2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

A FOUNDATION MODEL APPROACH: GENERATING POINT CLOUDS FOR SCAN-TO-BIM TRAINING AND BIM MODEL GENERATION

Nils Soeren Krause[1] and Joern Ploennig[1]
[1]University of Rostock, Justus-von-Liebig-Weg 2, 18059 Rostock, Germany
AI for Sustainable Construction

Abstract

Modern scan-to-BIM approaches often use machine learning (ML) models to reconstruct BIM models from point clouds. Their performance relies heavily on a large amount of training data. Generating this data requires a significant amount of manual effort for outlier elimination and labelling. The approach uses a foundation model (FM) for the generation of point cloud training data. The presented generator enables the training of AI algorithms for the scan-to-BIM method without additional hardware. It also allows future researchers to generate data for a scan-to-BIM workflow.

Introduction

A large number of algorithms and programs exist for the post-processing of point clouds including several machine learning (ML) approaches (Mirzaei et al., 2022). These ML models require a large amount of well labeled, high quality training data (Roh et al., 2019). Traditionally, training data is manually prepared by cleaning real-world measurements and annotating them with semantic labels. This is a large manual effort that scales to provide required training data sets. However, synthetically generated datasets offer a scalable alternative but they are strongly restricted to the variance embedded in the generation of code. This paper explores how foundation models (FMs) such as GPT-4 can overcome these limitations by generating synthetic point clouds. Large vision models like DALL-E 2, MidJourney, and Stable Diffusion, as well as large language models like ChatGPT and GPT-4, have already shown great promise in generative tasks like making architectural images in Paananen et al. (2024), generating floor plans in Ploennigs and Berger (2024), or assisting with for building design in Zhong et al. (2024). This raises the question can these models also generate synthetic 3D data, specifically point clouds, in a similarly effective manner? Synthetic point cloud data is primarily needed due to the requirements of training data for machine learning when processing big data. The development of powerful models requires large, high-quality data sets, which need to be comprehensively labeled and semantically segmented, as shown in Xie et al. (2020). Although algorithms already exist that can automate these preparatory tasks, they often require previously learned knowledge, which makes manual or semi-automated methods necessary. This work proposes the potential for generating point clouds with FMs. The paper starts with a state-of-the-art chapter that presents the relevant technologies in point cloud analysis and synthetic data generation. Following this chapter, the paper presents, an approach for generating geometric shapes of point clouds using foundation models. The approach is starting with an exploration and validation of simple shapes and extending it to buildings with various complex elements like windows. Subsequently, the paper explores the generation of synthetic point cloud data from generated floor plans. With the evaluated data, the paper presents a point cloud generator for a specific object. The generator should output variably shaped objects and produce high-quality and labeled point cloud data. The generated synthetic data is free of outliers, which reduces one step in the post-processing of point cloud data. These will save significant time in the workflow for the post-processing of point clouds. This work concludes with a comparison of noise types in real point clouds and synthetic data, as well as an evaluation of the generator against a point cloud reconstructed from a BIM object. The conclusion summarizes the findings and discusses potential approaches for future developments. This approach will generate variable data, which is suitable for any use case. This method has the potential to replace manual measurement processes for generating training data. Synthetic datasets could be produced by prompting a FM model with textual input.

State-of-the-Art

Since the introduction of terrestrial laser scanners by Fröhlich et al. (2004) it is no longer a technical challenge these days to collect point clouds from the real environment. In recent years, low-cost mobile alternatives, such as the iPad Pro in Krause and Guzmán-Merino (2024), have also become available. There are some problems with real life data, when it comes to the post processing of point clouds, the huge amounts of data collected by scanner demand a lot of manual work to get rid of outliers. We need to find a way to make point clouds simpler and find structures. Various algorithms exist that support those tasks: Examples of these methods include RANSAC for outlier elimination in Quan and Yang (2020), or the reconstruction of BIM mod-

