2026 European Conference on Computing in Construction
Corfu, Greece
July 12–15, 2026

SYN3D-LLM: SYNTHETIC 3D POINT CLOUDS GENERATED THROUGH LLM DISCUSSION

Nils Soeren Krause[1], Joern Ploennigs[1]
[1]University of Rostock, Germany
AI for Sustainable Construction

Abstract

Creating synthetic datasets that capture the diversity of real-world sites and measured point clouds remains challenging. The paper introduces SYN3D-LLM, a dialog-driven large language model framework for generating semantically annotated synthetic 3D point clouds by producing executable code through an iterative designer-critic loop. Experimental evaluations of the framework demonstrate that it improves the pass rate by discussing and testing generated code and by assessing the geometrical consistency between the initial prompt and the generated data. The paper demonstrates that SYN3D-LLM is versatile in generating point clouds for various cases.

Introduction

Most Scan-to-BIM, SLAM, or semantic segmentation models in the Architecture, Engineering, Construction, and Operations (AECO) sector are trained on real measured point clouds. This has the benefit of representing real-world conditions and complexities such as noise, occlusions, and varying point densities.

On the other hand, it requires extensive data collection efforts, including scanning diverse environments and manual annotation of point clouds, which can be labor-intensive and time-consuming. Furthermore, survey papers indicate that many Scan-to-BIM approaches are primarily evaluated on proprietary or project-specific datasets, hindering objective comparison within the State-of-the-Art (Abreu et al., 2023). This lack of standardized benchmarks limits reproducibility and comparability of workflows (Xu and Stilla, 2021; Luo et al., 2026) and the development of generalized models.

To address this challenge, researchers are exploring the use of synthetic point clouds (Noichl et al., 2021) as a complement to real measures to introduce a larger variety of data. Synthetic point clouds can provide a reliable foundation for benchmarking, as demonstrated by synthetic datasets such as ModelNet in Wu et al. (2015), which is widely used for object recognition. Studies such as those by Yue et al. (2025) show that synthetic point clouds are effectively augmenting real-world data, leading to improved performance of the trained model. The authors used a diffusion model to generate synthetic data, which, as before, requires training on large amounts of real data. Limiting this is the potential diversity of the generated data. Similar challenges arise in approaches that generate synthetic point clouds from IFC models Zou et al. (2026), which must be modeled and are available in only a limited variety.

To overcome these limitations, scalable and diverse synthetic point cloud generation methods are needed. We present SYN3D-LLM as an approach for generating synthetic point cloud data using large language models (LLMs) that do not require pre-training on real-world point clouds. While SYN3D-LLM does not introduce a new methodology, its contribution lies in integrating LLM-based self-refinement for AECO-specific point cloud generation. Beyond demonstrating that it is possible to generate synthetic point clouds with LLMs, we want to investigate the reliability and accuracy of the generated data in engineering contexts. We therefore focus on two key research questions:

- **RQ1:** How reliably does the proposed SYN3D-LLM framework generate executable point cloud code under iterative designer-critic refinements?
- **RQ2:** How accurately do the generated point clouds comply with the geometric dimensions and semantic constraints specified in the input prompts?

To address those research questions, SYN3D-LLM does not generate point clouds as a one-shot solution, but it uses an iterative framework to critically review and improve the point cloud drafts. To remove the requirement of data for training that generative models like Generative Adversary Networks (GAN) or Diffusion Models (DM) have, the approach relies on the LLMs capability to generate code from natural language prompts. For this SYN3D-LLM introduces a review-loop where the LLM assumes two roles: (i) as a code-generating designer and (ii) a code-reviewing critic. This enables SYN3D-LLM to generate point clouds of a wide complexity across a variety of construction types. The main contributions of this paper are:

Proposing SYN3D-LLM framework: Introduction of a dialog-based self-discussion framework in which an LLM generates synthetic point clouds as a programming designer.

Scalable, labeled data generation: An approach for generating different building classes, and directly annotated point cloud scenes for the AECO sector.

Evaluation of SYN3D-LLM: Investigation of the Pass rate and increasing token cost of iterations. As well as

constraining checks on the input and output information for geometrical consistency, and the semantic labels of the required information from the input prompts.

First, we introduce SYN3D-LLM in the SOTA context and distinguish it from existing work of synthetic point clouds for Scan-to-BIM workflows and semantic segmentation. Next, we describe the design and the functioning of the proposed framework in the methodology section. This is followed by data evaluation, a discussion of the results, an outlook, and a conclusion.

State-of-the-Art

The motivation for synthetic point clouds in the Scan-to-BIM context stems from the limitations of classical workflows and the growing need for benchmarks, scalable point cloud training data for machine learning and AI applications (Abreu et al., 2023).

