Sensors and Materials, Vol. 37, No. 9 (2025) 3855–3867
MYU Tokyo

S & M 4157

2D and 3D LiDAR with CNN Models for Detecting Sediment Accumulation Underground after Disasters

Woranidtha Krungseanmuang,[1] Fuka Morita,[2] Vasutorn Chaowalittawin,[1]
Posathip Sathaporn,[1] Chisato Kanamori,[2]
Tuanjai Archevapanich,[3] and Boonchana Purahong[4*]

[1]Department of Robotics and AI Engineering, School of Engineering,
King Mongkut's Institute of Technology Ladkrabang,
1 Chalong Krung Rd. Lat Krabang district, Bangkok 10520, Thailand
[2]Department of Mechanical and Intelligent Systems Engineering, Faculty of Engineering,
The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
[3]Department of Electronics and Communication Engineering, Faculty of Engineering and Architecture,
Rajamagala University of Technology Suvanabhumi,
217 Nonthaburi1 Rd. Suanyai district, Nonthaburi 11000, Thailand
[4]Department of IoT and Information Engineering, School of Engineering,
King Mongkut's Institute of Technology Ladkrabang,
1 Chalong Krung Rd. Lat Krabang district, Bangkok 10520, Thailand

(Received May 1, 2025; accepted June 11, 2025)

Keywords: 3D point cloud data, 2D object detection, semantic segmentation, CNN, YOLO, disaster

Global climate change impacts all regions and leads to natural disasters such as typhoons, which cause destruction, debris, and flooding. Postdisaster restoration is a very important activity that is mostly done manually and can be time-consuming and challenging, especially in subterranean environments owing to accumulated objects such as pipes, pillars, and mud distributed in confined underground areas. Therefore, in this study, we aim to utilize emerging AI technologies by comparing deep learning algorithms and evaluating four models for 2D object detection and four for 3D point cloud segmentation for detecting sediment accumulation and navigating around obstacles in underground areas after a disaster. Additionally, a custom dataset was developed to simulate underground disaster scenarios. As a result, the You Only Look Once version 11 (YOLOv11) model achieved the highest mean average precision 50 (*mAP*50: 91.1%) for general detection within the pillar-pipe dataset, whereas the YOLOv12 model performed the best in detecting pipes (*mAP*50: 87.7%). In the mud dataset, the YOLOv8 segmentation (YOLOv8-seg) model demonstrated superior performance with *mAP*50 scores of 93.0% (detection) and 86.4% (segmentation). For 3D point cloud segmentation, PointNet achieved the highest accuracy (98.61%), whereas RandLA-Net was optimal for pipe segmentation, achieving an intersection over union score of 37.1%. These findings highlight AI's potential to accelerate disaster recovery, reduce manual labor, and ensure faster cleanup. Integrating deep learning models into post-typhoon restoration efforts can enable communities to recover more quickly and efficiently after climate change impacts or disaster events.

*Corresponding author: e-mail: boonchana.pu@kmitl.ac.th
https://doi.org/10.18494/SAM5713

ISSN 0914-4935

1. Introduction

As climate change raises global temperatures, it increases the frequency and severity of natural disasters such as typhoons and hurricanes. These powerful storms cause not only immediate destruction but also long-term impacts on human health, economic stability, infrastructure, and the environment.

One major challenge is disaster waste, especially construction and demolition debris, which is often difficult to access because of blocked areas and biosecurity concerns. This study is focused on Japan, a country frequently hit by typhoons. Its temperate climate, mountainous landscape, and heavy rainfall make it especially vulnerable to flooding, particularly in low-lying and urban areas.

Many Japanese houses have a crawl space beneath the floor for ventilation and to protect wooden structures from decay. These spaces are particularly tricky to clean after floods. Currently, flood cleanup, which involves draining water, moving damaged items outside, cleaning, drying, disinfecting, and removing mud, is done manually by humans.

Postdisaster cleanup is critical for protecting public health, restoring essential services, and preventing further environmental harm. In this research, we explore how modern technologies like AI can, in the future, improve the process by shifting dangerous tasks from humans to sensors and robots.

Therefore, in this paper, we present a performance comparison of using 2D and 3D LiDAR with convolutional neural network (CNN) models to detect sediment accumulation in underground flooding after disasters. Regarding model evaluation, mean average precision 50 (*mAP*50) and mean average precision 50–95 (*mAP*50–95) are used to evaluate the performance of pipe and pillar object detection models. The same metrics are also used for evaluating mud segmentation. For point cloud data, performance is evaluated using accuracy and the mean intersection over union (mIoU). Our goal is to evaluate the performance of deep learning models with 2D and 3D LiDAR inputs, identify the best practices, and potentially integrate this technology into autonomous robots. Such integration would reduce manual labor, speed up postdisaster recovery, and minimize risks to human workers.