els, also known as scan-to-BIM in the papers by Tang et al. (2010), Rocha et al. (2020), and Justo et al. (2021). Automating these methods is possible by using models from the field of AI and machine learning. These models can process large amounts of data quickly and efficiently, for example, as shown in the work from Nguyen et al. (2019), for large-scale data mining. Fan et al. (2017) suggest that the reconstruction process can also benefit from the fusion of point clouds with additional sources like images. A central challenge in post-processing for point cloud data is the segmentation of individual objects in the point cloud. These can be carried out using models, for example, PointNet or PointNet++ Qi et al. (2017). Sarode et al. (2019) demonstrate the constant development of the models to enhance their precision and efficiency. The performance of the models depends heavily on specific parameters that need to be adapted to the respective applications. They also require extensive data sets for training or testing of these models, as shown by Qian et al. (2022). The data sets for training and testing can consist of real data or be generated synthetically. A real data set is, for example, the Sydney Urban Objects from 2013 with over 588 examples divided into 14 classes, published in De Deuge et al. (2013) or the data set named ScanObjectNN from 2019 with 2902 examples, presented in Uy et al. (2019). Another example of synthetic data is the data set ModelNet40-C from 2015, which contains 185.000 point clouds as data. The publication years of these data sets show that the method of synthetic point cloud data is not new. However, with the development of fast-processing AI models such as ChatGPT-4, Gemini, or DeepSeek, we have numerous new opportunities to discover more effective solutions. In conjunction with the size of these data sets, it illustrates the high demands placed on the amount of data to train models effectively. As well as that, there are high standards for the quality of the data. For example, algorithms like RANSAC shown in Li et al. (2023) can be used to reduce noise in the point cloud post-processing step. Traditional methods for processing point cloud data, such as the use of the RANSAC algorithm, are considered established approaches. In the current state of the art, modern methods are becoming more important, especially large language models (LLMs). For example, Hong et al. (2023) present 3D-LLM, a language model for 3D understanding that is able to process point cloud data and solve complex 3D tasks such as captioning, question answering, navigation and dialog guidance using targeted prompting strategies. Xu et al. (2024)) present another model, PointLLM, which combines LLMs with point cloud and point cloud text data and is used for generative tasks such as 3D object classification and description. Similarly, Yang et al. (2025)) with LiDAR-LLM use this combination to process LiDAR data. Further applications can be found in the field of point cloud registration with MiniGPT-3D Tang et al. (2024) and segmentation in SegPoint He et al. (2024). The literature makes it clear that the use of LLMs for processing 3D data is currently an interesting topic and a dynamically growing field of research. In this paper, however, we do not focus on the model-based processing of point cloud data, but on the upstream process of capturing and creating such data using foundation models (FMs). Synthetic point clouds are not a new topic, and they have already been used in the training of models for the scan-to-BIM method or labeling process shown by Ma et al. (2020). However, the creation of data is mostly based on modern approaches such as text-to-image diffusion models, which find their way to synthetic data in Nichol et al. (2022); Vahdat et al. (2022). The innovative approach in this paper is the step back to a text-to-code-to-cloud method, in which synthetic point clouds are generated through the automated adaptation and extension of Python code. Unlike other approaches that generate synthetic datasets, this method does not rely on a black box model from direct text to cloud. The prompted code for generation is editable and can be used for the pretraining of the FM for cloud generation. This allows greater flexibility and easier scaling for objects, of outlier free point clouds.

Data generation and Results

Foundation models are large visual or language models (LVM/LLM) that can analyze and generate content. A prominent example of a foundation model is ChatGPT-4, which is based on GPT-4 Achiam et al. (2023). ChatGPT is a large language model that can generate textual output based on input prompts and, in the recent version, also images using the Dall-E 2 LVM model. Despite these multi modal features, GPT cannot directly generate point clouds because they typically exist in binary, or non-textual, formats. There are various libraries, such as PyVista, Open3D, PyntCloud, or Laspy, that can process and create point clouds. However, using these libraries requires a certain amount of domain knowledge and increased manual effort in programming. To overcome these intrinsic limitations, this paper proposes a bypass strategy. It is well known that GPT models are good at generating codes Poldrack et al. (2023). But this paper takes advantage to inquire about how ChatGPT-4 generates the necessary point cloud data in Python code. The code is exported and executed to generate files with the point cloud data. By integrating loops and other mechanisms, the creation of an almost unlimited amount of high-quality data is possible. This method will generate high-quality data, which can be used for training and validating AI models. This approach opens promising prospects for the further optimization of the scan-to-BIM method and for pre- or post-processing with point cloud data in various application areas. In the following section, we dive into analyzing the generated data.

Simple geometries and room shapes

First, some basic geometric shapes were analyzed to verify the approach, including cones, cubes, cuboids, cylinders, prisms, pyramids, and spheres. The results show that these shapes could be generated by ChatGPT-4 without major

effort. The generated point cloud data was successfully exported in .ply format and could be visualized in suitable software. Figure 1 shows the resulting point clouds of the generated geometric shapes. The generated basic shapes worked well, but it failed for more complex forms like the prism and pyramid. To answer the, question of whether

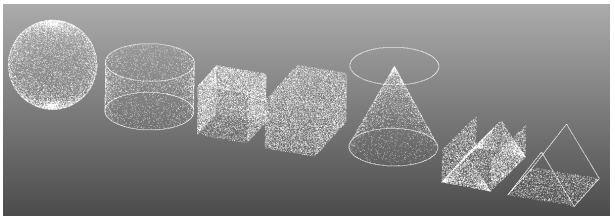

Figure 1: Geometric shapes generated by ChatGPT-4 from left to right: Sphere, cylinder, cube, cuboid, cone, prism, pyramid.

this suffices for room shapes, the approach explores different room shapes. The different generated shapes were shown in Table 1. Again, the approach works well for simple shapes but fails for more complex ones.