Traditional Scan-to-BIM workflows consist of several processing steps, starting with point cloud measurement, followed by geometry segmentation and segment classification, and culminating in the reconstruction of 3D models or IFC data. The problem is that most approaches are project-specific and work primarily on the data for which they were developed. As data acquisition and pre-processing, including labeling, are time-consuming (Abreu et al., 2023), most approaches are limited in their generalizability.

To address these problems, simulation-based approaches generate synthetic point clouds from existing 3D models. Virtual laser scanning of BIM or CAD models can produce labeled point clouds automatically (Noichl et al., 2021). These approaches create synthetic point clouds by sampling points from the surfaces of explicit geometric representations, with semantic labels inherited from the underlying CAD or BIM structure (Pierdicca et al., 2019). In particular, IFC-based environments enable the simulation of laser-scanning processes to generate semantically annotated point clouds for Scan-to-BIM applications (Ma et al., 2021). Experimental results demonstrated in that case that synthetic point clouds enhanced the semantic segmentation tasks and reduced the amount of manually annotated real-world data required for training and semantic segmentation (Xiao et al., 2022; Stefanelli et al., 2022). But in these approaches, the variability of the generated data is constrained by the availability and diversity of predefined geometric models used for the simulated generation of synthetic point clouds.

Machine learning approaches usually target generative latent-space models. Achlioptas et al. (2018) introduced a deep generative model for point clouds by training autoencoders (AE) and modeling the latent space using Generative Adversarial Networks (GANs) and Gaussian Mixture Models (GMMs). This approach demonstrated the ability to reconstruct previously unseen point cloud samples from the ModelNet dataset.

More recent work uses 2D diffusion models to generate architectural images Paananen et al. (2024), floor plans He et al. (2023), or building designs Zhong et al. (2024). These applications remain 2D, limiting their applicability to 3D point cloud generation. The use of diffusion models for 3D point cloud generation is presented in Yue et al. (2025), where the authors propose a method to generate synthetic point clouds to augment real-world training data. This method still relies on large amounts of real-world data to train the diffusion model and, therefore, does not address the core limitations.

In addition, generative text models such as Large Language Models (LLMs) are increasingly being used in the AECO domain. Zheng and Fischer (2023) uses an LLM for natural language processing to develop an assistant for information retrieval from BIM models, and Wei and Li (2025); Du et al. (2026) use LLMs to generate floor plans. In context of point clouds, Hong et al. (2023) and Ye and Gao (2024) use LLMs for 3D interpretation of point cloud data to solve tasks such as captioning, question answering, navigation, and guidance via prompting. Xu et al. (2024) presents PointLLM, which combines point clouds and text for classification and description. Similarly, Yang et al. (2025) proposes LiDAR-LLM, which uses this combination to process LiDAR data. Further applications include point cloud registration with MiniGPT-3D Tang et al. (2024) and segmentation with SegPoint He et al. (2024). While these works demonstrate the potential of LLMs for point cloud tasks, the generation of point clouds with LLMs remains underexplored (Liu et al., 2024).

The challenge with LLMs like GPT-4 Achiam et al. (2023) is that they can analyze and generate text, but they cannot directly generate point clouds, as these data are typically unstructured data, stored in binary or other non-textual formats. LLMs have demonstrated strong capabilities in generating code from natural language Poldrack et al. (2023), and various libraries for processing and creating point cloud data exist, such as PyVista, Open3D, PyntCloud, or Laspy. Yet, using these libraries requires domain knowledge and additional manual programming effort. This opens an alternative path for point cloud generation, in which LLMs generate executable code that produces point cloud data directly, without requiring pre-existing 3D models or real-world training data.

While LLMs are good at code generation, they may lack domain knowledge and knowledge about the point cloud libraries mentioned above (Gu et al., 2025). In addition, LLMs are prone to hallucinations, have limited spatial understanding, and exhibit limited reasoning about causal constraints (Ploennigs et al., 2025). Therefore, special precautions are needed to ensure that code-based point cloud generation yields feasible results. The key advantage of the LLM-based approach is its ability to translate natural language into executable code, enabling prompt-driven scene generation without predefined rule sets.

These limitations motivate the exploration and design of a discussion-based LLM framework called SYN3D-LLM, for generating synthetic, semantically point clouds.

Methodology of the Framework

The workflow of the framework is presented in Figure 1. It starts with a *user prompt* that specifies what kind of point cloud should be generated. This user prompt is extended by the Designer with a *system prompt* that provides additional instructions for generating the point cloud. This encompasses strict code generation constraints, including the output schema, programming language (Python), and a restriction that external libraries are not allowed. It also provides information on the helper functions that the LLM can use to generate the point clouds. For example, helper functions exist for creating wall or ceiling structures filled with surface points. After the user prompt and the Designer system prompt are combined, the Designer generates a first code version. This code is not directly executed, but is sent to another LLM, called the Critic.

