Hybrid 3D Graph Convolutional Network and Transformer for Lidar Fault Detection

Jae Ho Seong
AI&ICT Convergence Team
Korea Intelligent Automotive Parts Promotion Institute (KIAPI)
Daegu, Republic of Korea
wogh3569@kiapi.or.kr

Jun Yeong Kim
AI&ICT Convergence Team
Korea Intelligent Automotive Parts Promotion Institute (KIAPI)
Daegu, Republic of Korea
kwnsdud@kiapi.or.kr

Sun Ho Lee
AI&ICT Convergence Team
Korea Intelligent Automotive Parts Promotion Institute (KIAPI)
Daegu, Republic of Korea
2prefer@kiapi.or.kr

Tae Hyeong Kim
AI&ICT Convergence Team
Korea Intelligent Automotive Parts Promotion Institute (KIAPI)
Daegu, Republic of Korea
thkim@kiapi.or.kr

Bong Seob Kim
Research & Development Department
Korea Intelligent Automotive Parts Promotion Institute (KIAPI)
Daegu, Republic of Korea
bskim@kiapi.or.kr

Kyung Su Yun
Strategic Planning Division
Korea Intelligent Automotive Parts Promotion Institute (KIAPI)
Daegu, Republic of Korea
kadbonow@kiapi.or.kr

***Abstract*—Lidar sensors are essential perception components for autonomous vehicles, providing high-resolution 3D point clouds for tasks such as object detection and localization. Sensor faults, including channel failures, partial or full occlusions, and environmental noise, can degrade point-cloud quality and compromise driving safety. Detecting such faults is challenging due to the difficulty and rarity of safely collecting real-world fault data. To address this, we propose a pipeline that generates fault point clouds by systematically injecting diverse lidar fault patterns into normal driving data. We further introduce a hybrid neural network that integrates a 3D Graph Convolutional Network (3D-GCN) for local geometric feature extraction with a Transformer to capture long-range contextual dependencies. A stepwise verification procedure progressively analyzes point-cloud characteristics to enhance detection reliability. Experimental evaluation on synthetic test sets demonstrates that the proposed method achieves 96.25% classification accuracy with only 1.08 million parameters, outperforming existing approaches while supporting real-time inference. This framework provides a practical and effective solution for early lidar fault detection, facilitating robust perception in autonomous driving.**

***Index Terms*—Autonomous driving, Lidar fault detection, Synthetic fault data, 3D point clouds, 3D-GCN, Transformer**

I. Introduction

Lidar sensors provide high-resolution 3D point clouds that are essential for autonomous vehicle perception tasks, including object detection, localization, and mapping. However, lidars are susceptible to faults such as channel failures, occlusions, and surface contamination, which can degrade point cloud quality and compromise downstream perception modules [8]–[10]. Rapid and reliable fault detection is therefore critical for autonomous driving safety. Existing detection approaches can be broadly categorized into three types: (i) hardware- and signal-based methods leveraging internal sensor measurements [13], (ii) multi-sensor consistency checks using radar data [17], and (iii) data-driven point cloud methods, including variational autoencoder reconstruction and graph-based models [7], [16], [17]. Nevertheless, these approaches are constrained by the scarcity of labeled fault data, making real-world data collection both costly and hazardous. Synthetic fault generation provides a practical alternative [15], enabling efficient dataset augmentation. Consequently, effective models should capture both local geometric distortions and global contextual patterns to distinguish transient noise from persistent faults.

We propose a practical framework for lidar fault detection that integrates synthetic fault generation and a hybrid deep architecture, as shown in Fig. 1. A deformable 3D-GCN extracts direction-aware local features, while the Transformer module captures long-range contextual dependencies. The main contributions of this work are summarized as follows:

1) A synthetic fault generation pipeline that creates multiple realistic lidar fault modes for data-efficient training.
2) A hybrid detection network combining the 3D-GCN and Transformer module to effectively model both local and global features.
3) A multi-stage, confidence-driven verification scheme that reduces false positives during streaming inference.
4) Experimental validation on synthetic datasets demonstrating improved fault detection performance compared to existing methods.

The remainder of the paper is organized as follows: Section II reviews related work. Section III details the synthetic fault generation process and the hybrid network. Section IV presents experimental results. Section V concludes the study.