2. Literature Review

LiDAR for 3D object detection applications has been utilized in numerous studies. Alaba and Ball introduced a LiDAR-based object detection and feature extraction method.[1] He *et al.* proposed the Stereo RGB and Deeper LiDAR (SRDL) framework for combining semantic and spatial information and concluded that fusing multimodal data leads to more robust and reliable 3D object detection systems.[2] Gan *et al.* presented a hybrid filtering algorithm to denoise the single-view blisk point cloud data to remove noise. The octree downsampling method was used to simplify the blisk point cloud data, which is beneficial to accelerate the subsequent point cloud segmentation.[3] Jin *et al.* improved the PointNet classification accuracy using a functional multihead pooling encoding model. This approach reduced overfitting and improved

classification accuracy by up to 3.25% on datasets such as ModelNet40.[4] Sun *et al.* proposed a 3D point cloud classification algorithm based on improved PointNet++. These improvements led to better classification accuracy on datasets such as ISPRS Vaihingen and GML(B) by effectively handling uneven point distributions and complex scenes.[5] Barnefske and Sternberg conducted a detailed study on hierarchical patterns (HPs), examining their class distribution and combinations. They used seven hierarchical class combinations and four class size adaptation methods to systematically analyze the HPs.[6] Yang *et al.* introduced a combination of an average pooling layer with the X-Conv method to replace max pooling and address the issue of feature loss.[7] Zhou *et al.* proposed a new segmentation method based on PointNet and VoxelNet to improve the efficiency and accuracy of semantic segmentation.

A number of You Only Look Once (YOLO)-based studies have also been widely applied in segmentation tasks.[8] Casas *et al.* compared YOLOv5 and YOLOv8 in corrosion segmentation. Their work highlighted YOLOv8's high speed and segmentation accuracy across all datasets, particularly in handling complex corroded surfaces.[9] Shreyas *et al.* reviewed 3D object detection applications across fields such as robotics and the military.[10] Xu *et al.* introduced a metalearning approach for 3D classification,[11] while Shi and Rajkumar proposed a graph neural network for LiDAR-based object detection.[12] Deng *et al.* developed an obstacle detection algorithm for unmanned surface vehicles.[13]

Many modern applications leverage point cloud data with deep learning for classification and segmentation tasks.[14] Qi *et al.*[15] and Muzahid *et al.*[16] reviewed deep learning methods for multiview 3D recognition. Hou and Zhang focused on shallow mud detection in submarine channels using YOLOv5s-EF.[17] Aulia *et al.* developed a CNN-based detection system for autonomous mobile robots (AMRs). The model demonstrates improved classification and detection accuracy, indicating its potential for AMR applications.[18] Dos Reis *et al.* implemented YOLO with Kinect for obstacle detection, achieving high detection accuracy and low distance error.[19] Chikurtev *et al.* combined LiDAR and GPS for mobile robot localization, enabling precise navigation even in narrow or dynamic environments.[20] Baatar *et al.* automated underwater object annotation using sonar images.[21] Chang *et al.* compared OpenPose and MoveNet for real-time fall detection using webcams, emphasizing lightweight AI's role in eldery care and healthcare integration.[22] Zhang *et al.* proposed a CNN–GRU model for predicting regional weather using historical data. Tested on Beijing's 1901–2022 climate records, it outperformed LSTM and GRU models, showing higher accuracy in long-term forecasting.[23] Finally, Rakhsith *et al.*[24] and Pavani *et al.*[25] compared object and face detection techniques, highlighting YOLOv5's efficiency in real-time detection.

In this study, we aim to identify the best CNN-based object detection model by comparing the performances on both 2D images and 3D point clouds. While many groups use YOLO for detection, most focus on a single input type. We address this gap by comparing models across different data inputs to find the most effective approach for object detection and sentiment analysis.

3. Materials and Methods

We divided the materials and methods into two categories, 2D images and 3D point cloud data. The details of the data are as below, and the overall methodology flowchart is shown in Fig. 1.