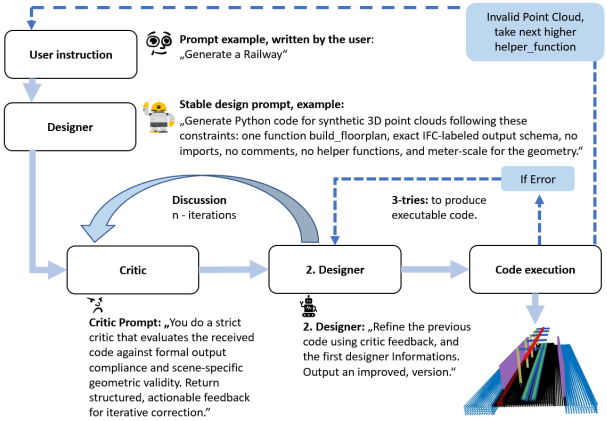

Figure 1: The proposed framework of SYN3D-LLM with its self-discussion functionality to obtain executable code through an iterative process for generating synthetic point cloud data.

The Critic receives the Designer's drafted code, another system prompt defining how it should review the code, and the information from the user prompt. The Critic's response is then sent, together with the previously created code, to the next iteration of the exchange between the Designer and Critic. The Designer then creates an improved code version based on the Critic's feedback and the user prompt. The framework performs a fixed number of Designer and Critic iterations, set to two by default, and the first design draft. With each draft and iteration, the framework has multiple tries to generate executable code (default: 3). The iteration setup represents a controlled experimental design rather than an optimisation stopping strategy. Increasing the number of iterations can improve refinements, but it also increases computational cost. The code's execution is checked by running it automatically in a Python environment. If execution succeeds and a constraint checker verifies that the point cloud contains points, the framework proceeds with the export. Over the iterations, the generated point cloud code becomes progressively more detailed. These improvements are not driven by changes in the *user prompt*, but emerge from the iterative refinement process between the Designer and Critic.

While writing the code, the LLM is instructed in the system prompt to label the individual objects and components based on the scene description.

The discussions between the Designer and Critic are automatically logged to *.CSV* and *.JSON* files for evaluation.

After all iterations are complete, the final code is executed to generate the point cloud. It is saved in three different formats: *.e57*, *.xyz*, and *.ply*.

The framework is complemented by a Graphical User Interface (GUI) in Open3D. It provides access to the settings (e.g., number of iterations), system prompts, and all scene descriptions and variations to generate the data. In the GUI, the discussions between the Designer and Critic are logged in real-time. The GUI also allows the selection of different building types and infrastructure scenarios. For each scenario, a different prompt templates exist that contains the system prompt for the Designer, along with helper functions and the system prompt for the Critic. Currently, templates exist for high-rise buildings, industrial halls, streets (with traffic signs), railway tracks, bridges, and apartments in different shapes. The code is available on GitHub [1].

Validation

The evaluation of SYN3D-LLM focuses on the framework itself and on the plausibility of the point clouds generated from the initial geometric dimensions specified by the combination of the **user prompt** and the **system prompt**. Standard baseline metrics were not applied, since commonly used measures such as Intersection over Union (IoU) or Chamfer Distance (CD) require ground-truth data, which are not available for the proposed SYN3D-LLM framework. The evaluation is exploratory, focusing on plausibility rather than strict geometric accuracy. Instead,

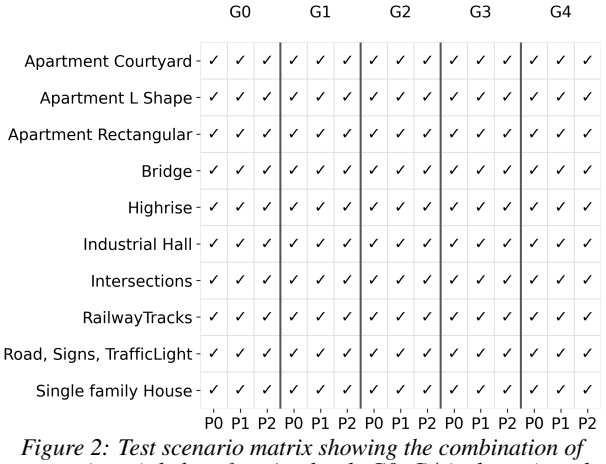

Figure 2: Test scenario matrix showing the combination of scenes (rows), helper function levels G0–G4 (columns), and prompt variations P0–P2 used for evaluation.

we constructed a structured matrix of test scenarios to systematically evaluate the generation process across different

[1] https://github.com/NILSK94/SYN3D-LLM-SYNTHETIC-3D-POINT-CLOUDS-GENERATED-THROUGH-LLM-DISCUSSION

building and infrastructure types in the AECO domain, as illustrated in Figure 2.

