's cleanliness can significantly limit perception

capabilities. This limitation is evidenced by a decreased object detection accuracy, as highlighted in a recent benchmark study [5]. This benchmark found that weather-related contamination, including rain, snow, and fog, reduces object detection performance. For instance, the PV-RCNN method achieved an Average Precision (AP) of 84.39 under clean conditions, which dropped to 51.35 with rain-affected data. That study assessed the impact of adverse weather conditions leading to particle accumulation on the sensor cover. In the same survey, using synthetic corruption data (Gaussian, uniform, and impulse noise only), object detection performance decreased by up to 20% of AP. On the other hand, sensor cover contamination can also result from external factors, such as water, dust, oil, mud, and other external sources.

The impact of these contaminants on the LiDAR lens represents a major threat to sensor data integrity. This emphasizes the need for a contaminant detection technique as an early warning before executing object detection. Contaminant detection is also crucial for triggering automatic cleaning procedures if contaminants are present. Perception anomalies remain a hot topic nowadays, as summarized in a survey of several techniques for anomaly detection [2]. However, research in contaminant detection still needs to be improved due to the challenge of scarcity of real-world LiDAR-contaminated datasets [14].

A key attribute of LiDAR point clouds is the intensity of each point, which reflects the return strength of a laser beam after striking an object. Different materials reflect varying intensities and an object's distance can diminish this intensity [1]. Typically, the longer the distance from the sensor to the object, the lower the object's intensity. Thus, detecting contaminants becomes challenging when relying solely on intensity values in dense or sparse point clouds, as contaminant presence can be confused with large distance points. Thus, a distance and environment-invariant contaminant detection model is highly needed for real-world environments.

In this paper, we employ graph representation for contaminant detection with the following contributions: i) We are the first to tackle LiDAR contaminant detection using graph representation by comparing different approaches. ii) We improve state-of-the-art approaches in terms of accuracy and tested in various contaminant types from various environmental settings. In particular, detecting contamination is challenging due to LiDAR sparsity. LiDAR point clouds at longer distances are more sparse than closer ones. To tackle this sparsity, we propose a voxel-based graph transformation to deal with contaminated LiDAR point clouds.

Specifically, we employ a Graph Neural Networks (GNNs)-based method that treats LiDAR points as connected nodes in a graph. GNNs are specialized neural networks designed for graph-structured data, wherein nodes represent entities and edges represent relationships. They excel in capturing spatial information by aggregating data from neighboring nodes, enhancing their understanding of relationships and dependencies within the graph. In a LiDAR point cloud, where an object is present, the points would be denser and contain more information than in a no-object area. This leads us to employ graph attention networks. Additionally, we deal with large regions of the environment and high-resolution LiDAR, which generates a large number of points. Processing all points as nodes in the graph increases computation time, so representing some points as a single node helps reduce the number of nodes. This approach can be defined as point voxelization. Finally, we introduce a graph representation of voxel point clouds using graph attention networks (voxel-GAT) for contaminant detection on autonomous vehicles, namely AutoGrAN.

The proposed method efficiently represents sparse LiDAR data and outperforms the state-of-the-art approach regarding classification performance and computational efficiency. We utilize three datasets, namely *"5m," "10m," and "20m,"* to reflect variations in location, distance, surrounding objects, and contaminant sources. Our approach employs a cross-data validation scheme, creating three models based on each dataset and testing them using a different dataset. Compared to the 3D CNN-based detection method, our proposed method improves the F1-score from 0.778, 0.850, and 0.840 to 0.936, 0.902, and 0.920, respectively, across six possible train-test dataset scenarios.

In terms of computational performance, our proposed method achieves competitive end-to-end processing times, from graph construction until obtaining contaminant results, all within an average of 128 milliseconds per point cloud frame running on CPU AMD EPYC 74F3, single core CPU clock speed 3.2G. This efficiency establishes the proposed method as a promising candidate for an early warning system in autonomous vehicle downstream tasks. As indicated in [10] and [22], the system has to complete the end-to-end processing within a latency of around 100 milliseconds to be effective. Additionally, considering human reaction times, which range from 390 to 600 milliseconds for detecting incoming hazards [13] [7], the proposed method's computation times is a significant advantage. It delivers promising results in contaminant detection for LiDAR on autonomous vehicles, demonstrating high accuracy, robustness, ease to train, and low computational overhead.

2 RELATED WORKS

2.1 LiDAR processing for automotive applications

The perception model is a crucial input for the decision-making and planning of an autonomous vehicle. The research was carried out to increase each perception model's accuracy, robustness, and generalization, e.g., in [9],[21]. To achieve this goal, the development of the Autonomous vehicle model needs to pay attention to various aspects ranging from research to development. In more detail, an AI model's development depends on the data, algorithm, and deployment aspects.