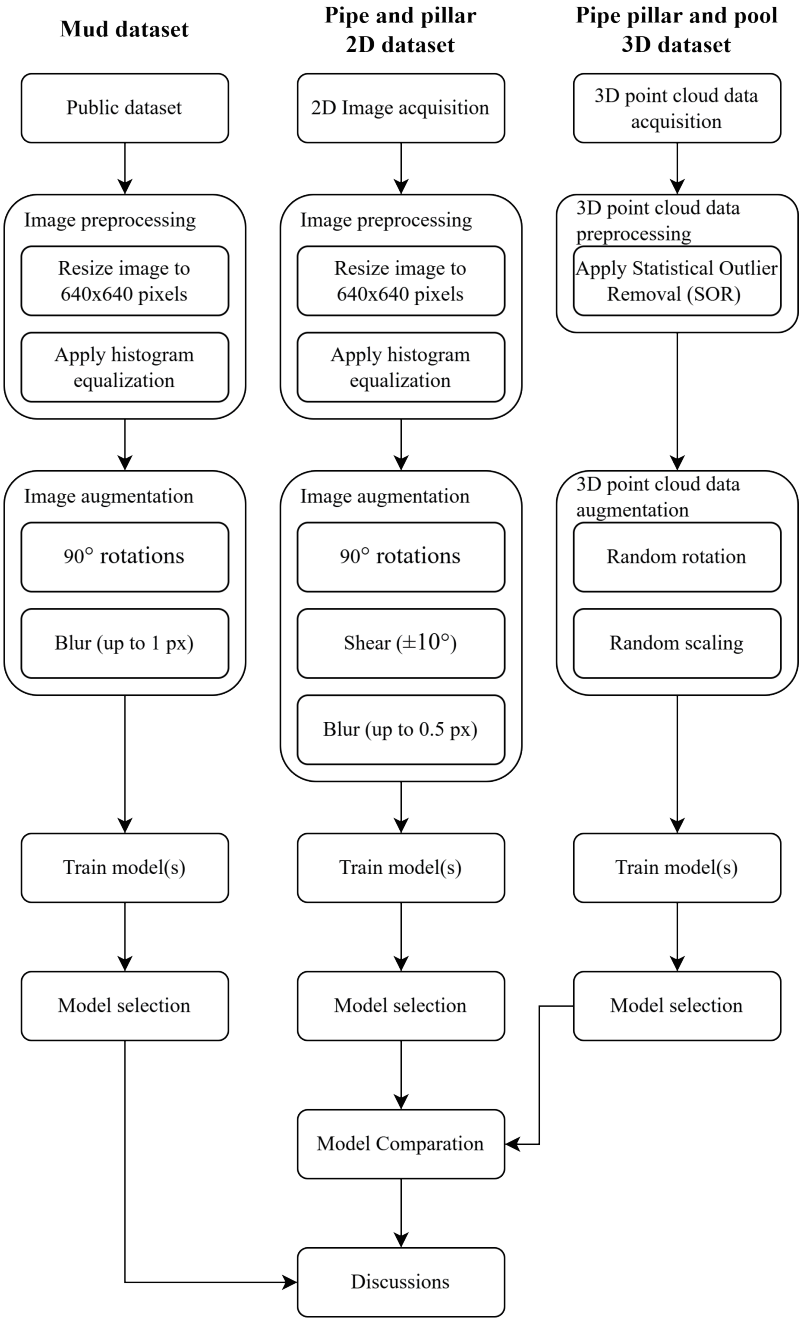


Fig. 1. Overall methodology flowchart.

3.1 2D image dataset

A custom dataset was developed to represent real-world disaster site conditions. The process began with the creation of a simulated postdisaster scenario, featuring a pool to replicate a flooded under-house area, along with pillars and pipes to represent mixed elements commonly found accumulating in water. Mud was also added to enhance realism. Images were captured under both daylight and low-light conditions to ensure environmental diversity. The resulting dataset[26] consists of 2094 images (3024 × 4032 × 3), as shown in Fig. 2. The dataset is divided into training 87.41% (1830 images), validation 8.31% (174 images), and testing 4.30% (90 images) subsets, with two classes: *pipe* and *pillar*

The images were resized to 640 × 640 pixels, and histogram equalization was applied after preprocessing. Then, data augmentation techniques were used to increase dataset variability, including 90° rotations (clockwise, counterclockwise, and upside down), shear transformations (±10° horizontally and vertically), and blurring (up to 0.5 pixels). Figure 3 presents examples from the pipe and pillar 2D dataset.

We utilized a 2D mud dataset, a publicly accessible resource available to the public.[27] This dataset was designed for instance segmentation and comprises 665 images partitioned into training (579 images), validation (56 images), and testing (30 images) sets. Prior to analysis, all images underwent preprocessing, including resizing to 640 × 640 pixels and histogram equalization to enhance image contrast. Additionally, data augmentation techniques were applied to increase dataset variability. These techniques included 90° rotations (clockwise, counterclockwise, and upside down), the introduction of noise up to 0.1% of pixels, and the

Fig. 2. (Color online) Example of original images from the camera.

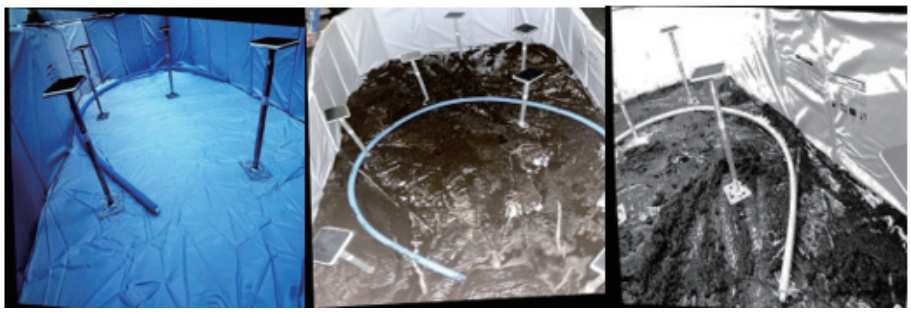


Fig. 3. (Color online) Pipe and pillar 2D dataset.

application of blur with a maximum radius of 1 pixel. Figure 4 shows example images of the mud dataset.

3.2 3D point cloud data dataset

The 3D point cloud data was obtained using the Livox MID-360 LiDAR sensor. The original data includes x, y, z and intensity information. Three-dimensional point cloud data acquired from sensors often contains noise. In particular, distance data at object boundaries is frequently inaccurate, resulting in point clouds extending backward like shadows from the edges of objects. Additionally, incorrect distance data may be obtained owing to specular reflections. To eliminate such outlier point cloud data that exists in space and not on the actual surface of the object, statistical outlier removal (SOR) was performed. In this method, a normal distribution of neighbor distances is assumed and points that lie beyond a statistically defined threshold are removed, improving the spatial coherence of the surface. The results of the SOR preprocessing steps are illustrated in Fig. 5. The mean distance μ_d can be calculated using Eq. (1), where k represents the number of neighboring points, and d_i is the distance between the target point and a neighboring point. σ_d, the standard deviation of distances, is obtained from Eq. (2). Additionally, the threshold is determined using Eq. (3), where std_{ratio} refers to the threshold multiplier.

$$\mu_d = \frac{1}{k}\sum_{i=1}^{k} d_i \tag{1}$$

$$\sigma_d = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(d_i - \mu_d)^2} \tag{2}$$

$$Threshold = \mu_d + std_{ratio} \times \sigma_d \tag{3}$$

To further enhance the dataset, data augmentation techniques are applied, specifically, random rotation and random scaling. These methods increase the number of training samples while preserving the original geometric structure of the point cloud data.