Table 1: Test of the generation of different room shapes.

Shape	Result	Description
Rectangle	succeeded	Rectangular
L-Shape	succeeded	Two rectangles
U-Shape	failed	Two walls
Z-Shape	failed	Two walls
T-Shape	failed	Cubic sample
H-Shape	failed	Three rectangles
Rectangle, carved wall	failed	Rectangle, cylinder
Rectangle, sloping ceiling	failed	Rectangle

Multiple rooms

In the next step, the paper investigated the capability of the LLM to create multiple rooms. A simple rectangle was used as a room template for this purpose, as shown in Figure 2. With the prompt: *Create the point cloud of a room; rectangular; length 3 m, width 5 m, height 2 m*, ChatGPT was able to fill this rectangle with additional points (Left in Figure 3) and leave only those points belonging to the wall (Right in Figure 3), with the next prompt (*Only points on the walls*), the latter is closer to real laser scans of buildings or objects. Buildings do not only

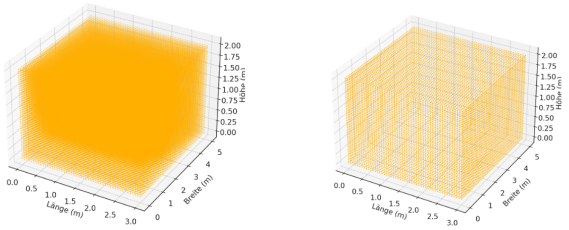

Figure 2: (Left) Point cloud of a filled cube; (Right) Point cloud with only surface points on the cube walls.

consist of individual rooms. To illustrate the point cloud closer to reality, the rectangular rooms were expanded to include multiple rooms spanning up to two floors. Therefore, the instruction (*Make four rooms with a distance of 36.5 cm between the walls*) was given to generate individual rooms arranged with a defined spacing between them. Figure 3 (Left) shows the results for four rooms with an equal size. The point cloud arranges all rooms in the same plane area. Figure 3 (Right) investigates how the model handles additional floors. We asked (*Generate a single large room above the rooms, distance to the upper edge of the lower rooms 30 cm*) for an additional floor with a single room that takes up the entire width of the level below. The important thing is that the FM correctly aligned the outer walls.

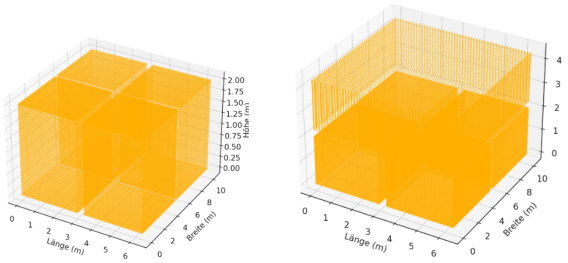

Figure 3: (Left) Point cloud of multiple rooms; (Right) Point cloud with another room on a top floor.

Adding Windows, Doors, and Furniture

In addition to the basic structure of a building, it is also important to include other components such as windows. In the next step of the evaluation process, a request (*Insert windows on the outer sides*) was made to integrate windows into the point cloud. These experiments only were only partially successful. The LLM added windows to the resulting point cloud, representing them as flat, rectangular areas without depth. Figure 4 (Left) depicts them as red lines. On the upper level, the window placement was quite good, and the windows were positioned on the outer walls. On the lower level, the arrangement was less precise, and windows were placed on every generated wall, including interior walls. However, the upper room only features external walls, making it impossible for the model to differentiate between outer and inner walls. In that case, the representation of the windows does not correspond to their typical appearance in real laser-scanned point clouds. Analyzing the generation of doors (*Add doors between the rooms*) reveals a similar result. Trhe FM represented the doors as flat, rectangular objects, colored in blue in Figure 4 (Right). Both examples show that while the model is capable of generating those objects, it still lacks deeper knowledge of proper arrangement and related spatial constraints like depth and object connections. Next, prompt is (*can you furnish the rooms inside?*) to the point cloud. Figure 5 (Left) shows the results. Furniture was placed on the lower level to avoid any overlap with other objects, such as walls. The arrangement adhered to the layout of the lower room, with the furniture represented as simple rectangular geometric shapes. For instance, the LLM placed a desk and a bed in each room. An intriguing aspect of this step is the model's apparent ability to distinguish between