In the paper by Wang et.al [20], it is mentioned that accuracy and robustness are important factors. High accuracy is certainly needed, considering that automotive vehicles require correct decision-making because they involve life-and-death situations. Errors in perception can cause decision-making errors, which can have fatal consequences. Robustness concerns the AI model's ability to perform well in every situation, dynamic environment, dynamic location, and object surrounding. We need to realize that there will be situations and scenarios of uncertainty in the real world. The sensor properties also influence the AI model, especially the perception model, which must be robust to all sensor conditions [3]. The perception model must be robust when deployed and tested in different environments, weather, and traffic situations [25].

Apart from accuracy and robustness, the perception model needs to consider ease of training. Autonomous vehicles must be usable under all conditions, making data collection a significant challenge in terms of both cost and time. The AI training process must be feasible with limited data. We lack datasets for training in every possible situation, including various adverse traffic conditions as well as challenges posed by weather and sensor contamination [24]. Efforts have been made with data degraded by simulators. However, simulating all scenarios, especially unusual ones (edge cases and anomalies), proves difficult. Formal verification using mathematical models can be applied to the created models, but scalability is limited, especially for complex systems like autonomous vehicles. Therefore, every AI model developed needs to undergo real-world testing to validate performance and identify new scenarios that may not have been previously considered. Real-world testing allows for the validation of AI performance and the discovery of previously unrecorded scenarios.

The last aspect to consider regarding AI models is low computational overhead. Autonomous vehicles require fast decision-making, where the perception model serves as input for the reinforcement learning agent in the autonomous vehicle's decision-making process. Therefore, it is crucial to maximize the speed of the subprocesses [17] [23]. Finally, developing LiDAR processing methods should be prioritized to meet the requirements of high accuracy, robustness, ease of training, and low computational overhead.

2.2 LiDAR point cloud transformation for contaminant detection

Driven by applications in autonomous driving, several LiDAR point cloud processing approaches have been developed. These can be categorized based on the type of data representation: 1D, 2D, 3D, and graph. The 1D approach creates a representative vector from the original 3D data, upon which classical machine learning techniques for tabular data are applied.

In research aligned with contaminant classification, the use of 1D transformations was introduced by Heinzler et al., who aggregated features from point clouds such as (x, y, z) for the cartesian and (r, θ, φ) for the spherical coordinates, echo number, intensity, and echo pulse width. In their study, Heinzler et al. attempted to classify weather conditions as clear, rainy, or foggy on LiDAR point clouds, achieving the best accuracy of 97.14% using SVM and KNN [6]. Similarly, the application of 1D transformation and KNN for classifying rain, fog, and snow was proposed by Rivero et al. [19]. However, Rivero placed the LiDAR sensor in a static position. Furthermore, both [19] and [6] did not address the classification of sensor cover contamination.

Another approach involves 2D transformation, which constructs a 2D depth and intensity image from the LiDAR point cloud. James et al. initiated the classification of sensor cover contamination using 2D transformations [8]. They transformed a 2D image of LiDAR data as input for a 2D CNN to classify contaminants like dirt, salt, and frost. The 2D CNN achieved promising results, reaching an accuracy of around 80% in classifying between clean and dirty conditions using a front-view transformation of LiDAR data. However, the data used in [8] was collected from a statically positioned sensor without any objects in front of the acquisition sensor and with the sensor cover fully contaminated. Therefore, it may not accurately reflect real-world environmental conditions.

The 2D transformation inevitably reduces spatial information from the original 3D point cloud. LiDAR processing that uses 3D data directly often employs voxelization techniques. This technique distributes the point cloud coordinates while preserving as much spatial information as possible. The application of 3D voxelization includes its use in PV-RCNN, which leverages 3D convolution as a backbone for 3D object detection [4]. However, the proposal to use 3D convolution for contaminant detection has not yet been established. Nonetheless, we have considered 3D as one of our baseline methods for contaminant detection.

The latest advancement in LiDAR processing methodologies is the introduction of graph processing networks, which aim to represent sparse LiDAR sensor data as graphs. Graph convolutional networks were then applied to this graph-represented data. Graph convolutions can be significantly more efficient than 2D and 3D convolutions on sparse tensors, as they avoid unnecessary iterations over zero elements.

The use of graphs for point cloud analysis was initiated by Point-GNN, which successfully employed a Graph Neural Network to process point clouds for object detection [15]. However, Point-GNN's limitation lies in its inability to capture the global context of the environment, especially in large settings. Subsequently, graph attention mechanisms were developed for object detection from raw LiDAR data [18]. Recent research has utilized Graph Attention Networks for 3D object detection [11]. These methods have inspired the use of voxels and attention in graph classification. A simple and highly accurate network is needed for contaminant detection. To the best of our knowledge, no study has leveraged a graph-based approach for contaminant classification research.

3 METHODOLOGY

This paper proposes point cloud contaminant detection as a graph classification problem. Due to the lack of a dedicated dataset for this task, we collected the LiDAR data ourselves, which includes varying levels of location, distance, sparsity, and various types and levels of contaminations (for details, refer to subsection 3.1). Recognizing the sparsity of the data in our dataset, we decided to use graph attention networks (GATs), which utilize the attention mechanism to handle data sparsity effectively. A LiDAR point cloud comprises up to 2 million points; if a single node represents a single point, it causes high computational costs. To address this challenge, we propose voxelization, which involves dividing the 3D space of point clouds occupied by some points into small cubic volumes called voxels so that a single node will represent several points. This reduction in the number of nodes significantly reduces computational time. We then create the graph structure from the voxelized data (detailed in subsection 3.2) to be passed into the GAT network (described in subsection 3.3). The network outputs a single global representation of each input voxel graph as a binary classifier, aiming to detect contaminant presence or absence.