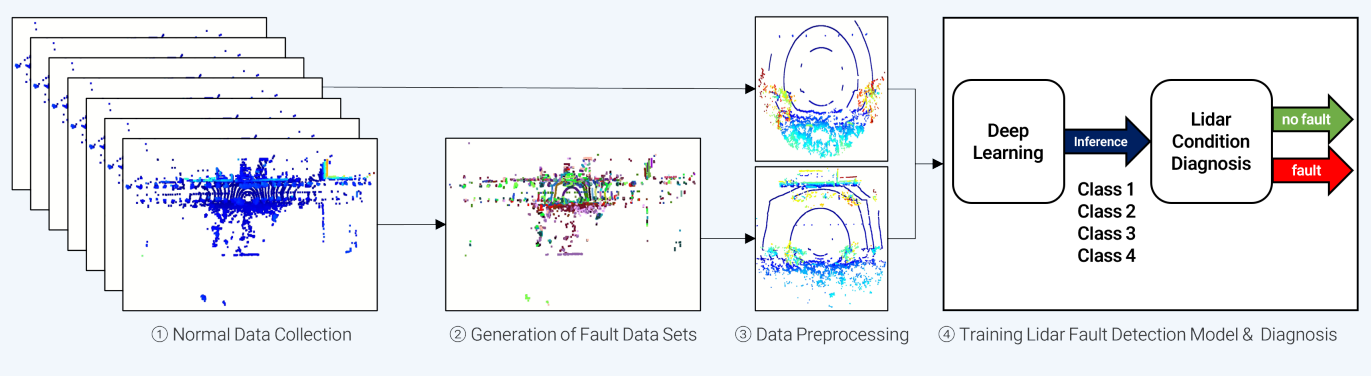


Fig. 1. Lidar fault diagnosis framework: hybrid 3D-GCN and Transformer-based fault detection for lidar condition assessment.

II. Related Work

A. Lidar Faults and Synthetic Data

Lidar faults, including channel failures, occlusions, and hardware degradation, can significantly degrade autonomous vehicle perception. Detection approaches include hardware- and signal-based diagnostics [13], multi-sensor fusion [17], and data-driven point cloud methods [7], [16], [17]. Hardware-based methods are typically sensor-specific and difficult to generalize. Multi-sensor approaches can be sensitive to calibration errors and environmental conditions. Data-driven methods offer scalability and flexibility but are constrained by the scarcity of labeled fault data, which is costly to collect. To overcome this limitation, synthetic fault data—including occlusion masks, Gaussian noise, and partial point dropout—can be generated to augment training datasets and create controlled evaluation scenarios [8]–[10], [14], [15]. Carefully designing fault types and systematic pipelines enables these synthetic datasets to approximate real-world conditions closely, supporting robust deployment of models in autonomous vehicle systems.

B. Point Cloud Learning

Although early point cloud methods applied voxelization or bird's-eye projections to enable 2D CNN processing, this often resulted in a loss of fine geometric details [3]. PointNet and its variants process raw point sets directly, preserving local structures and capturing neighborhood information [1], [2], while PointNext improves training efficiency and scalability for larger point clouds [12]. Graph-based representations treat points as vertices connected by edges, capturing local connectivity and structural relationships. Graph convolutional networks with deformable kernels extract expressive local features that adapt to varying point densities, whereas Transformer-based models effectively capture global context and long-range dependencies [4], [5], [11]. Inspired by the Conformer model for image classification [6], we propose a hybrid 3D-GCN + Transformer architecture that jointly models local and global features, improving robustness to sparse or partially missing points for enhanced lidar fault detection.

III. Our Method

A. Fault Data Generation

Due to the limited availability of real lidar fault data, synthetic fault data are generated from normal point clouds. A normal point cloud is defined as:

$$\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3 \mid i = 1, \ldots, N\}, \tag{1}$$

where N denotes the number of points, and each point $\mathbf{p}_i$ may include additional attributes such as intensity or return index. Points outside a predefined spatial region $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$ are first removed to ensure data quality before fault generation.