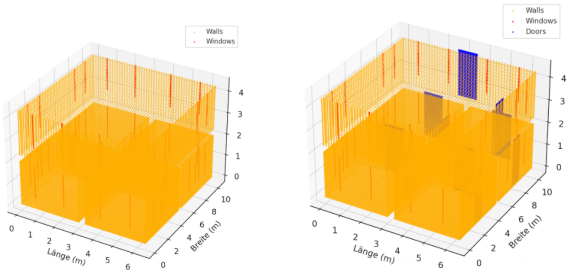

Figure 4: (Left) Point cloud of a room with windows in red lines; (Right) Point cloud of a room with doors in blue rectangles.

the upper and lower levels within a point cloud. The request specifically targeted the lower level, and the model adhered to this instruction. To complete the building with an outdoor scan, the LLM was prompted (*Make an outer wall around the outside of the room's distance to the inner wall 36.5 cm*) to add exterior walls. The prompt specified a specific distance from the interior walls to ensure that the exterior wall completely encloses the rooms. The exterior wall is visualized in Figure 5 (Right) as a grey point cloud.

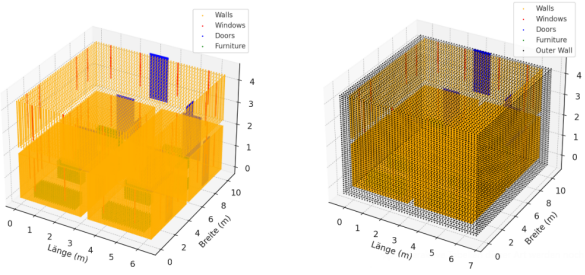

Figure 5: (Left) Point cloud of a room with furniture on lower floor; (Right) Point cloud of a room with a outer facade.

Adding a roof

The model required the addition of a roof to complete the building. For this purpose, the FM was prompted (*Make a saddle roof on the house*) to generate a common roof shape, in addition to a saddle roof. However, the FM initially generated only a rectangular shape on a new level above the building, as indicated by the red dots in Figure 6 (Left). To address this issue, a new prompt (*Try a different roof shape, the roof must be closed at the top*) was tested without a shape constraint to give more flexibility in the generation process. Under these conditions, the FM generated a roof-like upside-down construction in Figure 6 (Right). Further adjustments and additional commands were allowed to alter the element arrangement. With these modifications, the model was ultimately able to correct the shape and achieve the desired result.

Design Variations

In the next step of the investigation, a prompt (*to make variations to the building in terms of rooms and roof*

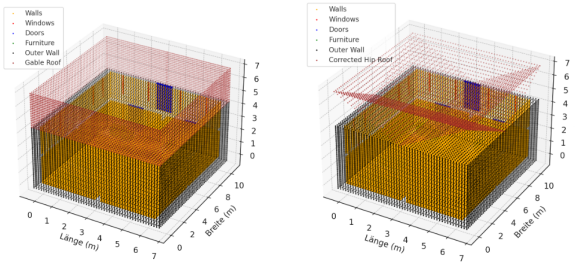

Figure 6: (Left) Point cloud optimization with a first try of a roof; (Right) Example of a variation without input instructions after generating the first point cloud example, of an roof.

shapes) was designed to evaluate the AI's ability to produce variants of the models, as they are very important for model training. The FM was instructed to alter the spatial arrangement of elements in different planes and to vary geometric shapes. For example, various geometric shapes, such as circular or triangular structures, were added to the generated spaces. However, the results of these variations were suboptimal. Although the FM generated different geometrical shapes, the rooms and objects it produced were no longer interconnected. Figure 7 (Left) provides an example of this variant. The process was repeated multiple times with similar results. But, by down-selecting prompts that worked, it was possible to add additional elements, such as foundations, balconies, and columns. This final result is the point cloud of the multi story building shown in Figure 7 (Right) which is parameterisable in the generated code.

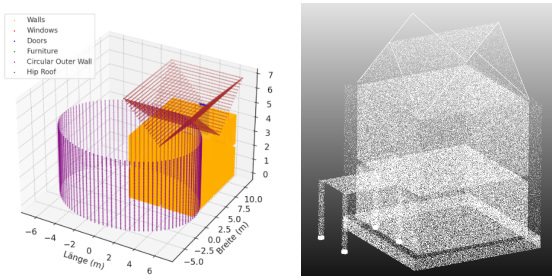

Figure 7: (Left) Point cloud with variations prompt; (Right) Example of a complex building, with foundation, balcony, supports, and a corrected roof.