A dataset consisting of these features was used to create training and test datasets for segmentation methods by utilizing CloudCompare. The dataset consists of 100 3D point cloud

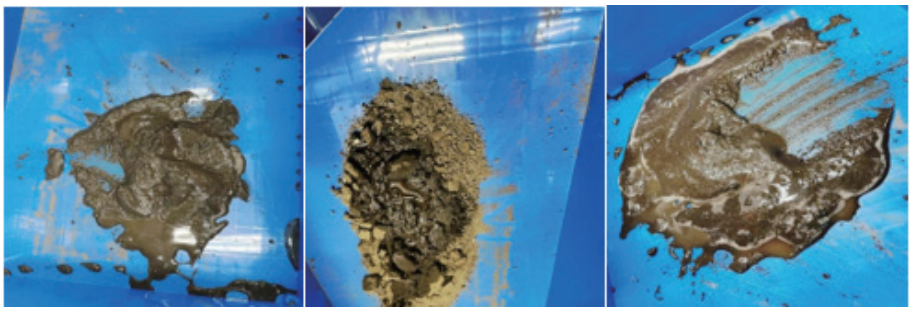
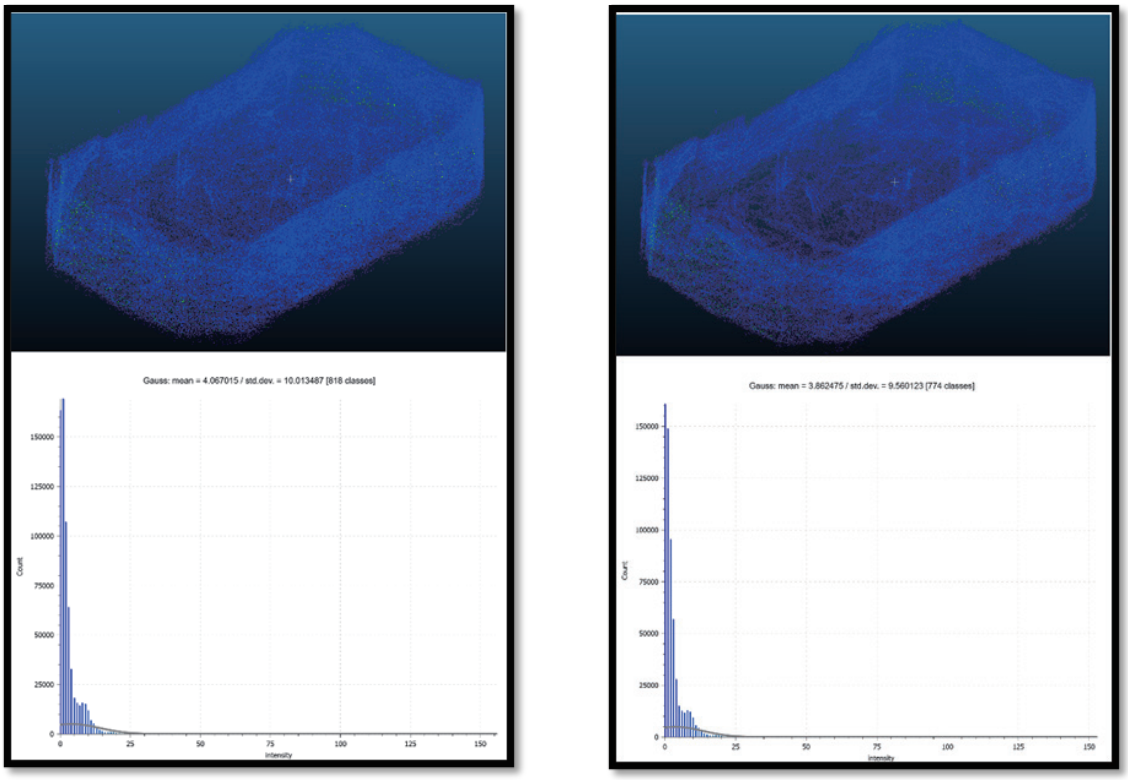


Fig. 4. (Color online) Mud dataset images.



(a) (b)

Fig. 5. (Color online) (a) Original 3D point cloud data and histogram. (b) 3D point cloud data and histogram after SOR.

images divided into a training set (70 images), a validation set (15 images), and a test set (15 images). This dataset is referred to as the pipe pillar and pool 3D dataset.

3.3 Model selection

3.3.1 2D model selection

The selection of the YOLO model family (YOLOv5, YOLOv8, YOLOv11, and YOLOv12[28–31]) for our endeavor was driven by its computational efficiency, a paramount consideration for autonomous robotic systems. YOLO has a single-stage detection architecture, in contrast to the region proposal-based approach of Faster R-CNN, significantly expediting processing. The single-stage architecture, coupled with its fully convolutional network (FCN) framework, facilitates rapid inference, achieving 30–60 frames per second (FPS) on graphical processing units (GPUs) and demonstrating compatibility with embedded systems such as Jetson Nano and Raspberry Pi. The deployment of multiple YOLO versions was a deliberate strategy aimed at ascertaining the optimal model configuration for our unique datasets. For the task of mud segmentation, our focus was narrowed to YOLOv8 and YOLOv11, as these are the only

iterations currently equipped to handle segmentation tasks. Moreover, YOLOv11's capacity for real-time segmentation on edge and cloud platforms renders it particularly germane to our specific application.

To mitigate the risk of overfitting, we implemented an early stopping mechanism. This involved monitoring the model's performance on a validation dataset during training. If the model's performance on the validation 1 set began to deteriorate after a predetermined number of epochs, the training process was automatically terminated, thereby ensuring the selection of a model with optimal generalization capabilities.