1) Channel Failure: Partial loss of lidar channels can occur due to LED failures or physical damage. To simulate channel failures, the point cloud is divided along the z-axis into dense and sparse regions:

$$\mathcal{P}_{\mathrm{K}} = \{\mathbf{p}_i \mid z_{\min}^{K} \le z_i \le z_{\max}^{K}\}, \tag{2}$$

$$\mathcal{P}_{\mathrm{DB}} = \{\mathbf{p}_i \mid z_i < z_{\min}^{K} \text{ or } z_i > z_{\max}^{K}\}. \tag{3}$$

Here, $\mathcal{P}_{\mathrm{K}}$ is the dense middle region, and $\mathcal{P}_{\mathrm{DB}}$ contains the sparse lower and upper regions.

In the dense region, K-Means clustering is applied to the z-coordinates:

$$\{\hat{C}_j^{\mathrm{K}}\}_{j=1}^{K_{\text{clusters}}} = \mathrm{KMeans}(\{z_i\}_{\mathbf{p}_i \in \mathcal{P}_{\mathrm{K}}}), \quad K_{\text{clusters}} = 50, \tag{4}$$

while in the sparse regions, DBSCAN is applied to all three coordinates:

$$\{\hat{C}_j^{\mathrm{DB}}\} = \mathrm{DBSCAN}_{\epsilon=0.5,\mathrm{minPts}=5}(\{\mathbf{p}_i\}_{\mathbf{p}_i \in \mathcal{P}_{\mathrm{DB}}}). \tag{5}$$

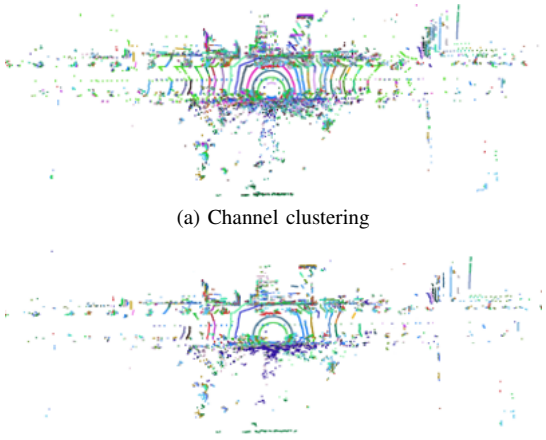
Clusters from both regions are combined:

$$\mathcal{C} = \{\hat{C}_j^{\mathrm{K}}\} \cup \{\hat{C}_j^{\mathrm{DB}}\}. \tag{6}$$

A subset of clusters, $\mathcal{C}_{\text{drop}}$, is randomly sampled according to the drop fraction p_{drop}. A drop fraction $p_{\text{drop}} \in [0.3, 0.8]$ is randomly selected to simulate moderate to severe channel failures. Each point $\mathbf{p}_i$ is then updated as:

$$\mathbf{p}_i' = \begin{cases} \mathbf{p}_i, & \mathbf{p}_i \in C_j \text{ and } C_j \notin \mathcal{C}_{\text{drop}}, \\ \emptyset, & \text{otherwise.} \end{cases} \tag{7}$$

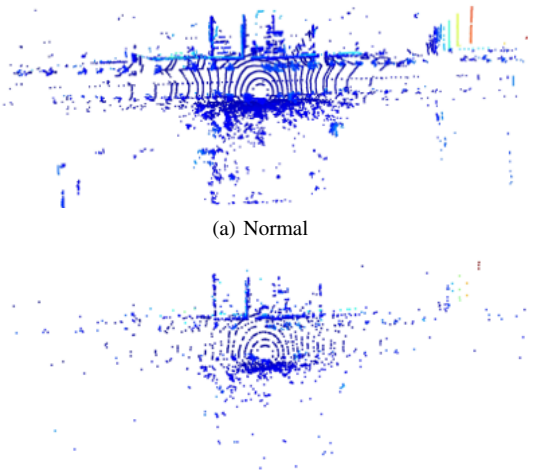
This procedure preserves local density characteristics while realistically simulating channel failure.



(a) Channel clustering

(b) Channel filtering

Fig. 2. Generation of channel failures with clustering of points and removal of selected clusters.



(a) Normal

(b) Global occlusion

Fig. 3. Global occlusion generated by randomly removing a fraction of points from the normal point cloud.