Templated Floor Plan Generation

However, a common challenge arises when a specific layout already exists in the form of a 2D floor plan. In such cases, the question is whether the FM can also generate a point cloud based on an existing floor plan. To address this, first a random image of a floor plan was generated in ChatGPT-4, as shown on the left in Figure 8. The image is quite good, with some minor errors in detail. In a previous work (Ploennigs and Berger, 2024) it was shown that it is possible to remove those errors and generate various room shapes by fine-tuning such multi-modal FMs. Next, the FM is instructed to analyze the generated image and create a synthetic point cloud. Figure 8 on the right displays the result. At first glance, it

appears good, but further inspection shows it is actually just a 2D point cloud of the gray scale image, meaning that the 3D effect comes from the original perspective of the image and is not represented in the point cloud. This is also recognizable by the points representing the measurement grid on the image outline. To fix that problem, the following generated code was added, to the input context of the FM, and it was asked for a point cloud. Our workflow for the prompts for this text-to-image-to-code-to-cloud variant is:(i) The FM was prompted with the code snippet def rectangular_profile. (ii) Check to create a floor plan. (iii) The prompts were modified to create multiple rooms in different geometric representations and shapes (Figure 9 Left). (iv) The FM was asked to optimize the floor plan and add more rooms, the rooms should represent the floor plan of a house (Figure 9 Right).

This priming of the FM with its own generated code resulted in a completely different outcome that is visible in Figure 9 right. In this instance, we observe numerous interconnected rooms that adhere to the original image's shapes and layouts. It is intriguing to note that the FM has solved the problem of generating a T-shaped room by teaching it with its own code. This was not possible when testing the room shapes in the first part of this paper, see Table 1. Of course, the resulting room layout does not represent the complexity in the original image, but it may be sufficient for simple single house layouts.

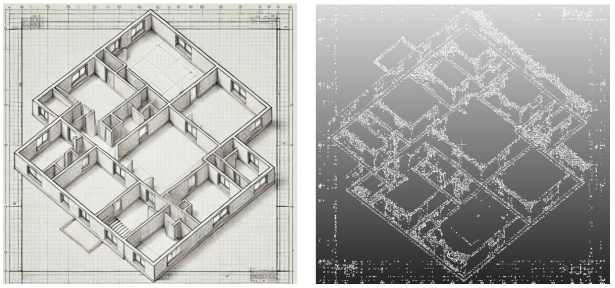

Figure 8: (Left) Generated floor plan image; (Right) Derived Point Cloud.

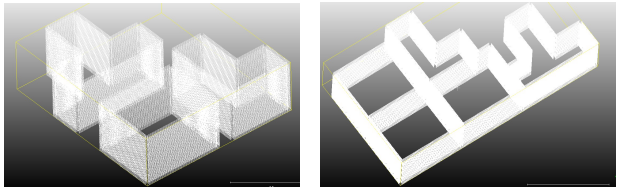

Figure 9: (Left) First optimization, with the instruction of different shapes; (Right) Second optimization, with the instruction to connect the different shapes and generate them as "floor plan".

Model generation details

Window generator

To make the generation process more comprehensible, the example of a window is used for illustration and examination. The FM was prompted to generate Python code for creating a point cloud representing a window. This code was then adapted to produce a point cloud in .ply format. An important feature of the generated code is that the many parameters, such as the height, width, and depth of the window, are customisable, as shown in Listing 1. These parameters create the base for the geometric variability of the window's dimensions and the density of the point cloud. The point density is a key factor for the visualization, as higher point density results in greater detail and accuracy. Also, for training scan-to-BIM models, it is very important to ensure variability in the training data.

Listing 1: Parameter of window point cloud generator.

```
import numpy as np
# Parameters for the window
width_win = 2.0  # in metres
height_win = 1.5  # in metres
depth_win = 0.30  # in metres
profile_width = 0.10  # in metres
density = 0.01
```

Listing 2 displays the function for creating the point cloud. The function generates a window surface defined in x and z coordinates. The density parameter controls the distribution of points that populate the surface. The points are arranged along parallel lines, creating a grid. This grid corresponds to the typical scanning strategy of a terrestrial laser scanner, where points are systematically captured along a predefined grid. As a result, the synthetic point cloud produced by the function provides a realistic and detailed representation of a scanned area, closely emulating the methods used in practical laser scanning.

Listing 2: Function to generate the window profile.

```
def rectangular_profile(x_start, y_start, x_end,
y_end, z_min, z_max, width_win, density):
    dx = x_end - x_start
    dy = y_end - y_start
    length = np.sqrt(dx**2 + dy**2)
    if length == 0:
        return []
    # direction of the profile
    ux, uy = -dy / length, dx / length
    nx = ux * width_win / 2
    ny = uy * width_win / 2
    # Generate points on line
    num_points = int(length / density)
    points = []
    for i in range(num_points + 1):
        t = i / num_points
        x = x_start + t * dx
        y = y_start + t * dy
        for z in np.linspace(z_min, z_max,
                int((z_max - z_min) / density)):
            points.append([x - nx, y - ny, z])
            points.append([x + nx, y + ny, z])
    return points
```

Listing 3 shows the function that the generator is using to complete the window's surface with areas left and right in the x, y dimensions (i.e., the red lines in Figure 4 (Left)). The definition of the surface position is based on a height of z, which is also the depth of the window.

