Development of time-series point cloud data changes and automatic structure recognition system using Unreal Engine

Toru Kato[1], Hiroki Takahashi[1], Meguru Yamashita[1], Akio Doi[1], Takashi Imabuchi[2]

[1] Iwate Prefectural University, Japan [2] Japan Atomic Energy Agency, Japan

Abstract: We have researched and developed a point cloud processing system on the Unreal Engine that recognizes changes between large time-series point cloud data measured by a laser scanner and performs structural data extraction.

When associating time-series point cloud data with structural information (pipes, tanks, etc.) of each point cloud, CAD data (structural data) is currently created interactively by humans. Unreal Engine is a game engine that excels in visualization of 3D information and is suitable for checking updated data and automating procedures. We developed a user interface that automatically performs a series of update procedures at the touch of a button, and evaluated the effectiveness of the interface.

Keywords: Unreal Engine, Point Cloud, Semantic Segmentation, Visualization.

1. INTRODUCTION

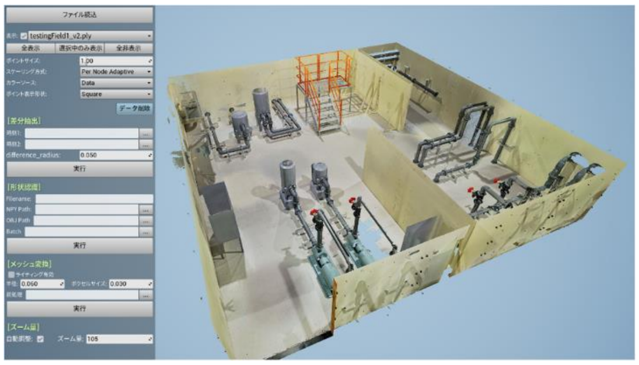
We have developed an automated point cloud processing system on Unreal Engine that detects changes over time in large point cloud data measured by a laser scanner and extracts structural data. In order to associate time-series point cloud data with structural information, such as that of pipes or tanks on a residential property, CAD data (structural data) is currently created interactively for individual point clouds by human operators.

However, the larger the residential property, the more time consuming it is to associate point cloud data with structural information. With each change in the structure of the facility, manual updating is required. Consequently, there are cases where it takes several months to half a year for a single facility to have its structures identified highlighting the current need for improvements in time reduction and operational efficiency. Additionally, the association of point cloud data with structural information is gaining attention as a method for recreating and preserving real spaces digitally.

In this context, we have attempted to automate and streamline the updating process by developing a prototype mechanism that automatically recognizes changes in point cloud data and reflects them in the structural data.

2. SYSTEM METHOD

We developed an interactive system on Unreal Engine for the extraction of differential information from time-series data, shape recognition of structures based on point cloud data, and interactive updating of 3D models. Unreal Engine is primarily a game engine designed for game development, and it is provided by the company Epic Games, known for developing games like Fortnite. The engine is available for free, and it enables the creation of high-quality 3D virtual worlds for various purposes beyond gaming, including VR, architecture, and television and film production (Fig. 1).



表示: ✓ testingField1_v2.ply
ポイントサイズ: 1.00
スケーリング方式: Per Node Adaptive
カラーソース: Data
ポイント表示形状: Square
時刻1:
時刻2:
difference_radius: 0.050
Filename:
NPY Path:
OBJ Path:
Batch:
ライティング有効
半径: 0.060 ボクセルサイズ: 0.030
前処理:
自動調整: ✓ ズーム量: 105

Fig. 1 Unreal Engine Viewer

The differential extraction for time-series point cloud data was implemented using the k-d tree method, a spatial partitioning technique, and was validated with extensive test data. This method confirms the detectability of point cloud data for components that have been moved, deleted, or added. To accurately recognize the shape of the differentially extracted point cloud data, a high-precision deep learning dataset was constructed using both point cloud data and CAD data. Partial scan data, obtained by laser measurement from arbitrary positions, including structural data with missing pipes, was used as training data.

The input information consists of 3D point clouds, 3D CAD data, and measurement point information. Partial scan data can be generated from arbitrarily defined measurement points, and it was utilized as training data for PointNet++ [1].

Regarding the automatic recognition and accuracy assessment of point cloud data through deep learning, point cloud data automatic recognition was performed using PointNet++. In the case of nine categories (duct, pipe, electrical equipment, auxiliary materials, etc.), high recognition accuracy was achieved, with an average Intersection over Union (IoU) of 95.2%.
The dataset used for training comprised 6,644 instances with a total of 77,124,436 vertices. The IoU values ranged from 0.85 to 0.97 (85% to 97%). The trained model was then utilized for the structural recognition of differential point cloud data.