2) Global Occlusion: Global occlusion occurs when a fraction of points across the full field of view of the lidar is lost, e.g., due to dirt, dust, dew, or snow. A removal proportion $p_{\text{remove}} \in [0.3, 0.5]$ is chosen to simulate moderate occlusion:

$$\mathbf{p}_i' = \begin{cases} \mathbf{p}_i, & \text{with probability } 1 - p_{\text{remove}}, \\ \emptyset, & \text{with probability } p_{\text{remove}}. \end{cases} \tag{8}$$

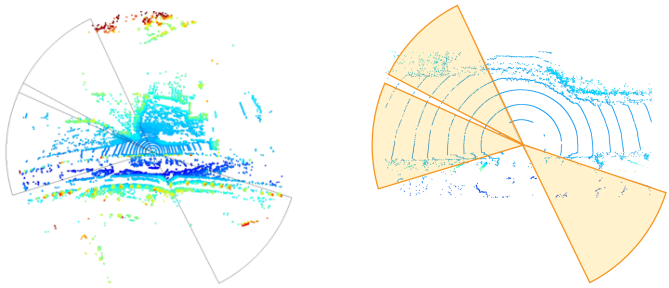
3) Partial Occlusion: Partial occlusion affects only specific angular sectors, e.g., due to dirt, dust, dew, or snow. The point cloud

$$\mathcal{P} = \{\mathbf{p}_i = (x_i, y_i, z_i)\}_{i=1}^{N} \tag{9}$$

is converted to cylindrical coordinates (ρ_i, ϕ_i, z_i):

$$\rho_i = \sqrt{x_i^2 + y_i^2}, \quad \phi_i = \arctan 2(y_i, x_i), \tag{10}$$

where ρ_i is the radial distance and ϕ_i the azimuth angle. Each occlusion sector $k = 1, \ldots, K$ is characterized by a maximum radial distance $\rho_{\text{mask}}^{(k)}$, an azimuthal range $\phi_{\text{range}}^{(k)} \in [25°, 45°]$, a vertical range $z_{\text{range}}^{(k)}$, and a point removal probability $r_{\text{del}}^{(k)} \in [0.5, 1.0]$. For each point cloud, the number of occlusion sectors K is randomly sampled from 1 to 4.



(a) Normal

(b) Partial occlusion

Fig. 4. Partial occlusion by selective point removal in cylindrical coordinate sectors.

Points within each sector are then removed stochastically:

$$\mathbf{p}_i' = \begin{cases} \mathbf{p}_i, & \mathbf{p}_i \notin \text{mask}^{(k)}, \\ \emptyset, & \mathbf{p}_i \in \text{mask}^{(k)} \text{ with probability } r_{\text{del}}^{(k)}. \end{cases} \tag{11}$$

The sector mask is defined as

$$\text{mask}^{(k)} = \{\mathbf{p}_i \mid \rho_i \le \rho_{\text{mask}}^{(k)},\ \phi_i \in \phi_{\text{range}}^{(k)},\ z_i \in z_{\text{range}}^{(k)}\}. \tag{12}$$

This stochastic selection preserves the overall point cloud structure while enabling reproducible partial-occlusion scenarios.

B. Hybrid Architecture

Our approach combines two complementary strategies: (i) 3D-GCN to capture local geometric structures [4], and (ii) Conformer-inspired integration of local feature extractors with global Transformer modules to model long-range dependencies [6]. Given a faulty point cloud $\mathcal{P}'$, preprocessing steps—including range filtering and distance-based sampling—produce the network-ready point cloud:

$$\mathcal{P}'' = \{\mathbf{p}_i'' \in \mathbb{R}^3\}_{i=1}^{N}. \tag{13}$$

These steps reduce tens of thousands to millions of points per second to a manageable input while preserving density variations crucial for fault detection.

1) Range filtering: Points outside a predefined 3D spatial region $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$ are removed to eliminate distant or noisy points. In our implementation, points with x or y outside $[-3, 3]$ m or z outside $[-50, 50]$ m are removed.
2) Distance-based sampling: Remaining points are sorted by Euclidean distance from the sensor origin, and the closest N_{input} points are selected to form a fixed-size input, preserving local density variations for effective fault detection.

The resulting sampled point cloud is:

$$\mathcal{P}_{\text{sampled}} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N_{\text{input}}}, \tag{14}$$

which serves as the input to the hybrid network.