Listing 3: Function to generate the window profile.

```
def x_y_area(x_min,x_max,y_min,y_max,z,density):
    points = []
    x_rng = np.arange(x_min,x_max+density,density)
    y_rng = np.arange(y_min,y_max+density,density)
    for x in x_rng:
        for y in y_rng:
            points.append([x, y, z])
    return points
```

The FM uses these functions to generate a window, as shown in Figure 10 (Left). The modular functions and the predefined parameters provide a lot of flexibility to customize the resulting model in many aspects. The generated code can be accessed for future development on GitHub [1].

Labeling

An essential part of post-processing point clouds is the labeling process. The FM's ability to support this task is examined using the example of the synthetic window point cloud. The FM was prompted to extend the window generation code by adding labels that assign class 1 (red) to the window profile and class 0 (blue) to the windowpane. This modification resulted in the labeling function in Listing 4. This approach ensures that the labeled data is seamlessly integrated into the generation process. The resulting labeled window is shown in Figure 10 (Right).

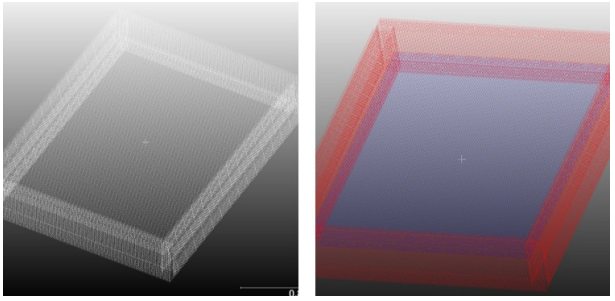

Figure 10: (Left) Generated point cloud of a single window.; (Right) Point cloud with colored labels: red) for the profile and blue) for the window pane.

Listing 4: Function to label the points in the point cloud generator.

```
def label_points(points_win, profile_width):
    labeled_points = []
    for point in points_win:
        x, y, z = point
        if (abs(x) >= width_win / 2 -
                    profile_width / 2 or
            abs(y) >= height_win / 2 -
                    profile_width / 2 or
            abs(z) >= depth_win / 2 -
                    profile_width / 2):
            # Profile
            labeled_points.append([x, y, z, 1])
        else:
            # Pane
            labeled_points.append([x, y, z, 0])
    return np.array(labeled_points)
```

[1] https://github.com/AI4SC/Point_cloud_generator.git

Comparison of synthetic point cloud data

This section presents a comparison between synthetically generated point cloud data of the window and point cloud data derived directly from a 3D BIM model. The reference point cloud data is generated from the 3D BIM model in the following order: (i) conversion of a CAD object into a mesh file, (ii) generation of the point cloud on the surface of the mesh. The entire process can be handled in the software CloudCompare. Figure 11 shows the BIM object on the left and the generated point clouds from Step (ii) on the right.

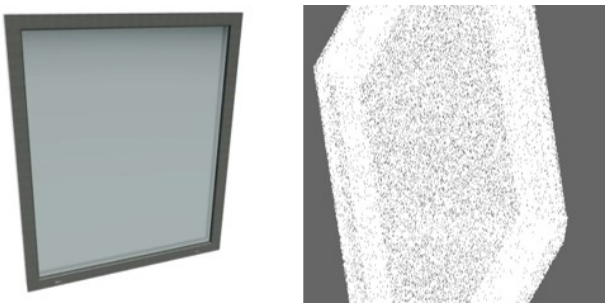

Figure 11: BIM object for the comparison (Left); generated cloud from the mesh (Right).

Figure 12 displays the comparison. It uses a C2C (cloud to cloud) distance calculation to show the relationship between the generated BIM object cloud and the object made from the point cloud generator with the FM model. The deviations in the area of the window frame were predominantly between 0 and 7 mm, shown in the first peak of the graph. The second peak of the graph shows that the windowpane deviations are 1.5 to 2.0 cm. The deviations for the window frame are presumably due to the alignment between the point clouds. Different thicknesses or installation depths in the window frame are increasing the deviations of the windowpane.

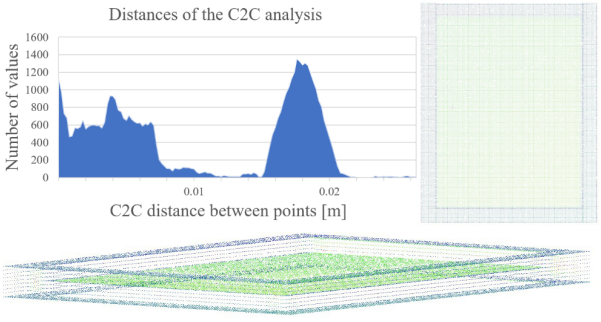

Figure 12: Cloud to cloud comparison, of the synthetic data, shown in a graph (Left) and the colored cloud, blue distances of the window frame, green distances of the window pane (Right, bottom).