3.1 Cover Contamination LiDAR Dataset

The point cloud data were captured using a test-bed car equipped with a LiDAR sensor, specifically the RS-Ruby from Robosense,

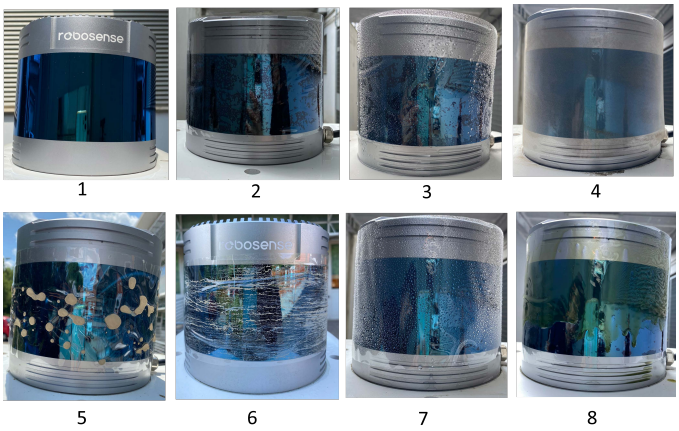

Figure 1: Applying contaminant on LiDAR cover: (1) clean,(2) clean-cover, (3) tap water, (4) dust, (5) mud drop, (6) mud uniform, (7) salt water, (8) lubricant oil.

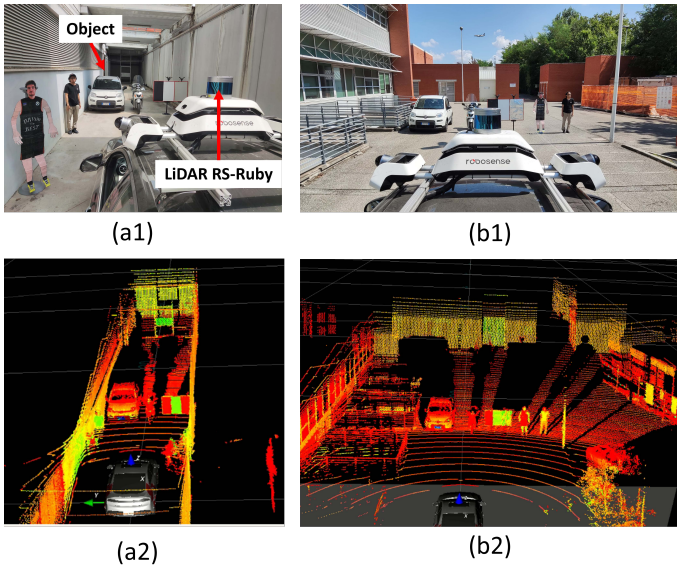

Figure 2: Contaminated LiDAR data acquisition using testbed-car, contaminated LiDAR, and surrounding object: (a1) data $5m$ in an underground narrow hallway, (a2) raw LiDAR acquired in $5m$, (b1) Data $10m$ data outdoor parking area, (a2) raw LiDAR acquired in $10m$.

rated at level L4+ or considered High Driving Automation. The LiDAR sensor has specifications of 128 signal beams, maximum 250 meters range, Horizontal resolution 0.2^o, minimum 0.1^o Vertical Angular Resolution, Horizontal FoV 360^o, Vertical FoV 40^o, 10 Hz frame rate.

This contaminant detection model differentiates between two classes: clean and contaminated. The clean class includes two conditions: (1) the sensor is free from contaminants, and (2) the sensor is covered with a clean-transparent plastic cover. The contaminated class covers various types of contamination, including water, dry and wet mud, dust, salt water, and engine lubricant/oil on top of clean-transparent plastic cover. The application of contaminants is shown in Figure 1. Each type of contaminant is applied separately in different experiments. Each contamination has low, middle, and high levels, corresponding to varying amounts of spray used when applying the contaminant to the sensor cover. For example, 'low' corresponds to one spray application, 'mid' to three, and 'high' to five. This approach aims to reflect the diverse characteristics of real-world contaminants. To simulate objects encountered on the road, we covered several objects, such as cars, pedestrians, motorbikes, whiteboards, and aluminum foil.

To evaluate high generalization capabilities with high accuracy and robustness, this research utilizes three distinct datasets representing varied environmental settings: the $5m$, $10m$, and $20m$ datasets. The $5m$ collected dataset is of a passage to an underground parking lot, where the hallway is below ground level and is confined by walls on both sides. In this dataset, the LiDAR is positioned with the car object around 5 meters ahead, surrounded by other objects. The $10m$ and $20m$ datasets have data collected in the exact location. The location is an outdoor parking lot with wider spatial dimensions compared to the $5m$ data. In the $10m$ dataset, the target object is positioned around 10 meters from the LiDAR, while in the $20m$ dataset, the target object is placed at around 20 meters. Figure 2 depicts our data acquisition setup.