Scenes: On the left side of Figure 2 are the scenes we generated, which are described in the code as part of the user input. For example: *A high-rise building with a central core and repeating floors, with an approximate footprint of 18 m × 18 m. It has at least six storeys, with floor levels at z = 0, 3, 6, 9, 12, and 15 m, and a top (roof) level at z = 18 m. The core consists of walls with additional inner walls and columns, windows in the outer walls, and a few corridor doors. Required IFC elements: IfcWall, IfcSlab, IfcColumn, IfcWindow.*

Helper functions G0–G4: At the top of Figure 2, we show the helper functions that are activated during generation. These helper functions act as modular constraints and processing helpers, guiding the LLM by providing code snippets and predefined functions for point cloud generation. The helper functions consist of the objects listed in Table 1.

Table 1: Helper functions for shapes and objects

Helper function	Includes
G0_NoHelper	None
G1_Minimal	Plane Surface
G2_BoxSurface	Rectangle Box
G3_BoxPlusRound	Rectangle Box, Round
G4_Specialized	Rectangle Box, Round, Specialized Objects

Prompt variations P0–P2: In addition, the bottom row of the matrix in Figure 2 shows the prompt variations *P0*, *P1*, and *P2*, which are used to steer the generation and evaluate how prompts influence LLM behavior. While the geometry profiles control which helper functions are available, the prompt variants modify the textual instructions given to the model, ranging from an unconstrained baseline to more specific and robustness-oriented testing. This test is derived from findings on the influence of role allocation in prompt engineering, which can improve LLM performance. The prompt variants are summarized in Table 2.

Table 2: Prompt variants for Apartments as an example

Prompt Variant	Instruction Add-on
P0_Base	None (baseline prompt)
P1_More_Specific	Be explicit: use clear axis-aligned geometry, avoid random noise, and keep openings placed on walls.
P2_Robustness	Robustness: include mild occlusion gaps and slight density variation across elements.

Evaluation metrics

In addition to evaluating the framework, we conducted quantitative analyses, including lightweight checks directly derived from the framework's generated point clouds and log files, to underline the qualitative analysis.

Pass Rate: measures the number of Designer-Critic iterations required until executable Python code is generated. An iteration is considered a pass if the generated code runs without errors and produces valid output. The metric is used across all scenes and configurations.

CountS (Count Successful): verifies each iteration produces a non-empty point cloud, both overall and for required IFC classes where applicable. Reporting the iterations that yield actual point sets rather than empty results.

B–IoU (Bounding-box IoU): measures the overlap between the bounding box of the generated point cloud and the target dimensions specified in the scenario prompt. Values range from 0 to 1, serving as a geometric plausibility indicator for how closely the generated cloud matches the intended dimensions.

Evaluation and Results

For the evaluation, we ran every variation shown in Figure 2. This comprises all scenes and configurations: fifteen runs per scene, each consisting of an initial design step followed by two Designer–Critic iterations. In total, we evaluated more than 300 runs of the framework using an OpenAI GPT-5.2 model, with fixed temperature settings of 0.6 for the designer and 0.4 for the critic. The resulting point clouds were subsequently analyzed. The evaluation starts with a qualitative analysis of the functionality, then moves to the quantitative metrics. To explain the results, we first focus on variations in the helper functions (*G0–G4*) across configurations of the rectangular apartment scene to illustrate how each variation refines the scene design.

Helper function G0–G4 evaluation: Figure 3 shows the result for *G0* (no helper function) and the refinement of the scene across iterations. Over the iterations, the model was able to place doors and windows in plausible positions, so that every room is accessible and has at least one window; The placement of supports remained unreliable. Overall, this highlights the importance of the iterative loop, which progressively refines the design.

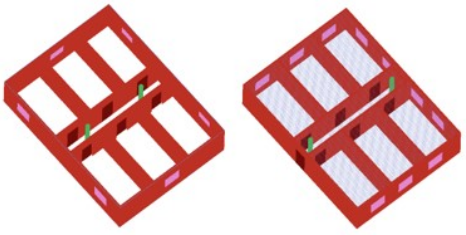

Figure 3: Apartment rectangular layout, prompt variation (P0), helper (G0), iteration 1 (left), iteration 2 (right)

In addition, we analyzed the helper functions provided within the SYN3D-LLM framework by examining their influence when invoked in configurations *G1* through *G4*. For the rectangular apartment layout, this influence was found to be minimal. Nevertheless, the results indicate that the LLM was capable of generating code autonomously without relying on helper functions, and that outputs generated without them tend to exhibit creative variance.

Result pass rate: When considering the pass rate of executable code generation, the results differ across the evalu-

ated helper function variants. Our analysis across all configurations shows that the helper function is crucial for producing executable code in the first iteration. The results are shown in Figure 4. The best pass rate was achieved with the helper function *G2*, which was designed to generate rectangular boxes filled with points on their surfaces. In this case, the function appears to guide the LLM toward producing more structured code output that can be executed directly. In contrast to runs without a helper function (*G0*), the LLM required more iterations to produce executable code.

