Advanced Engineering Informatics 71 (2026) 104326

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Enabling AI-driven modular building design: an auto-decoder approach for IFC 3D geometry representation

Sang Du [a,b], Lei Hou [a,b,*], Guomin (Kevin) Zhang [a,b], Yang Zou [c], Haosen Chen [a,b]

[a] School of Engineering, RMIT University, Melbourne 3000 Victoria, Australia
[b] Centre for Future Construction, RMIT University, Melbourne 3000 Victoria, Australia
[c] Department of Civil and Environmental Engineering, University of Auckland, Auckland 1023, New Zealand

ARTICLE INFO

Keywords:
Modular building design
Industry Foundation Classes (IFC)
3D geometry representation
Auto-decoder
Design for Manufacture and Assembly (DfMA)
Signed distance function

ABSTRACT

Modular building design requires numerous context-dependent component variants that traditional constraint-based methods cannot exhaustively enumerate. Industry Foundation Classes (IFC) models encode rich spatial and semantic context from completed modular projects. This context could enable Artificial Intelligence (AI) models to generate component variants and complement constraint-based methods. However, IFC 3D geometry that carries spatial context is not directly usable by AI models. This stems from IFC's complex data structure. To address this limitation, this paper proposes a readily deployable auto-decoder method that produces AI-compatible vectors from IFC geometry. First, an IFC export strategy that retains component spatial context is employed. Second, a sampling method that pairs 3D points with their distances to the nearest surface is applied. Third, an auto-decoder neural network that jointly optimises per-component vectors and the model weights is presented, yielding context-aware representation vectors for modular components. Finally, an octree-based decoder for accurate geometry recovery from vectors is employed. Experiments on real-world modular project data demonstrate that the resulting vectors preserve geometric fidelity and support component variant generation. Geometric fidelity is confirmed by the mean and maximum surface reconstruction errors of 14.57 mm and 51.94 mm, sufficient for modular building design analysis. Support for component variant generation is evidenced by geometric interpolation linearity exceeding 0.98 out of 1, showing excellent variant generation suitability. This method makes IFC spatial context accessible to AI-driven modular design methods, transforming Design for Manufacture and Assembly (DfMA) data into actionable knowledge. Codes available on GitHub.

1. Introduction

Modular building exemplifies industrialised construction by using prefabricated volumetric units that enable rapid on-site assembly, consistent quality, and enhanced productivity [1–4]. During modular building design, standardised modules and components need to be configured to accommodate project-specific context such as adjacent modules, connection interfaces, and mechanical, electrical, and plumbing (MEP) routing [5]. Given these standardised characteristics and repeated design patterns, the design process for modular buildings is inherently suited to automated approaches [5]. Current automated approaches predominantly focus on constraint-based methods, such as parametric configurators, rule-based optimisation algorithms, and genetic algorithms with predefined design rules [6–8]. Constraint-based methods rely on manually encoding explicit design rules. However, context differences across modular units and components create numerous possible scenarios [9]. Manual rule encoding cannot exhaustively enumerate all of them [9,10]. Data-driven methods can complement constraint-based methods by automatically learning patterns from existing data [11], thus handling unenumerated scenarios through generalisation.

Artificial Intelligence (AI) has emerged as a prominent data-driven paradigm in recent years [12,13]. For instance, in the prefabricated construction sector, AI models have demonstrated the capability to detect bottlenecks in modular prefabrication and to evaluate the surface quality of prefabricated concrete panels [14,15]. These applications demonstrate the potential of AI in the industrialised construction context. However, developing such AI-driven approaches requires

* Corresponding author at: School of Engineering, RMIT University, Melbourne 3000, Victoria, Australia.
E-mail addresses: sang.du@student.rmit.edu.au (S. Du), lei.hou@rmit.edu.au (L. Hou), kevin.zhang@rmit.edu.au (G.(K. Zhang), yang.zou@auckland.ac.nz (Y. Zou), haosen.chen@rmit.edu.au (H. Chen).