Noise in point clouds

Real world point cloud scans are usually full of noise. This noise comes from measurement errors, from the sensors, or physical phenomena like reflections, as well as computational errors from algorithms like SLAM (Simultaneous

localization and mapping). Therefore, it is necessary to reduce the noise with various filter techniques.Figure 13 presents different noise samples for scanning software variants on the iPad Pro 2022. The green lines are the noise of the BLK2GO scanner from Leica, the blue lines are the noises from different applications of the iPad Pro 2022. The application from RTAB-Map and Scaniverse produced more noise than the professional system from Leica, the BLK2GO, but it is interesting to see that the 3DScannerApp produced less noise than the BLK2GO. This demonstrates that the low-cost system is suitable for scanning. In Figure 14 the RTAB-Map shows larger dots in clusters. The Scaniverse application produces linear noise with larger distances between the dots, and the 3DScannerApp produces denser clouds, creating a straight line of individual dots. When considering the quality of the noise now, the Scaniverse application is more suitable than the BLK2GO for modeling building models. Because it's easier to determine the outer or inner edge of the geometry when modeling the building walls. In contrast, the synthetic point cloud data is free of noise.

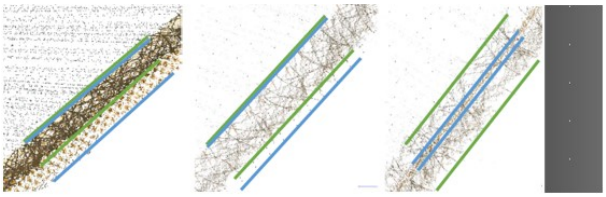

Figure 13: Noise comparing the BLK2GO-Leica with different iPad apps: a) RTAB-map; b) Scaniverse; c) 3DScannerApp; d) Synthetic point cloud.

Conclusion and Discussion

Our investigation provides valuable insight into the potential of generating synthetic point clouds using multi modal foundation models like ChatGPT-4. The quality of the data in terms of outlier elimination is excellent. The synthetic point cloud does not contain any outliers or missing points, and the elements can be automatically labeled. This feature removes the need for outlier elimination and the labeling process in preparing training data. Instead, it adds many means to generate diverse datasets with various parameters to vary size, shapes, and alignment. The comparison of the generated synthetic point cloud data showed promising geometric accuracy. The excellent accuracy shows that the FM is able to generate code for parameterisable synthetic point clouds. However, it is crucial to keep in mind that synthetic objects must be further adapted to the real world. In this method, a complete window with frames and panes was produced. In real data, the interaction of the built in window geometry with the wall needs to be known. This interaction would change the geometry of the synthetic point cloud. Otherwise, the shape can change because of the fact that real measuring systems have problems with reflective surfaces such as the windowpane. Furthermore, the scalability of the approach for synthetic point cloud generators has not been fully explored. The approach has been investigated for isolated geometry such as a window. For complex geometries, such as nonlinear, irregular or free-form objects, there is of course a need for further research. However, these objects also consist of mathematically defined functions, such as: curves, circles, ellipses, splines, Bezier curves or nurbs. The creation of synthetic generators for objects of varying complexity should also work. For this purpose, the mathematical operators would probably have to be provided to the FM model as a basis.

Future Research

In future studies, the topic of the synthetic generation of point clouds will be further investigated, with a particular focus on the creation of point cloud generators using Python. The generator will be extended so that it is possible to generate other components for civil engineering purposes as synthetic point clouds. The ultimate goal of this research is to develop a flexible dataset for training AI algorithms. This research aims to meet the growing demand for digital twins to simulate environmental conditions. Another area of research will be to investigate the usage of FM for analyzing real world data, such as the labeling of laser scans. In the end, a new point-aided design (PAD) system could emerge from the various generators of parameterisable components. The PAD system would function similarly to current Computer Aided Design (CAD) systems, utilizing IFC data. The PAD would work directly with point cloud data, enabling more flexible and precise design processes. In this system, point clouds could also function as parameterisable objects, enhancing the adaptability and functionality of design tools.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. arXiv:2303.08774.

De Deuge, M., Quadros, A., Hung, C., and Douillard, B. (2013). Unsupervised feature learning for classification of outdoor 3d scans. In Australasian conference on robitics and automation, volume 2.