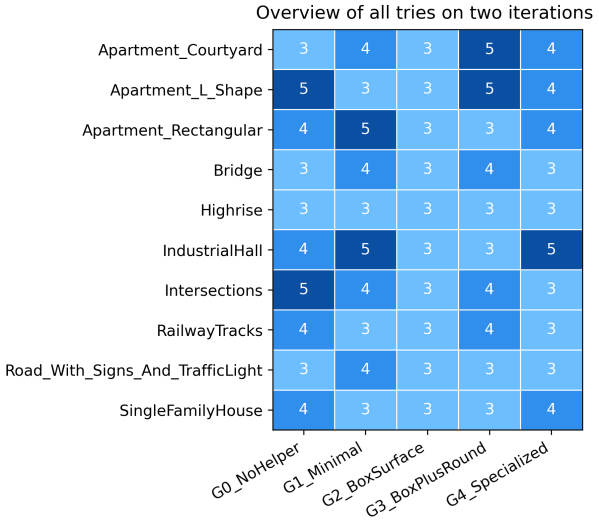

Figure 4: Heatmap of pass rates (two iterations and first design draft required) across scenes and configurations for executable code generation.

CountS: Across all executed iterations in Figure 4, CountS was counted in the Framework, indicating that no empty point cloud outputs occurred in generations because the maximum of 6, by a set default to 3, was not reached.

Prompt variation P0–P2 evaluation: To present the results for prompt variations *P0–P2* (shown for the apartment example in Table 2), we selected the bridge generation scenario to illustrate the influence of prompt variation. The first variant, *P0*, generated the scene shown in Figure 5.

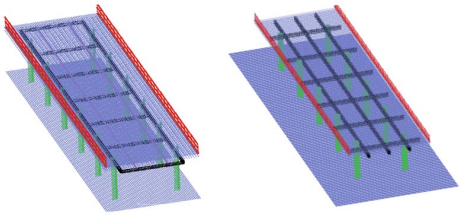

Figure 5: Bridge scene, prompt variation (P0), helper (G0), iteration 1 (left), iteration 2 (right)

Figure 5 shows the baseline prompt *P0* without any additional prompt instructions. While the LLM generates a bridge-like scene, the result is not among the strongest outcomes of the approach, as the geometry remains partially inconsistent and structurally incomplete. In particular, the placement of the supports is not well integrated into a coherent load-bearing bridge structure.

The run with the more specific prompt variant *P1* for the bridge addressed some of the issues observed in *P0*, as shown in Figure 6. The supports are better aligned with the bridge's load-bearing structure, and the LLM generates more detail, particularly in the cut and in the railings or other load-bearing elements. In addition, the model demonstrates the ability to introduce openings in the railings and to incorporate random noise into the generated point clouds. A specific noise setting is also available in the SYN3D-LLM GUI.

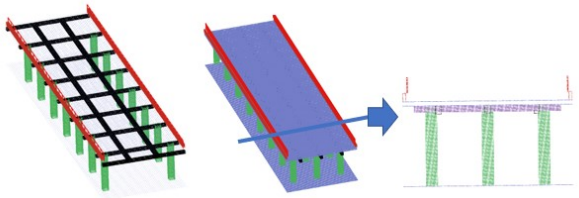

Figure 6: Bridge scene, prompt variation (P1), helper (G0), iteration 1 (left), iteration 2 (middle), cut of iteration 2 (right)

The results obtained with the robustness-oriented prompt variant *P2* are shown in Figure 7. The outcome clearly indicates that the LLM followed the instructions by introducing mild occlusion gaps and slight density variations across elements in the generated point cloud. Beyond improving the baseline prompt's arrangement of supports and load-bearing structure in the generated bridge scenes, *P2* also leads to greater object variation. For example, the LLM attempts to introduce foundations at the ends of the bridge, which are not yet properly connected to the main structure. In addition, the bridge girders beneath the deck are modeled as I-beams, representing substantial structural detail. This can be seen in the cut on the right side of Figure 7. Overall, these results demonstrate that prompt variation can produce meaningful improvements and more plausible outputs within the SYN3D-LLM framework.