3.3.2 3D point cloud data model selection

PointNet serves as a foundational model for direct point cloud processing, operating on 3D point data. It employs symmetric functions, such as max pooling, to achieve permutation invariance, allowing it to handle 3D data without the need for conversion to other formats such as voxel grids or meshes. While user-friendly and efficient in processing 3D points, PointNet faces limitations in capturing complex local relationships between points.[32] To address these limitations, PointNet++ was developed, utilizing hierarchical feature learning to manage varying data densities across different regions. It incorporates local region grouping and pointwise feature learning for enhanced data representation, achieving high efficiency in handling complex point cloud data. However, this comes at the cost of increased model complexity and training time.[33] Point-voxel CNN (PvCNN) introduces a hybrid approach, combining point clouds and voxel grids by converting point cloud data into voxel grids before CNN processing. This method leverages the advantages of voxel-based learning for deep feature extraction while preserving point cloud details. It is well suited to 3D data classification and segmentation, although the conversion process may result in some detail loss and requires increased storage because of the larger voxel grid sizes.[34] RandLA-Net employs local feature aggregation through random point selection from point cloud data to reduce processing complexity. This approach enables high processing speeds, making it ideal for tasks requiring real-time processing of large point clouds. However, the random point selection may lead to the loss of certain critical information.[35]

4. Results

The pipe and pillar 2D dataset was tested using multiple YOLO versions, as shown in Table 1. YOLOv11 exhibited the highest overall performance, achieving an *mAP*50 of 91.1% and an *mAP*50-95 of 67.3%. Notably, YOLOv12 achieved a comparable *mAP*50-95 score while demonstrating superior performance in pipe detection (87.7%) compared with YOLOv11 (84.7%). However, YOLOv11 maintained a slight advantage in pillar detection with an *mAP*50 of 94.8%, the highest among all tested models. The learning curves of the models on the pipe and pillar 2D dataset are shown in Fig. 6, and object detection via YOLOv11 is shown in Fig. 7.

For the mud dataset, YOLOv8-seg outperformed YOLOv11-seg in most metrics, as shown in Table 2. YOLOv8-seg attained an *mAP*50 of 93.0% for object detection, compared with 89.8%

Table 1
Results for pipe and pillar 2D dataset.

No.	Model	*mAP*50 (%)	*mAP*50-95 (%)	*mAP*50 by class (%)	
				pipe	pillar
1	YOLOv5	89.20	61.90	84.00	94.40
2	YOLOv8	90.20	65.20	86.10	94.30
3	YOLOv11	91.10	67.30	84.70	94.80
4	YOLOv12	90.90	67.30	87.70	94.10

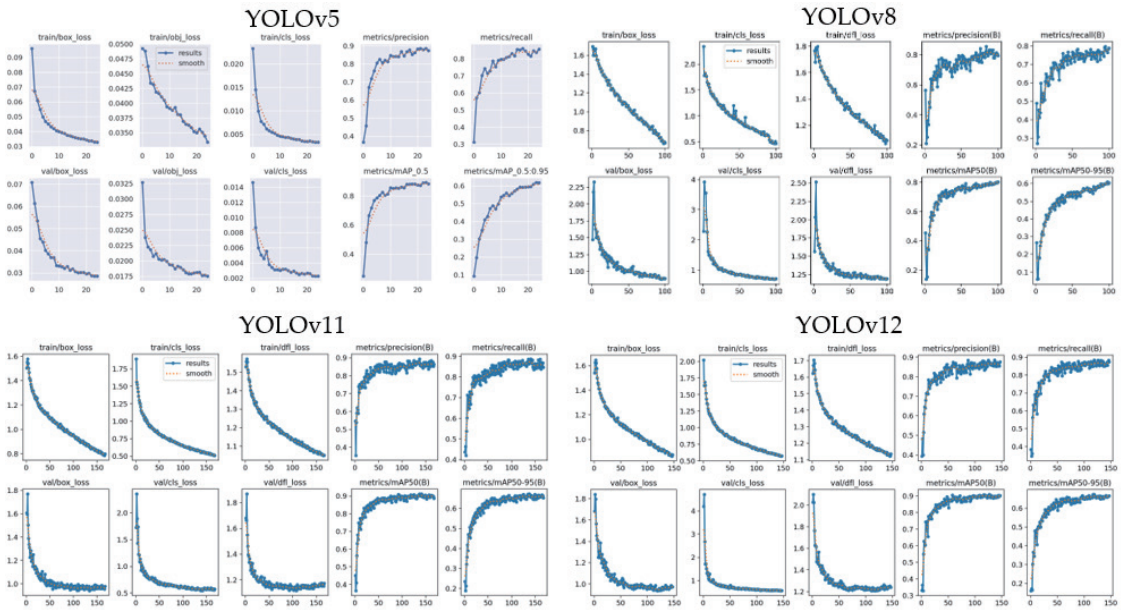


Fig. 6. (Color online) Learning curves of the models on the pipe and pillar 2D dataset.