Fan, H., Su, H., and Guibas, L. J. (2017). A point set generation network for 3d object reconstruction from a single image. In conference on computer vision and pattern recognition.

Fröhlich, C., Mettenleiter, M., et al. (2004). Terrestrial laser scanning–new perspectives in 3d surveying. International archives of photogrammetry, remote sensing and spatial information sciences, 36.

He, S., Ding, H., Jiang, X., and Wen, B. (2024). Segpoint: Segment any point cloud via large language model. In European Conference on Computer Vision. Springer.

Hong, Y., Zhen, H., Chen, P., Zheng, S., Du, Y., Chen, Z., and Gan, C. (2023). 3d-llm: Injecting the 3d world into large language models. Advances in Neural Information Processing Systems, 36.

Justo, A., Soilán, M., Sánchez-Rodríguez, A., and Riveiro, B. (2021). Scan-to-bim for the infrastructure domain: Generation of ifc-compliant models of road infrastructure assets and semantics using 3d point cloud data. Automation in Construction, 127.

Krause, N. S. and Guzmán-Merino, M. (2024). Experimental evaluation of ipad pro lidar, for building modeling and possibilities to connect with ai technologies. In 35. Forum Bauinformatik, fbi 2024.

Li, P., Wang, M., Fu, J., and Wang, Y. (2023). Plane detection based on an improved ransac algorithm. In CCAI. IEEE.

Ma, J. W., Czerniawski, T., and Leite, F. (2020). Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic bim-based point clouds. Automation in construction, 113.

Mirzaei, K., Arashpour, M., Asadi, E., Masoumi, H., Bai, Y., and Behnood, A. (2022). 3d point cloud data processing with machine learning for construction and infrastructure applications: A comprehensive review. Advanced Engineering Informatics, 51.

Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., López García, Á., Heredia, I., Malík, P., and Hluchỳ, L. (2019). Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. Artificial Intelligence Review, 52.

Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., and Chen, M. (2022). Point-e: A system for generating 3d point clouds from complex prompts. arXiv:2212.08751.

Paananen, V., Oppenlaender, J., and Visuri, A. (2024). Using text-to-image generation for architectural design ideation. Journal of Architectural Computing, 22.

Ploennigs, J. and Berger, M. (2024). Automating computational design with generative ai. Civil Engineering Design, 6.

Poldrack, R. A., Lu, T., and Beguš, G. (2023). Ai-assisted coding: Experiments with gpt-4. arXiv:2304.13187.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems, 30.

Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., and Ghanem, B. (2022). Pointnext: Revisiting pointnet++ with improved training and scaling strategies. Advances in neural information processing systems, 35.

Quan, S. and Yang, J. (2020). Compatibility-guided sampling consensus for 3-d point cloud registration. IEEE Transactions on Geoscience and Remote Sensing, 58.

Rocha, G., Mateus, L., Fernández, J., and Ferreira, V. (2020). A scan-to-bim methodology applied to heritage buildings. Heritage, 3.

Roh, Y., Heo, G., and Whang, S. E. (2019). A survey on data collection for machine learning: a big data-ai integration perspective. IEEE Transactions on Knowledge and Data Engineering, 33.

Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R. A., Lucey, S., and Choset, H. (2019). Pcrnet: Point cloud registration network using pointnet encoding. arXiv:1908.07906.

Tang, P., Huber, D., Akinci, B., Lipman, R., and Lytle, A. (2010). Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. Automation in construction, 19.

Tang, Y., Han, X., Li, X., Yu, Q., Hao, Y., Hu, L., and Chen, M. (2024). Minigpt-3d: Efficiently aligning 3d point clouds with large language models using 2d priors. In 32 ACM International Conference on Multimedia.

Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, T., and Yeung, S.-K. (2019). Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In CVF international conference on computer vision.

Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K., et al. (2022). Lion: Latent point diffusion models for 3d shape generation. Advances in Neural Information Processing Systems, 35.

Xie, Y., Tian, J., and Zhu, X. X. (2020). Linking points with labels in 3d: A review of point cloud semantic segmentation. IEEE Geoscience and remote sensing magazine, 8(4).

Xu, R., Wang, X., Wang, T., Chen, Y., Pang, J., and Lin, D. (2024). Pointllm: Empowering large language models to understand point clouds. In European Conference on Computer Vision. Springer.

Yang, S., Liu, J., Zhang, R., Pan, M., Guo, Z., Li, X., Chen, Z., Gao, P., Li, H., Guo, Y., et al. (2025). Lidar-llm: Exploring the potential of large language models for 3d lidar understanding. In AAAI Conference on Artificial Intelligence, volume 39.

Zhong, C., Yi'an Shi, L., and Wang, L. (2024). Ai-enhanced performative building design optimisation and exploration. In 29 Conference on Computer-Aided Architectural Design Research, volume 1.