We select a specific area in front of the LiDAR sensor to ensure the effect of contamination. This area encompasses all three dimensions for the training and testing datasets, with the LiDAR-equipped car serving as the reference coordinate (Point 0,0,0). We take the area of interest where target objects are placed in our environment. So, along the x-axis, we choose a span of 80 meters from the reference point (0 to 80 meters). For the y-axis, we select an area of 15.5 meters (-5.5 to 10 meters in coordinate); for the z-axis, we take all the return points. The same area of interest cropping procedure is applied to all three datasets. The $5m$ dataset typically yields around 140,000 points per point cloud, in contrast to the $10m$ and $20m$ datasets, which generally yield around 70,000 points per point cloud. The higher point density in the $5m$ dataset is due to the closer distance of the LiDAR to the target object, resulting in denser point clouds.

3.2 Graph Construction

To construct graphs, we first apply voxelization to reduce the number of points to avoid computational overhead. Each point cloud is converted into a three-dimensional voxel grid. The number of voxels differs for each point cloud depending on the original structure or environment in real-world situations.

In voxelization, we employ the concept of average pooling to maintain the primary feature representation of the point cloud, which, in the case of LiDAR, is the point cloud structure and intensity of each point. Each voxel computes and retains the average intensity of the points it encapsulates, thus maintaining spatial information of the original point cloud. Applying a 3D convolutional layer directly on 3D voxels might make our predictions less accurate because it processes every voxel, even the ones without any points. To address this issue, we propose converting voxels into graphs. In this graph representation, each node corresponds to

a voxel containing points and the edges to connect voxels closest to each other in three dimensions. This approach aims to enhance prediction accuracy by focusing exclusively on voxels containing relevant information and their immediate spatial relationships. The edges for each graph are created in a three-step process, as follows:

(1) Point sorting: The initial step is the most important of the whole process. We employ a QuickSort algorithm to sort the points according to their coordinates in the three dimensions X, Y, Z separately. For n, the number of points along a dimension, the algorithm will have a $O(n \log n)$ complexity. Since we have it in three dimensions (z, y, z), the overall complexity is $O(n \log n)$.
(2) Edge construction: This process considers adjacent points from the sorted lists in each dimension. Each point in every dimension can have at most two adjacent points, except for the first and last elements on the list. This results in approximately $2n$ edges for each dimension, totaling $6n$ across all three dimensions. This process has a $O(n)$ complexity. The list of edges is then saved in an edge index.
(3) Edge directionality: Contaminants primarily influence the intensity of the reflected LiDAR beams and the structure of objects rather than introducing directional effects. Therefore, in our graph construction, edges play a crucial role in capturing point distribution for each node according to its neighbor but do not necessitate bi-directionality. Therefore, the constructed graphs have un-directed edges. We utilize the coalesce function from the pytorch-geometric library to remove duplicate edges. In principle, this process would require $O(k \log k)$ where k is the total number of edges before removing duplicates. In the worst case, k can reach $6k$ for all three dimensions, so removing and looping a list of edges can require $O(k)$ in the worst case. However, since k depends on n, this complexity can also be considered as part of $O(n \log n)$ because our proposed construction graph is dominated by sorting operations.

We create a *Data* pytorch-geometric library object representing a graph object. We select all three coordinates and the beam's intensity as a feature vector for each instance of a voxel. The edges we created earlier become the *edge_index* property of the *Data* object, which forms the adjacency matrix of nodes. The proposed method will construct a graph with a different number of nodes and edges according to the original input point cloud structure.

3.3 Graph Attention Networks

We construct a deep learning model that employs Graph Attention Networks (GATs). In LiDAR data acquisition, areas with objects often yield dense point clouds, resulting in sparsity within the well-suited data for GATs. The GATs efficiently handle sparse data by representing it as a graph, with each point as a node. Leveraging attention mechanisms, they focus on relevant subsets of nodes, adapting to the local structure of the graph. These features empower GATs to excel in classifying contamination within point clouds generated by LiDAR, making them a powerful tool for our task. Figure 3 outlines the proposed GATs model for contaminant classification, followed by table 1, which details the number of parameters in the proposed model. The proposed architecture comes from empirical experiments to get the minimum parameters with the highest F1 score.

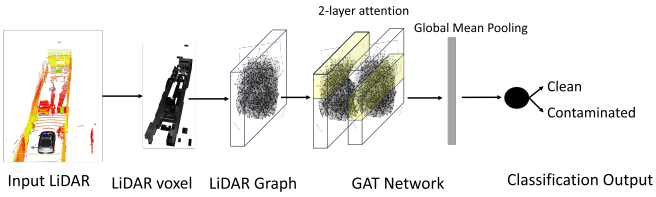

Figure 3: The Proposed LiDAR Contaminant Detection based on Graph Attention Networks.