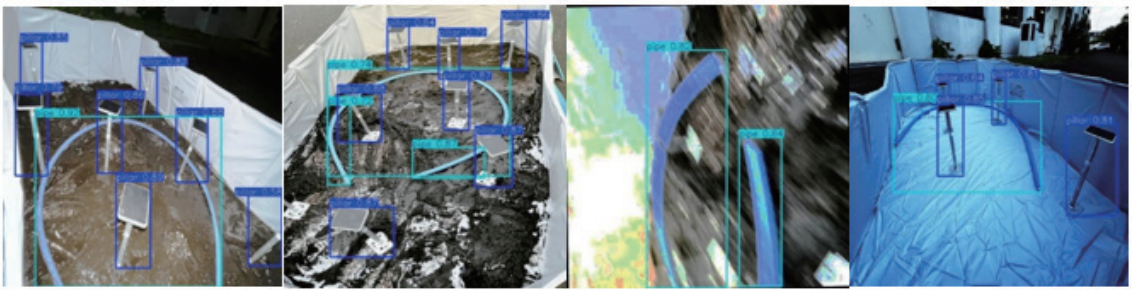


Fig. 7. (Color online) Pipe and pillar detection using YOLOv11.

Table 2
Results for mud 2D dataset.

No.	Model	Box		Mask	
		*mAP*50 (%)	*mAP*50-95 (%)	*mAP*50 (%)	*mAP*50-95 (%)
1	YOLOv8-seg	93.00	60.60	86.40	47.10
2	YOLOv11-seg	89.80	55.40	86.50	46.60

for YOLOv11-seg. Additionally, YOLOv8-seg surpassed YOLOv11-seg in segmentation performance, with an *mAP*50 of 86.4%, although its *mAP*50-95 of 47.1% was slightly higher than YOLOv11-seg's 46.6%. The learning curves of the models on the mud dataset are shown in Fig. 8, and mud object detection via YOLOv8-seg is shown in Fig. 9.

The evaluation of 3D point cloud data revealed that PointNet achieved the highest evaluation accuracy (98.61%), with the best mIoU of 43.05%. RandLA-Net, however, demonstrated superior performance for pipe segmentation, obtaining an IoU of 37.1%, surpassing other models in this specific class. Meanwhile, PointNet++ exhibited lower performance overall, with a best mIoU of 28.31% and an IoU for pipe detection of only 12.0%, as shown in Table 3.

These findings indicate that YOLOv11 was the optimal model for general object detection in the pipe and pillar dataset, whereas YOLOv12 was preferable for pipe detection. For mud detection, YOLOv8-seg was the most effective for both object detection and segmentation. In 3D

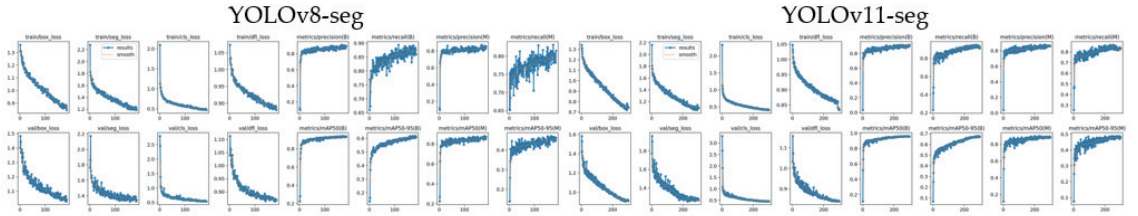


Fig. 8. (Color online) Learning curve patterns for mud dataset.

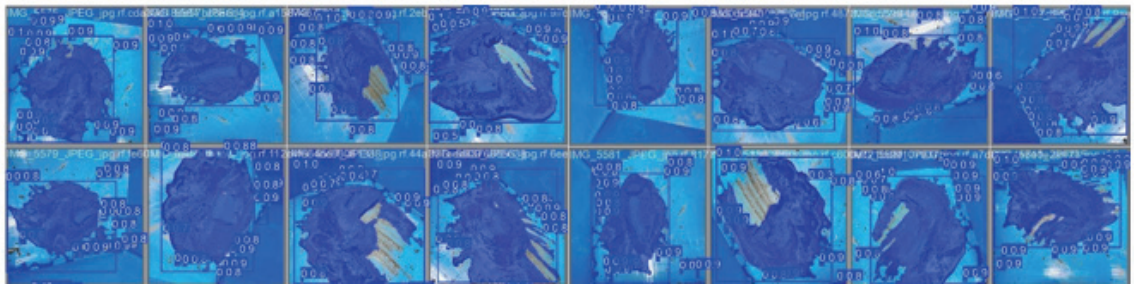


Fig. 9. (Color online) Mud detection using YOLOv8-seg.

Table 3
Results of training on 3D point cloud data.

No.	Model	Categories	Eval accuracy (%)	Best mIoU (%)	IoU by class (%)		
					pipe	pillar	pool
1	PointNet++	Semantic segmentation	94.25	28.31	12.00	6.50	94.30
2	PointNet	Semantic segmentation	**98.61**	**43.05**	30.30	**59.20**	**98.60**
3	PvCNN	Semantic segmentation	97.39	26.47	0.10	8.50	97.39
4	RandLA-Net	Semantic segmentation	95.21	39.78	**37.10**	43.10	97.45

point cloud classification, PointNet achieved the highest accuracy, but RandLA-Net was the best choice for detecting pipes. The results of semantic segmentation via PointNet are shown in Fig. 10.