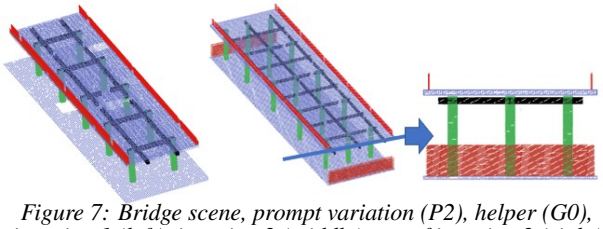

Figure 7: Bridge scene, prompt variation (P2), helper (G0), iteration 1 (left), iteration 2 (middle), cut of iteration 2 (right)

Prompt variant P2 realism effects: Figure 8 presents a qualitative comparison between a synthetic bridge point cloud generated using the prompt variant *P2* (left) and a real-world point cloud acquired with a BLK2GO mobile laser scanner (right). In both cases, mild occlusion gaps and line-like sampling patterns are observable. As illustrated in Figure 7, spatially varying point densities occur across structural elements, particularly along the bridge deck, as well as occlusion-induced shadow regions on the

ground behind supporting columns. In the synthetic example, these effects are generated through the coded point cloud. In real scans, similar characteristics arise from viewpoint dependencies, scanner motion, and the partial occlusions of mobile laser scanning. This visual analysis indicates that SYN3D-LLM generates geometric imperfections like those observed by mobile laser scanners, without explicitly simulating the sensor behavior.

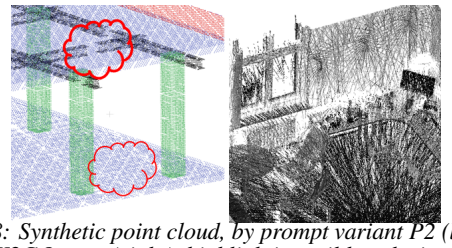

Figure 8: Synthetic point cloud, by prompt variant P2 (left) and a BLK2GO scan (right), highlighting mild occlusion gaps, line-like sampling patterns.

Geometrical check B–IoU: The radar plot in Figure 9 shows how closely the generated synthetic point clouds match the scene dimensions specified in the prompt. Overall, the alignment of the bounding boxes with the target dimensions is minimal.

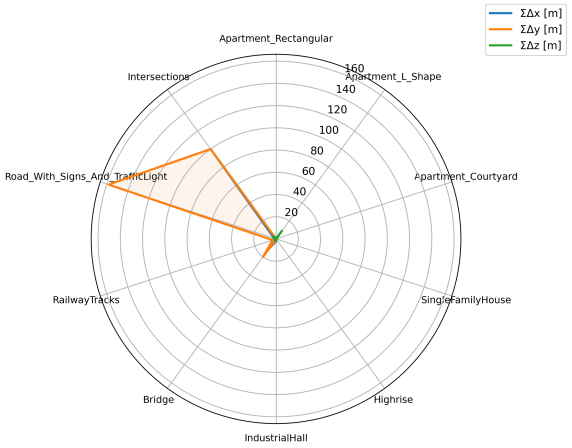

Figure 9: Radar plot showing B-IoU along each axis, comparing generated point clouds to target scene dimensions.

The LLM struggled to generate street scenes with signs and traffic lights, as well as intersection scenes. This can also be observed in the generated data. Figure 10 shows that roads and intersections can be generated, but that there are still major issues in representing these scenes correctly.

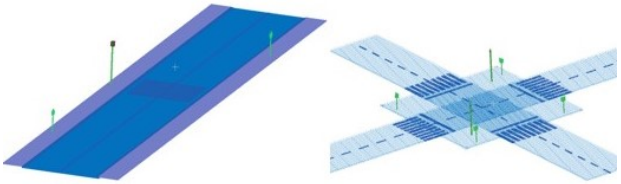

Figure 10: Generated road scene with signs and zebra crossing (left), intersection scene (right).

Semantic labels: The SYN3D-LLM framework stores labels for individual IFC objects as RGB color values in the point cloud. The input scene includes information on the required and permitted IFC classes. The evaluation compares the stored RGB color values in the point cloud with the required and permitted IFC classes specified in the input prompts. The results show a 100 % match for the required and permitted IFC classes explicitly specified in the input prompts. This verifies class presence only and does not assess instance-level correctness or spatial relationships.

Computational cost (Tokens): While the qualitative results demonstrate that iterative refinement substantially improves the structural plausibility and semantic consistency of the generated point clouds, this improvement comes at an increased computational cost.

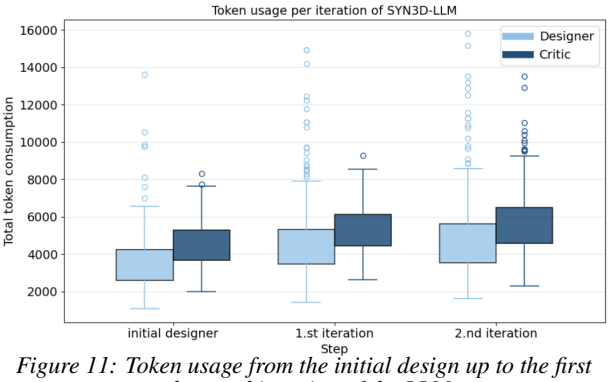

Figure 11: Token usage from the initial design up to the first and second iteration of the LLMs.