Table 1: The Proposed network for LiDAR Contaminant Detection.

conv1.att_src: 64
conv1.att_dst: 64
conv1.bias: 64
conv1.lin_src.weight: 256
conv2.att_src: 2
conv2.att_dst: 2
conv2.bias: 2
conv2.lin_src.weight: 128
Total Parameters: 582

The model uses two GAT layers (GATConv) to process node features in the graph, generating an adaptive representation of each node based on its local context. The first layer employs multi-head attention mechanisms to enrich and diversify feature representations with four heads. Each node's features pass through this first GAT layer (conv1), followed by an ELU activation for non-linearity. The second layer (conv2) reduces the output feature dimension to one for each node. GAT's attention mechanism assigns varying weights to node neighbors, enhancing relationship capture. Since our task is graph classification, we then pass the graph through a (global mean pooling) aggregation function to get a representation of all the node features of the graph into a single value for binary classification: clean or contaminated. The model's output, the Log_softmax applied to this value, yields a probability distribution for clean and contaminated classes. Training employs the Adam optimizer with $lr = 0.005$ and $weight\_decay = 5e-4$.

4 RESULTS

4.1 Setting Experiments

4.1.1 Dataset Preparation. The numbers of contaminated class instances are 360, 328, and 364 for the $5m$, $10m$, and $20m$ datasets, respectively. The contaminated class combines various contamination sources explained in 3.1. While 360 instances represent the clean class across all datasets, we consider the data balanced. Each point cloud instance contains between around 70,000 and 140,000 points. Our proposed method transforms each point cloud instance into a graph, resulting in node-edge pairs: (n)1361-(e)3879 for the $5m$ dataset, (n)2209-(e)6359 for the $10m$ dataset, and (n)2514-(e)7242 for the $20m$ dataset. Training and testing are conducted using: CPU

AMD EPYC 74F3 24 core 48 threads, single core CPU clock speed 3.2G, single GPU NVIDIA GeForce RTX 3090 with 24265MiB, and RAM 250G DDR4.

To demonstrate the robustness of the contaminant detection method, we will conduct a cross-environment between training and testing data. Thus, three models will be developed, each trained on data from an environment, for example, 5m, and then tested on data from the other environments, 10m and 20m, and vice versa. Utilizing three different datasets captures the dynamic aspects of the point cloud that are close to real-world conditions. This approach will also highlight the model's generalization and transferability capabilities. Therefore, even with limited training data, the detection model can perform effectively across different environmental settings.

4.1.2 Baseline Method. To demonstrate the complexity of contaminant detection in dynamic situations, we will compare classification models ranging from manual feature engineering to the most straightforward transformations in 1D and 3D. The point cloud is transformed into 1D data by extracting information such as x, y, z coordinates, and intensity values, resulting in seven features: minimum, maximum, mean, standard deviation, and percentiles at 25, 50, and 75. This process generates 28 1D features that represent the statistical distribution of the point cloud. Subsequently, we compare various machine learning algorithms, including Logistic Regression, Multilayer Perceptron, Support Vector Machine, Naive Bayes, k-Nearest Neighbors, Decision Tree, and Random Forest. Additionally, we compare these methods to another baseline, namely the 3D approach, which preserves the spatial information from the point clouds. In the 3D approach, voxelization is performed, and the model is constructed using a 3D Convolutional Neural Network (3D CNN).

4.2 Contaminant Detection

The performance of contaminant detection is analyzed in terms of F1-score, graph construction time, and inference time across all possible scenarios. Table 2 evaluates the performance of various machine learning algorithms in detecting contaminants, measured by F1 scores, at three different data scales (5m, 10m, and 20m). The models compared include well-established classification techniques such as Decision Tree, Logistic Regression, Naive Bayes, and Support Vector Machine, as well as more modern approaches like Multi-Layer Perceptron, K-Nearest Neighbors, 3D Convolutional Neural Network, and the proposed method, namely Voxel-GAT.

From the data presented in Table 2, we observe that shallow learning models, such as Decision Trees (DT), Logistic Regression (Logit), and Naive Bayes (NB), are not able to perform adequately in contaminant detection tasks, with their average F1 score being below 80% across all tested data scales. This suggests that shallow learning methods may lack the capacity to capture and model complex relationships in three-dimensional spatial data, often involving nonlinear interactions and patterns that are difficult to separate. This limitation is likely because these models do not exploit the spatial structure in the data and tend to have more superficial feature representations, resulting in lower performance in tasks that require a deep contextual understanding. On the other hand, models such as 3D CNNs show improved performance compared to traditional models, supporting the theory that three-dimensional spatial data structures are inherently better suited to tasks with spatial characteristics.