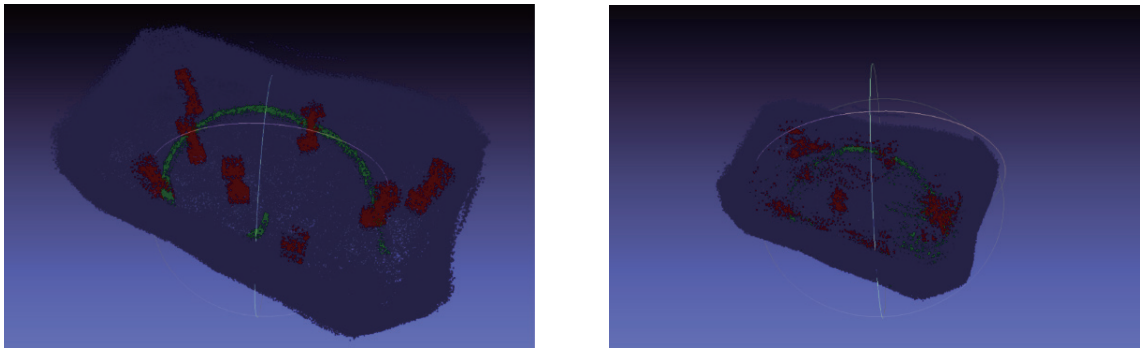
5. Discussion

In this study, we assessed multiple versions of YOLO and 3D point cloud models to determine which were most effective on custom disaster-site datasets. The results showed that model performance varied depending on object type, data characteristics, and how the models process spatial features. YOLOv12 produced the best results on the general-purpose COCO dataset because it was trained and optimized on diverse, large-scale image data, making it highly generalizable. However, YOLOv11 proved to be the most effective for our custom pipe and pillar dataset, achieving an *mAP*50 of 91.1%. This is likely because YOLOv11's feature extraction layers and anchor configurations aligned better with the visual patterns, shapes, and scales present in the custom data, particularly for pillar detection, where it achieved the top class-specific performance (*mAP*50 of 94.8%).

Interestingly, when focusing specifically on pipe detection, YOLOv12 outperformed YOLOv11 (87.7% vs 84.7%), suggesting that YOLOv12's updated architecture was better at capturing the elongated, cylindrical shapes of pipes, possibly as a result of improved backbone layers or multiscale processing, which handle fine details better.

For the mud dataset, YOLOv8-seg performed best, achieving an *mAP*50 of 89.8% and an *mAP*50–95 of 60.6% for bounding box detection. Its superior segmentation performance (*mAP*50–95 of 47.1%) over YOLOv11-seg likely comes from YOLOv8's enhanced segmentation head, which includes finer spatial resolution and better boundary refinement. These improvements allowed YOLOv8 to handle the irregular, often low-contrast mud surfaces more effectively, where subtle texture differences make detection challenging.

In the 3D point cloud classification task, PointNet achieved the highest overall accuracy (98.61%) and mIoU (43.05%), largely because its architecture is designed to directly process raw point clouds, capturing global and local features without needing voxelization or grid conversion.



(a) (b)

Fig. 10. (Color online) (a) Original 3D point cloud data. (b) After semantic segmentation using PointNet.

However, when focusing on pipe-specific 3D object detection, RandLA-Net performed better, as its local feature aggregation mechanism and efficient handling of large-scale, sparse point clouds allowed it to better capture fine, pipelike structures.

Importantly, detecting mud using 3D point cloud data remained challenging because LiDAR technology excels at reconstructing complex, raised surfaces such as trees or debris, where laser signals reflect cleanly. In contrast, mud on flat surfaces produces weak, noisy, or incomplete point returns, making it difficult for the models to extract distinctive features or consistent geometric patterns.

Overall, the results highlight that model performance is determined not only by architecture but also by the match between model design and the dataset's visual or geometric characteristics, the quality of pre- and postprocessing steps, and the specific detection tasks (general vs class-specific performance).

6. Conclusions

In this study, the importance of selecting suitable CNN models for different applications involving 2D images and 3D LiDAR inputs was highlighted. YOLOv11 was the most effective for general object detection on our custom 2D dataset, while YOLOv12 performed best for detecting pipes. For mud segmentation in 2D data, YOLOv8 achieved the highest accuracy. In the 3D domain, PointNet proved to be the most effective overall, whereas RandLA-Net was better suited for pipe-specific tasks. However, detecting mud using LiDAR-based 3D point clouds remains a significant challenge owing to the inherent difficulties in data acquisition and labeling under such conditions. Alternative approaches, such as sensor fusion or advanced annotation techniques, should be explored in future research to improve the accuracy and practicality of mud detection in real-world scenarios.

References

1 S. Y. Alaba and J. E. Ball: Sensors **22** (2022) 9577. https://doi.org/10.3390/s22249577
2 Q. He, Z. Wang, H. Zeng, Y. Zeng, Y. Liu, S. Liu, and B. Zeng: IEEE Trans. Intell. Transp. Syst. **24** (2023) 152. https://doi.org/10.1109/TITS.2022.3215766
3 X. Gan, L. Tan, and X. Wang: Proc. 2022 5th World Conf. Mechanical Engineering and Intelligent Manufacturing (IEEE, 2022) 990–994. https://doi.org/10.1109/WCMEIM56910.2022.10021495
4 Z. Jin, F. Xu, Z. Lu, and J. Liu: IEEE Access **12** (2024) 127931. https://doi.org/10.1109/ACCESS.2024.3456107
5 Z. Sun, J. Hu, S. Jiang, Z. Li, K. Deng, and Z. Zhang: Proc. 2024 4th Int. Conf. Machine Learning and Intelligent Systems Engineering (IEEE, 2024) 294–298. https://doi.org/10.1109/MLISE62164.2024.10674497
6 E. Barnefske and H. Sternberg: IEEE Access **11** (2023) 3826. https://doi.org/10.1109/ACCESS.2022.3233411
7 H. Yang, W. Cui, and J. Zhang: Proc. 2024 3rd Int. Conf. Artificial Intelligence, Computing and Information Technology (IEEE, 2024) 1–4. https://doi.org/10.1109/AICIT62434.2024.10730111
8 W. Zhou, J. Lu, and W. Yue: Proc. 2019 Chinese Control and Decision Conf. (IEEE, 2019) 803–808. https://doi.org/10.1109/CCDC.2019.8832574
9 E. Casas, L. Ramos, C. Romero, and F. Rivas-Echeverría: Array **22** (2024) 100351. https://doi.org/10.1016/j.array.2024.100351
10 E. Shreyas, M. H. Sheth, and Mohana: Proc. 2021 Int. Conf. Recent Trends in Electronics, Information & Communication Technology. (IEEE, 2021) 735–738. https://doi.org/10.1109/RTEICT52294.2021.9573964
11 F. Xu, J. Chen, Y. Shi, T. Ruan, Q. Wu, and X. Zhang: Inf. Sci. **662** (2024) 120272. https://doi.org/10.1016/j.ins.2024.120272
12 W. Shi and R. Rajkumar: arXiv preprint (2020). https://doi.org/10.48550/arXiv.2003.01251