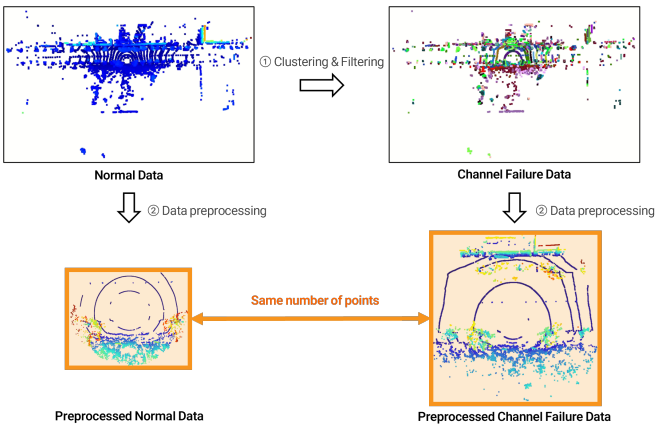


Fig. 5. Comparison of preprocessed lidar point clouds under normal and channel failure scenarios.

1) 3D Graph Construction: A directed k-nearest neighbor (k-NN) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed:

$$\mathcal{N}_i = \arg\operatorname{top}_k \min_j \|\mathbf{p}_i - \mathbf{p}_j\|_2, \tag{15}$$

where each vertex represents a point, and edges connect to its k nearest neighbors.

2) Local Feature Extraction and Initial Transformer: Local features are extracted using deformable-kernel surface convolution:

$$\mathbf{h}_i^{(0)} = \text{LocalFeature}(\mathbf{p}_i). \tag{16}$$

A Transformer encoder captures long-range dependencies:

$$\mathbf{H}^{(0)} = [\mathbf{h}_1^{(0)}, \ldots, \mathbf{h}_N^{(0)}]^\top, \tag{17}$$

$$\mathbf{H}' = \text{GlobalContext}(\mathbf{H}^{(0)}). \tag{18}$$

Local and global features are fused element-wise:

$$\mathbf{h}_i^{(0,\text{fused})} = \mathbf{h}_i^{(0)} \odot \mathbf{h}_i', \tag{19}$$

providing graph convolution layers with features enriched by both local geometry and global context.

3) Graph Convolution Layers: For $l = 1, \ldots, L$:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} w_{ij}^{(l)} \mathbf{h}_j^{(l)}\right), \quad w_{ij}^{(l)} = f_\theta(\mathbf{p}_j - \mathbf{p}_i), \tag{20}$$

where f_θ is a learnable MLP and σ is a non-linear activation.

4) Global Pooling and Classification: Point-wise features are aggregated by global max-pooling:

$$\mathbf{h}_{\text{global}} = \max_{i=1,\ldots,N} \mathbf{h}_i^{(L+1)}. \tag{21}$$

A fully connected layer with softmax predicts the fault class:

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{W}_{\text{fc}} \mathbf{h}_{\text{global}} + \mathbf{b}_{\text{fc}}). \tag{22}$$

C. Stepwise Lidar Fault Detection Flow

To enhance robustness, lidar fault detection is complemented by a stepwise verification procedure that combines simple heuristics with model inference confidence. This procedure includes the following stages:

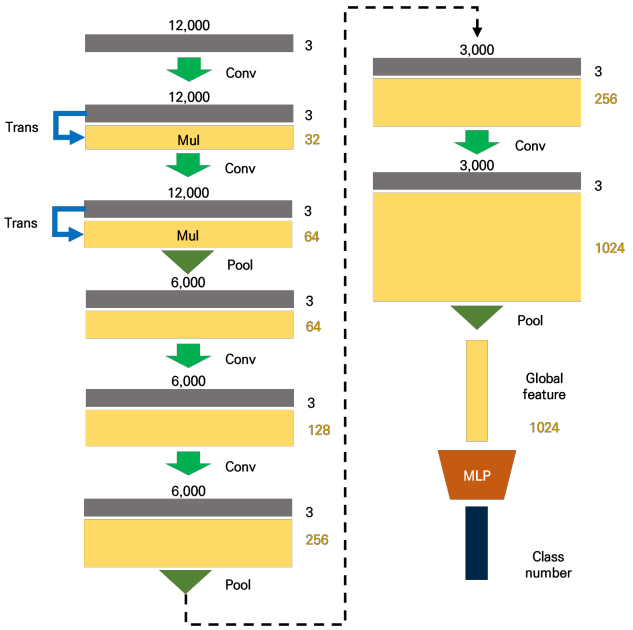


Fig. 6. Hybrid 3D-GCN and Transformer architecture for lidar fault detection.