Regarding the time-series data update, adding and deleting the surrounding meshes of differential point clouds is made possible using Open3D functions and labels based on shape recognition. After the update, deleted differential point clouds are excluded from the mesh loading target by removing the corresponding surrounding meshes. Added differential point clouds are processed through labeling, DBScan (grouping of distant differential point clouds), and voting (uniform labeling based on the most frequent label within each group). Subsequently, Open3D's BPA (Ball Pivoting Algorithm) is employed to mesh the labeled point clouds [2] [3].
By adding the resulting mesh to the loading target with the corresponding label name, the system can effectively handle the update.

3. EVALUATION

Using point cloud data obtained from 3D laser scanner measurements we created point cloud data for the initial state (t0) and the changed state (t1) (Fig. 2, Fig. 3). Next, we conducted the extraction of cut-out point cloud data using differential calculation on the two point cloud datasets, t0 and t1 (Fig. 4).
After performing differential extraction, if the distance between one vertex in t0 and another vertex in t1 is within a specified error threshold, they are considered the same vertex, and that vertex is judged to contain no differences. By specifying an error threshold of 0.2 m, considering slight positional discrepancies during measurement, we obtained the differential point cloud data for the removed pipes and tanks.

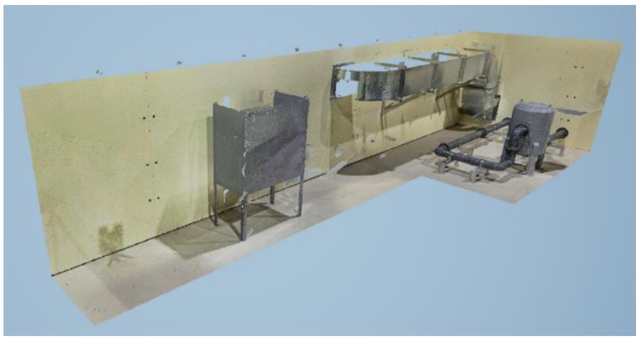


Fig. 2 Measurement results at time t_0 (initial state)

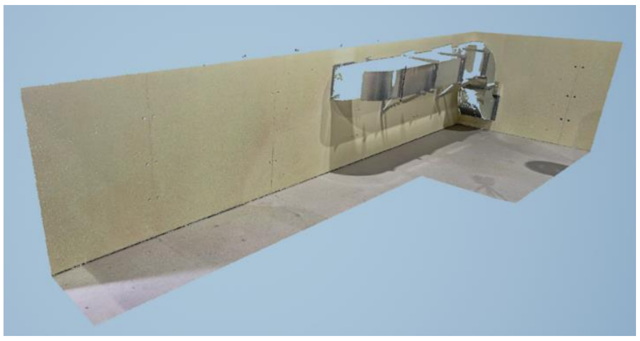


Fig. 3 Measurement results at time t_1 (changed state)

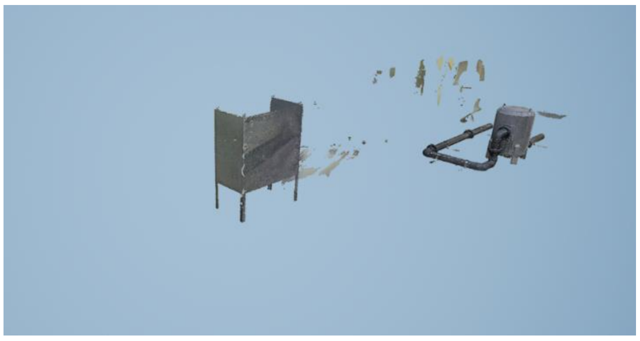
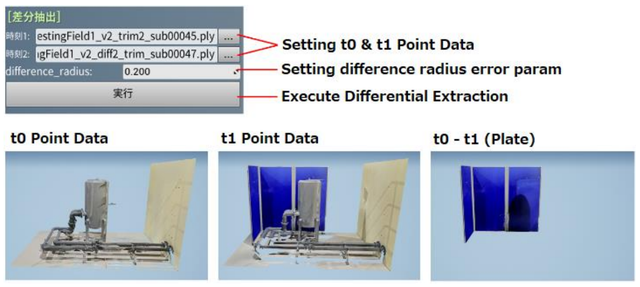


Fig. 4 t_0-t_1 Differential extraction (error threshold 0.2 m)

Below are some examples of using the developed system. The user first specifies the point cloud data for initial state t_0 and the changed state t_1, along with an error threshold, then presses the execute button to automatically perform differential extraction and display the results (Fig. 5, Fig. 6).