Figure 11 shows the token usage per iteration. The results show an increase in the number of tokens required from the initial generation to the first and second iterations. Overall, the Critic needed more tokens to evaluate the design. The round dots in the plot indicate outliers, which are typical for complex tasks.

Discussion

The aim of the paper was to investigate whether an iteration-based, self-discussing framework is capable of generating semantically labeled synthetic point clouds and to examine how reliably the framework follows its input prompts. The results of the proposed SYN3D-LLM framework demonstrate that it successfully generates semantically labeled point clouds. The analysis indicates that the generated point clouds capture the scene's structural elements, but precise geometric alignment is limited in evaluation because the generation is performed without relying on ground-truth data.

The helper functions tested in the framework primarily improve the framework's pass rate and reduce the APIs costs. We can therefore conclude that integrating helper functions is beneficial. But there are limitations, the helper functions may also constrain the LLMs independent creative freedom in generating scenes. Same applies to the predefined input prompts in the framework. Hallucinations are difficult to assess. We should note that the bounding boxes examined using the geometric B-IoU show deviations for certain classes, such as bridges and roads. The

labels indicate that the LLM did not incorporate any new objects into the generated data without being prompted to do so. While semantic hallucinations were not observed, minor geometric deviations in dimensioning cannot be fully ruled out, as they would not be captured by this method. When interpreting the experiments carried out and using the proposed framework, several limitations must be taken into account. The framework was not developed with the aim of generating point cloud data that faithfully reproduces real scan conditions. The absence of an explicit sensor model may introduce a synthetic-to-real gap and limits generalisation to real-world data. The evaluation shows scan patterns that reflect some essential characteristics of real data acquisition, but we need to implement it in future developments. The framework focuses on the variance of point cloud generation. The paper shows that the challenge is not data generation alone, but the controlled, systematic variation of data. Even with model-based approaches, the variance remains in the initial models. In contrast, the variance in SYN3D-LLM is constrained by user prompts, creating a wider variation space.

In particular, for evaluating post-processing steps in point cloud pipelines. Algorithmic methods can be applied to the generated data from SYN3D-LLM and evaluated on constrained data, for example, in the context of ceiling reconstruction.

Outlook

Now we demonstrate our generated data in a short:

Exploratory qualitative case study: that uses the Scan-to-BIM reconstruction approach from Zbirovskỳ and Nežerka (2025) to reconstruct ceilings. We have not yet achieved any results for walls or other components. The SYN3D-LLM framework still needs improvement in this regard. For the Cloud2BIM approach to work, a density of 0.002 meters should be used during generation. The results can be seen in Figure 12. The reconstructed lower ceiling matched the point cloud accurately, but the upper ceiling did not.

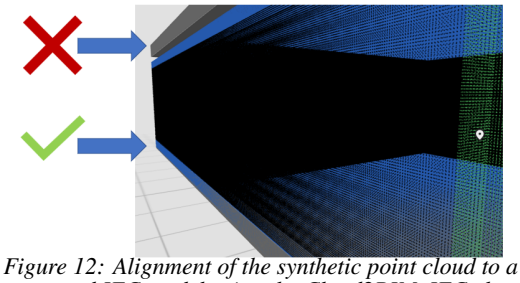

Figure 12: Alignment of the synthetic point cloud to a reconstructed IFC model using the Cloud2BIM. IFC elements shown in grey, point cloud shown in blue, black, and green.

Future work should investigate stopping criteria for controlled refinements in the iteration process and for optimising the computational cost. Furthermore, agent-based approaches are being considered for future development, similar to the Model Context Protocol approach for IFC-based modeling by Nithyanantham et al. (2025). Our next approach could leverage tools that parameterize different point cloud types, enabling more detailed data generation based on user inputs.

Conclusions

This paper presents the SYN3D-LLM framework and demonstrates the potential of synthetic point cloud generation via self-discussing LLMs.

Regarding **RQ1**, the evaluation shows that the framework reliably generates executable code across all tested configurations. The results indicate that helper functions, particularly the G2 configuration, improve first-iteration pass rates. Configurations without helper functions (G0) require additional iterations but still converge to valid outputs within the allowed number of tries.

Regarding **RQ2**, the generated point clouds show plausible structure, but limited geometric precision in complex scenes. Semantic accuracy reached 100% for the required IFC classes specified in the input prompts. But the infrastructure scenes, such as roads and intersections, remain challenging, with larger geometric deviations observed.

Overall, SYN3D-LLM represents a promising approach for generating synthetic point clouds at scale, variations, and for testing reconstruction workflows in the AECO domain. Further optimization is needed, particularly for complex infrastructure scenarios.

References

Abreu, N., Pinto, A., Matos, A., and Pires, M. (2023). Procedural point cloud modelling in scan-to-BIM and scan-vs-BIM applications: A review. ISPRS Int. J. of Geo-Information, 12(7):260.