The proposed method, Voxel-GAT, stands out in this comparison by demonstrating high F1 scores across all data scales, indicating its effectiveness in detecting contaminants with high consistency. This suggests that Voxel-GAT can capture complex spatial relationships in the data, an essential aspect of tasks that detect patterns or anomalies in three-dimensional spatial data. Voxel-GAT excels in ensuring distance-environment invariance, which is critical for explaining differences between the trained and tested data. As previously described, the 10m data set is similar to the 20m data set, whereas the 5m data set differs regarding the location and distance of surrounding objects. The proposed model is more robust across different scenarios, being trained on 5m and tested on 10m and 20m data, and vice versa. For the cross-train-test dataset, we have six possible scenarios, which, scenarios 1,2,3,4,5 and 6. All of the scenarios introduced are in table 2. Based on the table, scenarios 3 and 5 are trained on 10m or 20m and still performing well on the 5m test data—a result not achieved by the baseline methods. The consistently superior performance of Voxel-GAT across all data scales confirms its adaptability and robustness in handling variations in data size and complexity, making it a promising choice for real-world contaminant detection applications.

The proposed method demonstrates outstanding performance, as reflected in the confusion matrix shown in Figure 4. We examine six Voxel-GAT confusion matrices for each scenario, corresponding to the scenario numbers in Table 2. Based on the confusion matrix displayed in Figure 4, it is evident that the Voxel-GAT model generally experiences a low number of False Negatives, with no instances of False Positives, except in Scenario 4. In this scenario, where the model was trained with 10m data and tested with 20m data, several cases of False Positives were observed. A closer inspection of the case data for each misclassification, detailed in Table 3, reveals that errors occurred with data originating from various sources of contaminants, namely water, uniform mud, mud drop, dust, and salt. Interestingly, misclassifications predominantly happen in cases of low contamination, where the contamination level is the lowest. An exception is observed with mud drops, where misclassifications occur at low, mid, and high contamination levels. This phenomenon is understandable, given that mud drops cover the sensor at specific locations with a volume of contaminant that is insignificant compared to the total sensor coverage.

The false positives identified in Scenarios 4 and 6 were attributed to the clean cover, which was mistakenly classified as contamination. This issue is not particularly problematic due to the nature of the cover installation. Contaminants were applied to the protective cover surrounding the LiDAR to safeguard the experimental (and expensive) LiDAR hardware. This cover introduces noise into the LiDAR image and sometimes acts as an anomaly/contaminant. Notably, the contaminant classification approach consistently recognized clean LiDAR data (without the cover) as non-contaminated. The instances involving the cover are the primary cause of false positives in Scenarios 4 and 6.

However, the problem with the cover data does not invalidate the robustness of the proposed contaminant detection model. If we focus only on the critical anomalies of middle and high contamination, we still observe that we correctly classify most cases. As only

Table 2: Evaluation result of all cross train and test data scenario for contaminant detection from baseline and the proposed method Voxel-GAT. The number reported is F1-score.

No. Scenario	Model	Testing Data	1D DT	1D Logit	1D MLP	1D NB	1D RF	1D SVMrb	1D KNN	3D CNN	Voxel-GAT
1	$5m$	$10m$	0.308	0.308	0.308	0.308	0.308	0.711	0.308	0.780	0.918
2	$5m$	$20m$	0.336	0.336	0.336	0.336	0.336	0.728	0.336	0.777	0.954
	average F1		**0.322**	**0.322**	**0.322**	**0.322**	**0.322**	**0.719**	**0.322**	**0.778**	**0.936**
3	$10m$	$5m$	0.676	0.512	0.336	0.333	0.526	0.486	0.720	0.733	0.889
4	$10m$	$20m$	0.343	0.349	0.825	0.336	0.971	0.936	0.900	0.968	0.915
	average F1		**0.509**	**0.430**	**0.580**	**0.334**	**0.748**	**0.711**	**0.810**	**0.850**	**0.902**
5	$20m$	$5m$	0.361	0.333	0.336	0.333	0.361	0.381	0.568	0.733	0.876
6	$20m$	$10m$	0.871	0.833	0.955	0.308	0.977	0.940	0.977	0.948	0.964
	average F1		**0.616**	**0.583**	**0.645**	**0.320**	**0.669**	**0.660**	**0.772**	**0.840**	**0.920**

these examples cause real problems in both autonomous driving applications and perception systems, they should be the focus of the proposed work. In practical applications in production-ready hardware, there would also be no problem with the cover and contamination as the LiDAR will operate without any additional plastic cover.

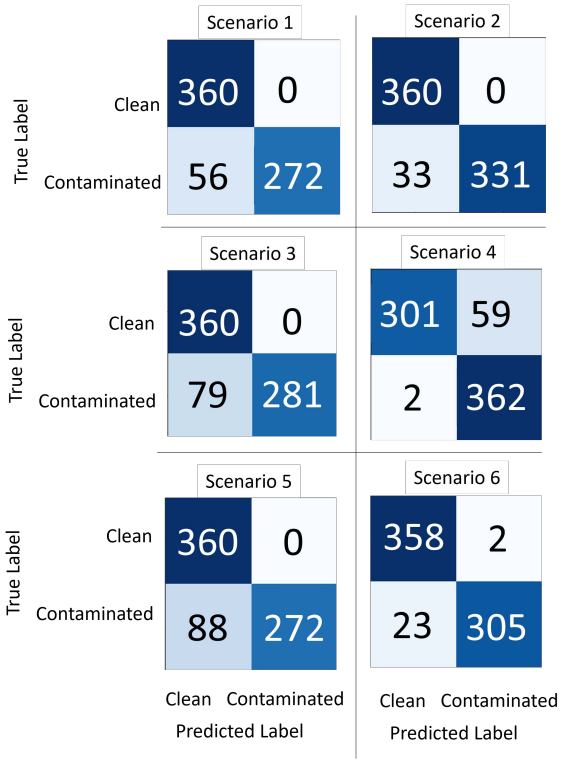

Figure 4: Confusion matrix of the proposed method in all scenarios.