1) Point Count Threshold Check: Let N_{points} denote the number of points in a received point cloud and N_{thresh} a predefined threshold. A potential lidar fault is suspected if:

$$N_{\text{points}} < N_{\text{thresh}} \tag{23}$$

Otherwise, data acquisition is further verified.

2) Data Acquisition Timer-Based Check: The duration required to acquire the point cloud, T_{duration}, is compared with a threshold T_{thresh}, setting a timer fault flag as:

$$\text{TimerFlag} = \begin{cases} \text{True}, & T_{\text{duration}} \geq T_{\text{thresh}} \\ \text{False}, & T_{\text{duration}} < T_{\text{thresh}} \end{cases} \tag{24}$$

If consecutive timer faults exceed a predefined limit $T_{\text{counter\_thresh}}$, a data acquisition error is concluded.

3) Model Confidence-Based Verification: After initial checks, the pre-trained lidar fault detection model provides a confidence score $C_{\text{model}} \in [0, 1]$. The lidar fault flag is set according to:

$$\text{LidarFaultFlag} = \begin{cases} \text{True}, & C_{\text{model}} \geq C_{\text{thresh}} \\ \text{False}, & C_{\text{model}} < C_{\text{thresh}} \end{cases} \tag{25}$$

If consecutive model fault flags exceed a counter threshold $L_{\text{counter\_thresh}}$, a lidar fault is concluded.

4) Final Fault Output: If a lidar fault is detected through either the timer-based accumulation or model confidence evaluation, the system outputs the fault result through a designated interface, potentially signaling other modules within an autonomous vehicle control system. This stepwise procedure ensures robust and reproducible lidar fault detection by combining lightweight heuristics with learned feature representations, complementing the hybrid 3D-GCN and Transformer architecture described above.

Fig. 7. Real-world lidar data collection setup across highways, urban roads, and rural environments.

TABLE I
PERFORMANCE COMPARISON OF LIDAR FAULT DETECTION MODELS ON THE SYNTHETIC VALIDATION SET.

Model	Sampling	Acc (%)	#Param
CNN	FPS	30.62	3.32M
CNN	Distance-based	67.34	3.32M
PointNet++	FPS	28.28	1.48M
PointNext	FPS	32.64	0.80M
3D-GCN	Distance-based	94.45	0.89M
3D-GCN + Transformer (Ours)	Distance-based	**96.25**	1.08M

IV. EXPERIMENTS

A. Experimental Setup

Experiments were conducted using synthetic fault data generated from real-world lidar scans, enabling controlled learning and evaluation of labeled fault scenarios. Normal point clouds were collected using a Velodyne HDL-32E (32 channel) lidar across highways, urban, and rural roads during both day and night, with a sampling rate of 10 to 15 Hz. The dataset consists of 7,000 normal point clouds and 21,000 synthetic fault point clouds (7,000 per fault type: channel failure, global occlusion, partial occlusion). Using the proposed distance-based sampling, $N_{\text{input}} = 12{,}000$ points per cloud were selected. The dataset was split into training, validation, and testing subsets in a 70 percent, 15 percent, and 15 percent ratio. Preprocessing steps were applied to all point clouds, and optional data augmentation, including rotations, translations, and Gaussian noise, was performed. Evaluation metrics include classification accuracy and confusion matrices.

The hybrid 3D-GCN + Transformer network was implemented in PyTorch with 4 graph convolution layers ($F = 64$), $k = 16$ neighbors, and 2 Transformer encoder layers. Training was performed with a batch size of 16 using the Adam optimizer (1×10^{-3}) for 100 epochs, incorporating online data augmentation. All experiments were conducted on an NVIDIA RTX 3090 GPU, as shown in Fig. 7. Baseline models include CNN with Farthest Point Sampling(FPS), CNN with distance-based sampling, PointNet++ [2] with FPS, PointNext [12] with FPS, and 3D-GCN with distance-based sampling [4].