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.

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. (2018). Learning representations and generative models for 3D point clouds. In ICML, pages 40–49. PMLR.

Du, C., Esser, S., Nousias, S., and Borrmann, A. (2026). Text2BIM: Generating building models using a large language model-based multiagent framework. J. of Computing in Civil Engineering, 40(2):04025142.

Gu, X., Chen, M., Lin, Y., Hu, Y., Zhang, H., Wan, C., Wei, Z., Xu, Y., and Wang, J. (2025). On the effectiveness of large language models in domain-specific code generation. ACM Trans. on Software Engineering and Methodology, 34(3):1–22.

He, S., Ding, H., Jiang, X., and Wen, B. (2024). SegPoint: Segment any point cloud via large language model. In ECCV. Springer.

He, Z., Wang, Y.-H., and Zhang, J. (2023). Generative AIBIM: An automatic and intelligent structural design pipeline integrating BIM and generative AI.

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. NeurIPS, 36.

Liu, D., Huang, X., Hou, Y., Wang, Z., Yin, Z., Gong, Y., Gao, P., and Ouyang, W. (2024). Uni3D-LLM: Unifying point cloud perception, generation and editing with large language models.

Luo, J., Yang, Z., Shi, P., and Ye, Q. (2026). Indoor scan-to-BIM automation: From mobile perception to 3D building modelling. Automation in Construction.

Ma, J. W., Han, B., and Leite, F. (2021). An automated framework for generating synthetic point clouds from as-built bim with semantic annotation for scan-to-bim. In 2021 winter simulation conference (WSC), pages 1–10. IEEE.

Nithyanantham, B. K., Sesterhenn, T., Nedungadi, A., Garijo, S. P., Zenkner, J., Bartelt, C., and Lüdtke, S. (2025). MCP4IFC: IFC-based building design using large language models. arXiv preprint arXiv:2511.05533.

Noichl, F., Braun, A., and Borrmann, A. (2021). BIM-to-scan for scan-to-BIM: Generating realistic synthetic ground truth point clouds based on industrial 3D models. In EC3, volume 2, pages 164–172. European Council on Computing in Construction.

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

Pierdicca, R., Mameli, M., Malinverni, E. S., Paolanti, M., and Frontoni, E. (2019). Automatic generation of point cloud synthetic dataset for historical building representation. In International Conference on Augmented Reality, Virtual Reality and Computer Graphics, pages 203–219. Springer.

Ploennigs, J., Berger, M., Wortmann, T., Kirchner, J., Beetz, J., Roitberg, A., Menzel, K., and Ommer, B. (2025). Building foundation models - potentials, challenges and research directions for using LLM and LVM in AEC. In EC3.

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

Stefanelli, M. G., Jain, A. R., Jalui, S. K., and Agapaki, E. (2022). Synthetic point cloud generation for class segmentation applications. arXiv preprint arXiv:2205.03738.

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 ACM Multimedia.

Wei, Y. and Li, X. (2025). Text-to-code generation for modular building layouts in building information modeling. arXiv preprint arXiv:2509.23713.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D shapenets: A deep representation for volumetric shapes. In CVPR, pages 1912–1920.

Xiao, A., Huang, J., Guan, D., Zhan, F., and Lu, S. (2022). Transfer learning from synthetic to real lidar point cloud for semantic segmentation. In Proceedings of the AAAI conference on artificial intelligence, volume 36, pages 2795–2803.

Xu, R., Wang, X., Wang, T., Chen, Y., Pang, J., and Lin, D. (2024). PointLLM: Empowering large language models to understand point clouds. In ECCV. Springer.

Xu, Y. and Stilla, U. (2021). Toward building and civil infrastructure reconstruction from point clouds: A review on data and key techniques. IEEE J. of Selected Topics in Applied Earth Observations and Remote Sensing, 14:2857–2885.

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, volume 39.

Ye, Y. and Gao, W. (2024). LLM-PCGC: Large language model-based point cloud geometry compression. arXiv preprint arXiv:2408.08682.

Yue, H., Wang, Q., Huang, L., and Zhang, M. (2025). Enhancing point cloud semantic segmentation of building interiors through diffusion-based scene-level synthesis. Automation in Construction.

Zbirovskỳ, S. and Nežerka, V. (2025). Cloud2BIM: An open-source automatic pipeline for efficient conversion of large-scale point clouds into IFC format. arXiv preprint arXiv:2503.11498.

Zheng, J. and Fischer, M. (2023). Dynamic prompt-based virtual assistant framework for BIM information search. Automation in Construction, 155:105067.

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

Zou, Y., Liang, T., Chen, W., Ren, Z., and Wen, Y. (2026). A BIM-derived synthetic point cloud (SPC) dataset for construction scene component segmentation. Data, 11(1):16.