https://doi.org/10.1016/j.aei.2026.104326
Received 22 October 2025; Received in revised form 12 December 2025; Accepted 8 January 2026
Available online 13 January 2026

access to high-quality design data [16]. Such data exists in Industry Foundation Classes (IFC) models, which encode rich spatial and semantic context from completed modular projects [17–19]. IFC models from completed modular building projects serve as de facto, installation-validated design exemplars, encapsulating valuable Design for Manufacture and Assembly (DfMA) knowledge accumulated through real-world assembly processes. This context, represented as structured data in IFC models, has supported various AI-based studies, including intelligent construction management, automated risk assessment, automated compliance checking, and autonomous construction robotics [20–25]. Despite this promise, research on AI-driven modular building design has rarely leveraged IFC directly, thereby overlooking the enriched spatial context it offers. For example, Liu et al. [26] utilised a graph-constrained generative adversarial network to generate modular high-rise residential building floor plans based on a layout image dataset. Similarly, Ghannad and Lee [27] developed a coupled generative adversarial network to automate the design of modular housing using real-world building layouts. Despite IFC's native encoding of 3D geometric and spatial information, both approaches relied on 2D layout representations derived from alternative sources. Overlooking IFC's 3D geometric data is especially limiting for modular building design, where design decisions regarding component variants depend on the accurate spatial context of neighbouring components and modular units [28,29]. Therefore, effective reasoning about these design considerations requires AI to comprehend 3D spatial context.

The problem stems from the fact that AI models have difficulty directly processing IFC's complex geometry data [16], creating a critical gap between IFC data and AI-compatible formats. IFC employs multiple paradigms for geometry representation, with each paradigm relying on distinct mathematical formulations and modelling procedures to describe geometry [17]. Table A.1 (Appendix A) summarises the main paradigms for solid geometry representation in IFC. By contrast, AI-driven methods require data formats to be unified and fixed-dimensional [30]. To address this, existing research in the Architecture, Engineering, and Construction domain has explored indirect approaches to utilising BIM geometry, most notably programmatic methods [31–33]. Programmatic methods describe construction procedures rather than explicit surfaces, which often strip away spatial and semantic context between objects. These limitations motivate the search for alternative representations that retain complete spatial context and support AI utilisation.

Among the possible alternatives, 3D geometry embedding emerges as a promising solution. This method converts 3D geometry into unified, fixed-length vectors for AI use while preserving fine-grained geometric detail. Broadly, such methods can be grouped into explicit and implicit embeddings [34]. Explicit embeddings are effective for discriminative tasks such as classification [35–46], whereas implicit embeddings achieve more substantial geometry fidelity [47–50]. In this study, we focus on signed distance function (SDF)-based implicit embeddings, which describe a shape using points in space and their distance to the shape [51]. This method yields AI-compatible vectors. However, existing SDF-based methods typically normalise objects to facilitate shape learning [48]. This step removes absolute scale, position, and orientation. DfMA-centric modular building design requires precise spatial information for assembly feasibility, inter-unit coordination, and module-to-module interface design [28,52,53]. This includes inter-unit joints, MEP service clearances, and lifting frames. It must also support the generation of component variants. These variants need to remain compatible with the same spatial and fabrication constraints. As a result, such normalisation limits the suitability of current SDF methods for modular building design workflows that require both accurate spatial context and continuous exploration of DfMA-compliant component variants.

To bridge the representation gap between enriched IFC data and the requirements of AI-driven design, this study presents an auto-decoder that produces AI-compatible vectors from IFC geometry while retaining essential spatial context, including real-world scale, position and orientation. The learned representation supports volumetric spatial reasoning and continuous exploration of component variants. In this way, accumulated DfMA knowledge from past projects becomes actionable for future designs. The contributions of this study include:

1. Proposing a unified SDF-based auto-decoder that converts IFC geometry into compact AI-ready latent vectors while preserving accurate spatial context. This unified representation supports downstream AI tasks, including smooth interpolation of shape, size, spatial arrangement, and spatially aware reasoning. It also enables variant generation, which is essential for modular building design.
2. Developing a ready-to-use web-based tool with an interactive interface that automates the complete IFC-to-representation pipeline, facilitating the application of this method to data-driven modular building design.