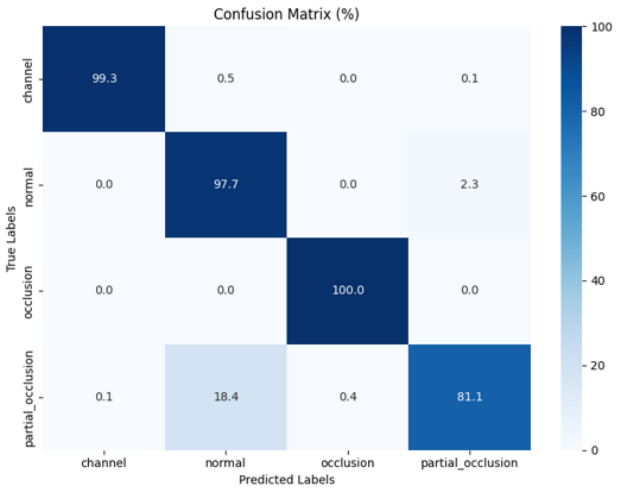


Fig. 8. Confusion matrix of the proposed 3D-GCN + Transformer model on the synthetic test set.

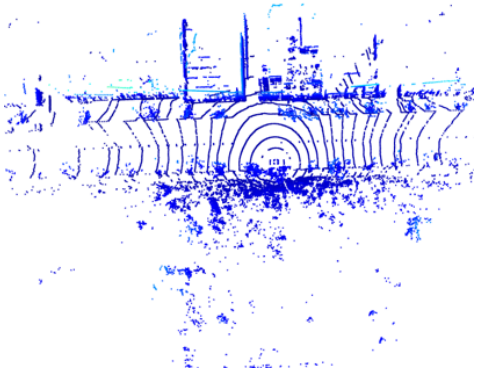
B. Results

Table I summarizes the validation performance on the synthetic dataset during training. The proposed 3D-GCN + Transformer model achieved 96.25% validation accuracy, outperforming CNN with distance-based sampling (67.34%), CNN-FPS (30.62%), PointNet++-FPS (28.28%), PointNext-FPS (32.64%), and standalone 3D-GCN with distance-based sampling (94.45%). Ablation studies indicate that including the Transformer module improved validation accuracy by 1.8

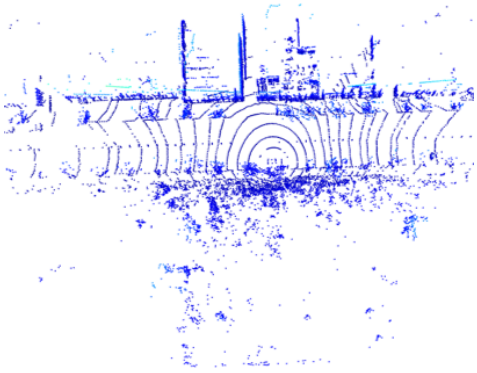
To evaluate generalization, a separate test set was used to compute the confusion matrix, shown in Fig. 8. On the test set, channel failures and global occlusions were classified almost perfectly, while partial occlusions achieved 81.1% accuracy. Distance-based sampling contributed significantly by preserving local point density variations, which are critical for detecting lidar faults, as illustrated in Fig. 9. In contrast, FPS methods tend to obscure these density differences, often producing patterns resembling global occlusions, which reduces classification performance.

C. Discussion

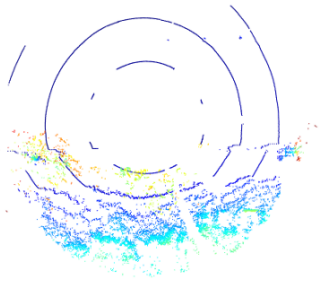
The experimental results demonstrate that the proposed hybrid 3D-GCN + Transformer framework generalizes well across multiple lidar fault types. The model effectively combines local geometric feature extraction with global context modeling, enabling robust detection even in sparse point clouds. Distance-based sampling is critical for maintaining local point density variations, which facilitate distinguishing between fault types. FPS is commonly adopted in point cloud classification models, has traditionally been used in deep learning-based point cloud methods to classify specific objects while preserving their shape as much as possible. However, it can obscure density differences and produce patterns similar to global occlusions, resulting in lower performance in both partial and global occlusion scenarios.



(a) Original



(b) FPS



(c) Distance-based

Fig. 9. Comparison of point cloud sampling methods: (a) original (47,150 points), (b) FPS-sampled (12,000 points), and (c) distance-based sampled (12,000 points), which better preserves spatial distribution.