[差分抽出]
時刻1: estingField1_v2_trim2_sub00045.ply
時刻2: ıgField1_v2_diff2_trim_sub00047.ply
difference_radius: 0.200
実行
Setting t0 & t1 Point Data
Setting difference radius error param
Execute Differential Extraction
t0 Point Data
t1 Point Data
t0 - t1 (Plate)
Fig. 5 Differential extraction example 1: plates added

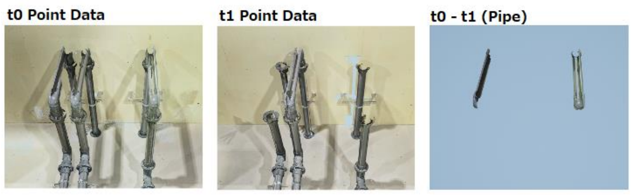
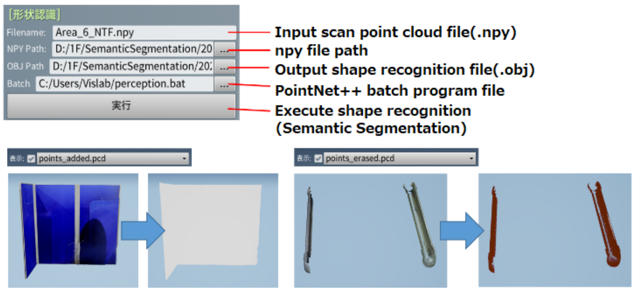


Fig. 6 Differential extraction example 2: pipes removed

Shape recognition (semantic segmentation) using deep learning is then performed on the extracted differences, such as those shown in Figs. 5 and 6. In the examples, shape recognition is performed for the added shielding plates and the removed piping (Fig. 7).


[形状認識]
Filename: Area_6_NTF.npy
NPY Path: D:/1F/SemanticSegmentation/20
OBJ Path D:/1F/SemanticSegmentation/20:
Batch C:/Users/Vislab/perception.bat
実行

Input scan point cloud file(.npy)
npy file path
Output shape recognition file(.obj)
PointNet++ batch program file
Execute shape recognition (Semantic Segmentation)

表示: points_added.pcd
表示: points_erased.pcd

Fig. 7 Shape recognition of difference data

Figs. 8 and 9 provide visual examples of the updated meshes, highlighting the meshes targeted for removal or addition. For clarity, the added elements are colored red using the BPA mesh, while the removed elements are colored blue on the existing mesh for visualization purposes.

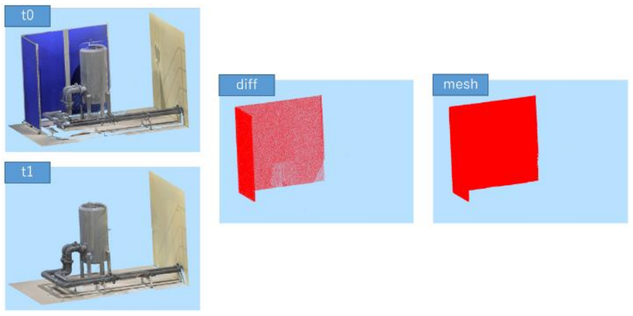


Fig. 8 Updated mesh data example 1: plate added

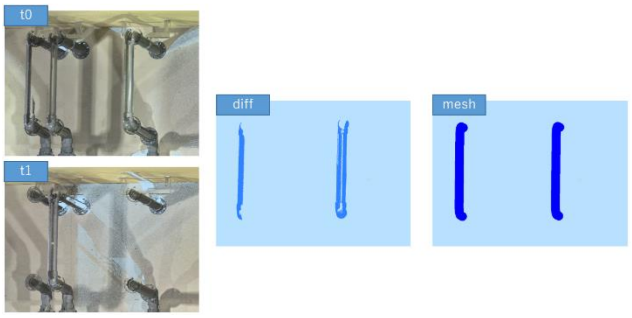


Fig. 9: Updated mesh data example 2: pipe removed

Fig. 10 demonstrates the loading and display of the updated mesh results in Unreal Engine. It demonstrates that the mesh data reflects the addition of the shielding plates and the removal of the pipes. Enabling lighting allows for shaded representation of the mesh.

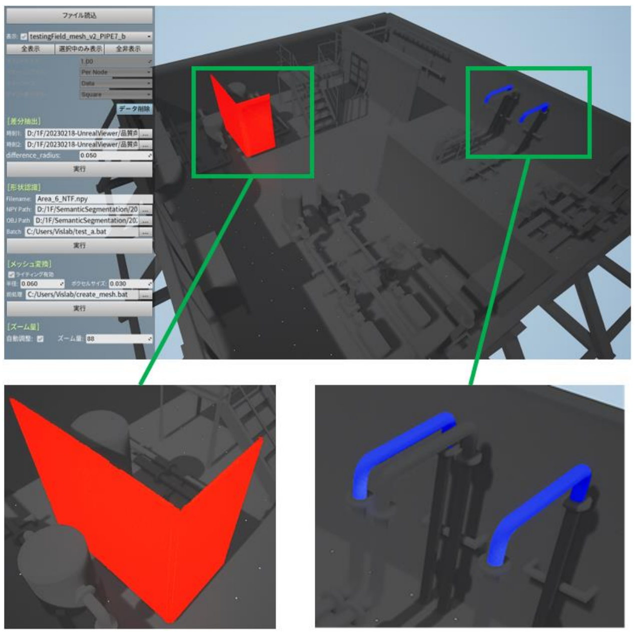


Fig. 10 Updated mesh display (top: overall view, bottom left: plate added, bottom right: pipe removed)

The computational environment for each process is summarized in Table 1.