The remainder of this paper is organised as follows: Section 2 examines current methods for converting IFC geometry data, Section 3 details the methodologies, Section 4 presents the experimental outcomes, Section 5 discusses and suggests future research directions, and Section 6 draws conclusions.

2. Literature review

3D geometry embedding is a specialised method for utilising 3D geometry as training data in the computer vision domain. The method employs a dedicated process to map the geometries into fixed-length vectors. 3D geometry embedding methods can be broadly categorised into explicit methods and implicit methods, based on the type of input data and the approach used for mapping [34]. A summary of the main categories, their inputs, and their typical strengths and limitations is provided in Table 1. The following section offers a brief explanation of each pathway.

2.1. Explicit embedding methods

Explicit methods encode the visible geometry of an object into fixed-length vectors using multi-view images, point clouds, voxel grids, or polygonal meshes. Multi-view projection converts BIM elements into sets of rendered 2D views, allowing direct reuse of standard image-based models; for example, Zhou et al. [44] generated product images from BIM and trained a recommender for component styles. This pathway benefits from mature 2D computer vision tooling and uniform data formats, but inevitably discards volumetric and topological information, and the choice of viewpoints can bias what the model "sees." Point cloud encoders, such as PointNet [36] and DGCNN [37], operate on sampled surface points, aggregating both global shape and local neighbourhood features; they preserve metric shape cues but are sensitive to sparsity, uneven sampling, and noise, and they lack explicit connectivity. Voxel-grid methods, exemplified by VoxNet [39] and 3D-VAE [54], discretise space into regular 3D grids that align well with 3D CNNs and are robust to small gaps. However, their resolution is limited by memory and computation, so fine geometric detail is lost. Polygonal mesh encoders such as MeshCNN [42] and MeshGPT [35] operate directly on vertices, edges, and faces, preserving high-fidelity surfaces, normals, and topology, thereby enabling accurate shape analysis and simulation. However, their performance depends strongly on mesh quality and consistent triangulation. In general, explicit embeddings perform well on discriminative tasks, such as classification and retrieval of BIM components. In contrast, the resolution and cleanliness of the underlying explicit geometry limit the fidelity of their reconstruction.

2.2. Implicit embedding methods

Implicit methods represent a 3D object as a continuous field in space and train a neural network to approximate that field, yielding a compact

Table 1
Comparison of explicit and implicit 3D geometry embedding methods.

Methods	Studies	Key advantages	Key limitations	Typical tasks
Multi-view projection (Explicit)	Zhou et al. [44]; Koo et al. [45]; Xu et al. [46].	Reuses mature 2D CV pipelines; simple, uniform data format compatible with standard image models.	Discards volumetric and topological information; view selection may bias representations.	BIM element classification, defect identification, component style suggestion.
Point cloud (Explicit)	PointNet [36]; DGCNN [37]; Point-E [38].	Preserves metric shape; capture both global structure and local neighbourhood features.	Sensitive to sparsity, uneven sampling and noise; lack explicit topology.	Shape retrieval, component recognition, sparse geometric analysis.
Voxel grid (Explicit)	VoxNet [39]; VoxGRAF [40]; Vox-E [41];3D-VAE [54].	Regular 3D structure compatible with 3D Convolutional Neural Networks (CNNs); robust to small gaps.	Resolution limited by compute/memory; fine geometric detail lost.	Coarse 3D understanding, occupancy reasoning, volumetric synthesis.
Polygonal mesh (Explicit)	MeshCNN [42]; MeshNet [43]; MeshGPT [35]	Retains high-fidelity surfaces, normals, and explicit topology; suitable for accurate shape analysis	Highly dependent on mesh quality and consistent triangulation.	Simulation-oriented tasks, precise BIM component analysis, mesh-based generation.
SDF-based (Implicit)	DeepSDF [48]; NeuS [47]; BuildNet3D [55]	High-fidelity reconstruction; smooth interpolation; compact latent codes	Requires per-shape normalisation that removes absolute position, scale, and orientation, thereby losing spatial context.	Shape reconstruction, generative modelling, interpolation.
Neural radiance field (Implicit)	NeRF [56]; Block-NeRF [57]	Photorealistic novel view synthesis: models view-dependent appearance; strong multi-view consistency	Primarily image-space supervision; geometry is implicit in density/radiance; optimisation and rendering can be computationally expensive; less aligned with BIM's explicit object semantics.	Novel view synthesis, scene appearance modelling, view-consistent 3D scene representation.
Occupancy/hybrid implicit field (Implicit)	Occupancy Networks [49]; IF-Net [50].	Flexible topology; can integrate CNN feature volumes for higher detail	Same normalisation issue; weaker separability for discriminative tasks; can be computationally heavy.	Topology-flexible reconstruction, high-detail implicit modelling.