Regarding computational performance, Table 4 compares the inference time of the baseline methods and the proposed method when running on both CPU and GPU. The 'CPU' column displays the inference time on the CPU, while the 'GPU' column shows the inference time when the process is executed on the GPU. Although the shallow machine learning methods are generally faster, their F1-score performance is significantly lower than Voxel-GAT.

Table 3: Mis-classification cases of the proposed method Voxel-GAT.

No. Scenario	False Positive	False Negative
1	-	water:low; mudUniform:low,mid; dust:low; salt:low
2	-	water:low; mudUniform:low
3	-	water:low; mudDrop:low,mid,high; dust:low; salt:low
4	cover	water:low
5	-	water:low; mudDrop:low,mid,high; dust:low; salt:low
6	cover	water:low; mudUniform:low; salt:low

Table 4: Inference time of baseline method and proposed Voxel-GAT, running on CPU and GPU.

Method	CPU (in second)	GPU (in second)
DT	5.44E-07	x
Logit	1.20E-06	x
MLP	6.82E-06	x
NB	5.46E-07	x
RF	1.62E-06	x
SVMrb	1.44E-05	x
KNN	1.46E-04	x
3D CNN	6.10E-04	4.03E-05
Voxel-GAT (Proposed)	4.99E-03	7.91E-05

For the proposed method, the inference time on the CPU is recorded at 4.99e-03 seconds. On the GPU, the inference time reduces to 7.91e-05 seconds, indicating that the method benefits significantly from the parallel processing capabilities of the GPU. In comparison, the baseline 3D CNN has an inference time of 6.10e-04 seconds on the CPU and 4.03e-05 seconds on the GPU. This means that 3D CNN is about 1.96 times faster on the GPU than Voxel-GAT.

An early warning system should not only account for inference time but also include preprocessing time. The time required for preprocessing is 0.010 seconds, 0.129 seconds, and 0.128 seconds for 1D, 3D, and graph data, respectively. While 1D processing is faster, the time taken for voxel and graph creation is not significantly

longer, on average. When combining preprocessing with inference time, the total end-to-end time from raw data to detection result does not significantly increase. The time to create a graph is approximately 0.128 seconds or 128 milliseconds. However, considering the trade-off between the F1 score and inference time, Voxel-GAT is the preferable choice, offering a much higher F1 score than 3D CNN with a competitive end-to-end computational process.

5 CONCLUSIONS

The LiDAR contaminant detection method for autonomous vehicles has been developed. We compared all transformation methods, from 1D and 3D, to a developed graph-based transformation. The proposed method is superior in all experimental scenarios, including cross-environments and contamination, on real-world LiDAR sensors installed on test-bed vehicles.

The proposed method achieves an average F1-score above 0.902, with the graph processing time from creation to detection under around 128 milliseconds. We utilize a compact architecture containing two Graph Attention Networks layers capable of processing point cloud inputs of dynamic sizes. Ultimately, the proposed model has been proven to effectively classify cleanliness or contamination in complex and extreme LiDAR point clouds. This initiates further research into designing near-sensor LiDAR processing methods that are low-cost and sustainable using a graph processing approach.

ACKNOWLEDGMENTS

This research is supported by EU through the National Recovery and Resilience Plan (NRRP) Mission 4, Component 2, Investment 3.3 (g.a. DM 352/2022). We also thank FEV Italia s.r.l. for the collaboration, access to their hardware, and help in experimental work. This research was supported in part by the UAE Technology Innovation Institute (TII) through the Zero-Trust project. This research was also partly supported by the Graph-Massivizer project (g.a. 101093202). We also thank to Giammarco Cecchini for helping during data acquisition.

REFERENCES

[1] Csaba Benedek, Andras Majdik, Balazs Nagy, Zoltan Rozsa, and Tamas Sziranyi. 2021. Positioning and perception in LIDAR point clouds. *Digital Signal Processing* 119 (2021), 103193. https://doi.org/10.1016/j.dsp.2021.103193

[2] Daniel Bogdoll et al. 2022. Anomaly detection in autonomous driving: A survey. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4488–4499.

[3] Krzysztof Czarnecki and Rick Salay. 2018. Towards a Framework to Manage Perceptual Uncertainty for Safe Automated Driving. In *Computer Safety, Reliability, and Security*, Barbara Gallina, Amund Skavhaug, Erwin Schoitsch, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 439–445.

[4] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. 2021. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 1201–1209.