13 L. Deng, T. Guo, H. Wang, Z. Chi, Z. Wu, and R. Yuan: OCEANS 2022, Hampton Roads (2022) 1–8. https://doi.org/10.1109/OCEANS47191.2022.9977340
14 J. Hong, K. Kim, and H. Lee: IEEE Access **8** (2020) 190529. https://doi.org/10.1109/ACCESS.2020.3023423
15 S. Qi, X. Ning, G. Yang, L. Zhang, P. Long, W. Cai, and W. Li: Displays **69** (2021) 102053. https://doi.org/10.1016/j.displa.2021.102053
16 A. A. M. Muzahid, H. Han, Y. Zhang, D. Li, Y. Zhang, J. Jamshid, and F. Sohel: Neurocomputing **608** (2024) 128436. https://doi.org/10.1016/j.neucom.2024.128436
17 J. Hou and C. Zhang: Heliyon **10** (2024) e31029. https://doi.org/10.1016/j.heliyon.2024.e31029
18 U. Aulia, I. Hasanuddin, M. Dirhamsyah, and N. Nasaruddin: Heliyon **10** (2024) e35247. https://doi.org/10.1016/j.heliyon.2024.e35247
19 D. H. Dos Reis, D. Welfer, M. A. De Souza Leite Cuadros, and D. F. T. Gamarra: Appl. Artif. Intell. **33** (2019) 1290. https://doi.org/10.1080/08839514.2019.1684778
20 D. Chikurtev, N. Chivarov, S. Chivarov, and A. Chikurteva: IFAC-PapersOnLine **54** (2021) 351. https://doi.org/10.1016/j.ifacol.2021.10.472
21 G. Baatar, T. Pfützenreuter, D. Karimanzira, D. Shea, and J. Albiez: Proc. OCEANS 2021: San Diego – Porto (IEEE, 2021) 1–5. https://doi.org/10.23919/OCEANS44145.2021.9705994
22 Y. Chang, Y. Zheng, and Z. Ku: Sens. Mater. **37** (2025) 1657. https://doi.org/10.18494/SAM5420
23 H. Zhang, T. Lo, W. Zhang, H. S. Chen, J. Xiong, C. Yu, and T. Lan: Sens. Mater. **37** (2025) 2013. https://doi.org/10.18494/SAM5469
24 L. A. Rakhsith, K. S. Anusha, B. E. Karthik, D. A. Nithish, and V. K. Kumar: Proc. 2021 Second Int. Conf. Electronics and Sustainable Communication Systems (IEEE, 2021) 1619–1626. https://doi.org/10.1109/ICESC51422.2021.9532809
25 D. Pavani, A. N. N. Reddy, and N. Saw: Proc. 2023 IEEE Technology & Engineering Management Conf. - Asia Pacific. (IEEE, 2023) 1–5. https://doi.org/10.1109/TEMSCON-ASPAC59527.2023.10531302
26 Pipe and Pillar Dataset (Version 8): https://universe.roboflow.com/nellielly/pipe-and-pillar/dataset/8 (accessed March 2025).
27 Mud Seg3 ZYZ13 Dataset (Version 2): https://universe.roboflow.com/nellielly/mud-seg3-zyz13/dataset/2 (accessed March 2025).
28 Ultralytics, YOLOv5: https://docs.ultralytics.com/models/yolo5/ (accessed February 2025).
29 Explore Ultralytics YOLOv8: https://docs.ultralytics.com/models/yolo8/ (accessed February 2025).
30 Ultralytics YOLO11: https://docs.ultralytics.com/models/yolo11/ (accessed February 2025).
31 Ultralytics YOLO12: https://docs.ultralytics.com/models/yolo12/ (accessed February 2025).
32 C. R. Qi, H. Su, K. Mo, and L. J. Guibas: arXiv preprint (2017). https://doi.org/10.48550/arXiv.1612.00593
33 C. R. Qi, L. Yi, H. Su, and L. J. Guibas: arXiv preprint (2017). https://doi.org/10.48550/arXiv.1706.02413
34 Z. Liu, H. Tang, Y. Lin and S. Han: arXiv preprint (2019). https://doi.org/10.48550/arXiv.1907.03739
35 Q. Hu, Y. Yang, and X. Zhang: arXiv preprint (2020). https://doi.org/10.48550/arXiv.1911.11236