V. Conclusion

We proposed a hybrid 3D-GCN + Transformer framework for lidar fault detection, leveraging distance-based sampling to preserve point cloud connectivity and density variations. The model achieved 96.25% accuracy, with the Transformer module contributing a 1.8% improvement, highlighting the benefit of long-range dependency learning. Distance-based sampling outperformed FPS by retaining density patterns essential for detecting subtle faults. Although effective on synthetic datasets, real-world lidar faults may involve complex interactions not fully captured synthetically. Future work will extend synthetic data coverage, develop adaptive threshold strategies, and validate the framework on large-scale real-world datasets for real-time autonomous vehicle integration. Overall, the proposed approach provides a practical and robust solution for lidar fault detection with potential for real-time deployment.

Acknowledgment

This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP), funded by the Korean government (MSIT) under grant No. RS-2023-00229833, for the development of intelligent teleoperation technology for cloud based autonomous vehicle errors and limit situation.

References

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 652–660.

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[3] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7652–7660.

[4] Z. H. Lin, S. Y. Huang, and Y. C. F. Wang, "Convolution in the cloud: Learning deformable kernels in 3D graph convolution networks for point cloud analysis," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1800–1809.

[5] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point Transformer," in *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2021, pp. 16259–16268.

[6] Z. Peng, W. Huang, H. Guo, X. Huang, J. Liang, J. Li, and J. Han, "Conformer: Local Features Coupling Global Representations for Visual Recognition," in *Proc. ICCV*, 2021, pp. 367–376.

[7] N. Sboui, M. Hadded, H. Ghazzai, M. Elhadef, and G. Setti, "Anomaly detection in autonomous vehicle's LiDAR sensor data using variational autoencoders," in *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, 2024, pp. 1–5.

[8] T. Goelles, B. Schlager, and S. Muckenhuber, "Fault detection, isolation, identification and recovery (FDIIR) methods for automotive perception sensors including a detailed literature survey for LiDAR," *Sensors*, vol. 20, no. 13, p. 3662, 2020.

[9] B. Schlager, T. Goelles, S. Muckenhuber, and D. Watzenig, "Contaminations on LiDAR sensor covers: Performance degradation including fault detection and modeling as potential applications," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 738–747, 2022.

[10] J. K. James, G. Puhlfürst, V. Golyanik, and D. Stricker, "Classification of LiDAR sensor contaminations with deep neural networks," in *Proc. Computer Science in Cars Symposium (CSCS)*, vol. 8, 2018.

[11] M. H. Guo, J. X. Cai, Z. N. Liu, T. J. Mu, R. R. Martin, and S. M. Hu, "PCT: Point Cloud Transformer," *Comput. Visual Media*, vol. 7, no. 2, pp. 187–199, 2021.

[12] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem, "PointNext: Revisiting PointNet++ with Improved Training and Scaling Strategies," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, pp. 23192–23204, 2022.

[13] F. Chang, E. Jafarzadeh, J. Del Gatto, G. Cran, and H. Sadjadi, "Failure Mode Investigation to Enable LiDAR Health Monitoring for Automotive Application," in *Proc. Annu. Conf. PHM Soc.*, vol. 15, no. 1, Oct. 2023.

[14] M. Dreissig, D. Scheuble, F. Piewak, and J. Boedecker, "Survey on LiDAR Perception in Adverse Weather Conditions," *arXiv preprint* arXiv:2304.06312, 2023.

[15] J. Park, K. Kim, and H. Shim, "Rethinking data augmentation for robust LiDAR semantic segmentation in adverse weather," in *Proc. European Conf. Computer Vision (ECCV)*, Cham, Switzerland: Springer, 2024, pp. 320–336.

[16] G. Jati, M. Molan, F. Barchi, A. Bartolini, G. Mercurio, and A. Acquaviva, "ANZIL: Attention-based network for zero-risk inspection of LiDAR point cloud in self-driving cars," *Expert Systems with Applications*, 2025, Art. no. 128412.

[17] A. Omoseebi, A. Ali, A. Muthanna, M. Alkhalidy, R. R. Kumar, and M. Tiwari, "Radar and LiDAR sensor anomaly detection using deep multimodal architectures," 2025.