[5] Yinpeng Dong, Caixin Kang, Jinlai Zhang, Zijian Zhu, Yikai Wang, Xiao Yang, Hang Su, Xingxing Wei, and Jun Zhu. 2023. Benchmarking Robustness of 3D Object Detection to Common Corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1022–1032.

[6] Robin Heinzler et al. 2019. Weather Influence and Classification with Automotive Lidar Sensors. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. 1527–1534.

[7] Infineon Technologies AG. 2024. From ADAS to Autonomous Driving. https://www.infineon.com/cms/en/discoveries/adas-to-ad/. Accessed: 2024-02-10.

[8] Jyothish K James et al. 2018. Classification of lidar sensor contaminations with deep neural networks. In *Proceedings of the Computer Science in Cars Symposium (CSCS)*. 8.

[9] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. 2020. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision* 12, 1–3 (2020), 1–308.

[10] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. 2018. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. 751–766.

[11] Bin Lu, Yang Sun, and Zhenyu Yang. 2023. Voxel Graph Attention for 3-D Object Detection From Point Clouds. *IEEE Transactions on Instrumentation and Measurement* 72 (2023), 1–12. https://doi.org/10.1109/TIM.2023.3301907

[12] Nguyen Anh Minh Mai et al. 2022. Camera and LiDAR analysis for 3D object detection in foggy weather conditions. In *2022 12th International Conference on Pattern Recognition Systems (ICPRS)*. 1–7.

[13] MIT News. 2019. How fast can humans react to car hazards. https://news.mit.edu/2019/how-fast-humans-react-car-hazards-0807. Accessed: 2024-02-10.

[14] Birgit Schlager et al. 2022. Contaminations on Lidar Sensor Covers: Performance Degradation Including Fault Detection and Modeling as Potential Applications. *IEEE Open Journal of Intelligent Transportation Systems* 3 (2022), 738–747. https://doi.org/10.1109/ojits.2022.3214094

[15] Weijing Shi and Ragunathan (Raj) Rajkumar. 2020. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] Li Tang et al. 2020. Performance Test of Autonomous Vehicle Lidar Sensors Under Different Weather Conditions. *Transportation Research Record* 2674, 1 (2020), 319–329. https://doi.org/10.1177/0361198120901681

[17] Xiaolin Tang, Kai Yang, Hong Wang, Jiahang Wu, Yechen Qin, Wenhao Yu, and Dongpu Cao. 2022. Prediction-Uncertainty-Aware Decision-Making for Autonomous Vehicles. *IEEE Transactions on Intelligent Vehicles* 7, 4 (2022), 849–862. https://doi.org/10.1109/TIV.2022.3188662

[18] Sumesh Thakur, Bivash Pandey, Jiju Peethambaran, and Dong Chen. 2022. A Graph Attention Network for Object Detection from Raw LiDAR Data. In *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. 3091–3094. https://doi.org/10.1109/IGARSS46834.2022.9883734

[19] Jose Roberto Vargas Rivero et al. 2020. Weather Classification Using an Automotive LIDAR Sensor Based on Detections on Asphalt and Atmosphere. *Sensors* 20, 15 (2020). https://doi.org/10.3390/s20154306

[20] Hong Wang, Wenbo Shao, Chen Sun, Kai Yang, Dongpu Cao, and Jun Li. 2024. A Survey on an Emerging Safety Challenge for Autonomous Vehicles: Safety of the Intended Functionality. *Engineering* (2024). https://doi.org/10.1016/j.eng.2023.10.011

[21] Oliver Willers, Sebastian Sudholt, Shervin Raafatnia, and Stephanie Abrecht. 2020. Safety Concerns and Mitigation Approaches Regarding the Use of Deep Learning in Safety-Critical Perception Tasks. In *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops*, António Casimiro, Frank Ortmeier, Erwin Schoitsch, Friedemann Bitsch, and Pedro Ferreira (Eds.). Springer International Publishing, Cham, 336–350.

[22] Akihiro Yamaguchi, Yousuke Watanabe, Kenya Sato, Yukikazu Nakamoto, Yoshiharu Ishikawa, Shinya Honda, and Hiroaki Takada. 2017. In-vehicle distributed time-critical data stream management system for advanced driver assistance. *Journal of Information Processing* 25 (2017), 107–120.

[23] Kai Yang, Xiaolin Tang, Sen Qiu, Shufeng Jin, Zichun Wei, and Hong Wang. 2023. Towards Robust Decision-Making for Autonomous Driving on Highway. *IEEE Transactions on Vehicular Technology* 72, 9 (2023), 11251–11263. https://doi.org/10.1109/TVT.2023.3268500

[24] Yuxiao Zhang, Alexander Carballo, Hanting Yang, and Kazuya Takeda. 2023. Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing* 196 (2023), 146–177. https://doi.org/10.1016/j.isprsjprs.2022.12.021

[25] Xingyu Zhao, Kizito Salako, Lorenzo Strigini, Valentin Robu, and David Flynn. 2020. Assessing safety-critical systems from operational testing: A study on autonomous vehicles. *Information and Software Technology* 128 (2020), 106393. https://doi.org/10.1016/j.infsof.2020.106393