Table 1 Computational environment

CPU	Intel® Core™ i7-11700
RAM	64GB (8GB×8)
GPU	NVIDIA QUADRO RTX 3060Ti
VRAM	8GB×1
OS	Windows 10

The processing speed of the difference extraction was measured for both the original point cloud data (Case A) and the downsized point cloud data (Case B), and the processing time increased with the number of vertices at t0 and t1. Shape recognition took approximately 1 to 2 minutes for Case A and 20 to 30 seconds for Case B. For meshing, BPA was used for the downsized point cloud data (preferred in terms of suitability) and the processing time for Case B was measured, as shown in Figure 8. The time required for meshing is dominated by the BPA of the added difference, and the smaller the number of difference vertices, the faster the meshing. Therefore, in Figure 10, the results from Case B are used to create and display the updated mesh. Comparing the results listed in Table 2, it is most efficient to store the base (high quality) point cloud data before and after the update, generate a reduced amount of data (about 1/10) just before performing the difference extraction, and use the reduced data for the mesh update.

4. CONCLUSION

In this system, we successfully performed differential extraction, shape recognition, and meshing of data before and after changes in point cloud data to effectively visualize structural changes. In the differential extraction, we obtained only the differential point cloud data that was added or removed between the initial and changed point cloud datasets, and confirmed the removal of some pipes and the addition of shielding plates in the examples. The calculation of differences utilized the k-d tree method, a spatial partitioning technique, allowing the extraction of data with distance errors by comparing point cloud data before and after changes.

Next, for the shape recognition of differential point clouds, we prepared pre-associated training data for point clouds and structures such as pipes and tanks. The learning and recognition processes were carried out using PointNet++, a deep learning network that excels in point cloud recognition. The system was trained to classify structures into nine categories, enhancing recognition accuracy for each type of structure.

Among the recognized point clouds, the point clouds corresponding to added structures were meshed using the BPA, confirming its effectiveness. BPA is widely used for meshing point cloud data and proved suitable for the laser-scanned point cloud data used in this study.

Finally, for the pre-updated structural data, we removed pipes marked for deletion and added mesh data for shielding plates. The updated structural data were created and visualized. The updated data were colored red for added data and blue for removed data to facilitate visual recognition of the updated areas. Regarding computation time, the processing time for differential extraction increased proportionally with the number of point cloud data points. However, downsampling to approximately one-tenth before differential extraction significantly reduced the automatic update time, including shape recognition and meshing, to about one-tenth of the original time. This ensured accurate visualization of the updated data. Therefore, downsampling the differential point cloud data in advance proved to be an effective strategy.

ACKNOWLEDGEMENT

This research is part of the results of a project subsidized by the Ministry of Economy, Trade and Industry (METI) under the "Subsidy for Decommissioning and Contaminated Water Countermeasures Project (Development of technologies for improving the environment inside nuclear reactor buildings (development of digitization technologies for environment and source distribution to reduce exposure))" that started in FY2021.

REFERENCES

[1] C. R. Qi, et al., “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space,” Proc. of the 31st Int’l Conf. on NeurIPS, pp. 5105-5114, 2017.

[2] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, “The Ball-Pivoting Algorithm for Surface Reconstruction”, IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 4, 1999.

[3] A. Doi, et al., "Development of Digitalization Techniques of Environment and Source Distribution for Radiation Exposure Reduction (10) Recognition of Environmental Data and 3D Modeling by Deep Learning," Atomic Energy Society of Japan, 2022.

Table 2 Processing times for original and downsampled datasets

Data Case	Pattern	Sampling	Data Vertices	Processing Time		
				Diff.	Recognition	Meshing
Case A Base Data	Added Plate	1/1	[t0] 6,033,594 [t1] 7,662,734	3h18min	60sec	More than 12hours
	Removed Pipe		[t0] 35,679,109 [t1] 34,299,227	9h38min	120sec	
Case B Down Sample Data	Added Plate	About 1/10	[t0] 635,633 [t1] 746,744	45min	20sec	47sec
	Removed Pipe		[t0] 3,000,407 [t1] 2,858,528	2h30min	30sec	