latent code that supports high-fidelity reconstruction and smooth interpolation. SDF-based approaches such as DeepSDF [48], NeuS [47], and BuildNet3D [55] learn signed distance fields that describe how far any point is from the surface, enabling precise surfaces and expressive latent spaces that can capture subtle shape variation. Neural radiance field methods, including NeRF [56] and Block-NeRF [57], instead learn a volumetric field of density and radiance, achieving photorealistic novel view synthesis with strong multi-view consistency. However, they are primarily supervised in image space, and their geometry remains implicit in the learned density. Occupancy and hybrid implicit field methods, such as Occupancy Networks [49] and IF-Net [50], classify points as inside or outside an object and may condition the implicit decoder on voxel-aligned feature volumes to recover greater local detail with flexible topology. Across these families, a common practice is to normalise each shape by translating it to the origin, scaling it to unit size, and aligning it to fixed axes before learning. This canonicalisation improves shape-focused generalisation but removes absolute position, real-world scale, and project-coordinate orientation. As a result, while implicit embeddings excel at reconstruction and interpolation, they tend to perform less well on discriminative tasks. They are poorly suited to BIM applications where modular variants depend on spatial and semantic context, such as adjacency, interface constraints, service routing, and assembly tolerances.

2.3. Summary and research gap

Existing explicit and implicit geometry embedding methods offer useful ways to convert 3D shapes into AI-ready vectors and have shown strong performance on discriminative and generative tasks in computer vision benchmarks. In engineering applications, however, AI tools do not yet provide sufficient accuracy to replace design judgment. Their role is better understood as supplying high-fidelity geometric information that assists designers in visual reasoning. Many explicit embeddings compress visible geometry into formats that are convenient for learning but lose volumetric, topological, or scale-related cues that are important for modular construction. Many implicit embeddings focus on learning canonicalised shapes and normalise each object to a unit coordinate frame, thereby removing the project-coordinate spatial context needed to distinguish modular IFC components and their interfaces. At the same time, established design practice is organised around visual inspection of drawings, models, and coordinated layouts, so AI-supported workflows should allow geometry encoded in latent vectors to be reconstructed and visually inspected rather than remaining only as abstract numerical features. These considerations motivate the need for a representation that both preserves the spatial setting of IFC components and supports accurate, interpretable geometry reconstruction. The present study, therefore, investigates an SDF-based auto-decoder as a practical approach to obtaining IFC-aligned latent vectors that can be converted back into viewable, spatially coherent geometry for AI-driven modular design tasks.

3. Methodology

This study proposes an auto-decoder method for transforming IFC 3D geometric data into AI-compatible representation vectors suitable for AI-driven modular building design. Implemented as an interactive web-based tool, the method makes the process straightforward and efficient. Inspired by the architecture of DeepSDF [48], this study develops a pipeline in four parts:

1. Component extraction, isolating the target component from IFC while preserving spatial context.
2. Sampling and SDF generation, converting the extracted component into spatial point samples with signed distance values.
3. Auto-decoder implementation, training an auto-decoder neural network that comprises a shared decoder and component-specific representation vectors.
4. Decoding process, recovering geometry from the learned model and representation vectors.

The overall process is summarised in Fig. 1, and specific adaptations are detailed in the following sections.

To ensure practical deployment, the proposed methodology has been packaged as a readily deployable interactive web-based tool, as shown in Fig. 2. The tool streamlines the complete workflow through guided prompts and automated configurations, allowing users to process IFC data, train models, and reconstruct geometries with ease. The web-based deployment aligns with current industry trends and enables the use of high-performance cloud computing resources, facilitating scalable computation while reducing the need for complex local setups.

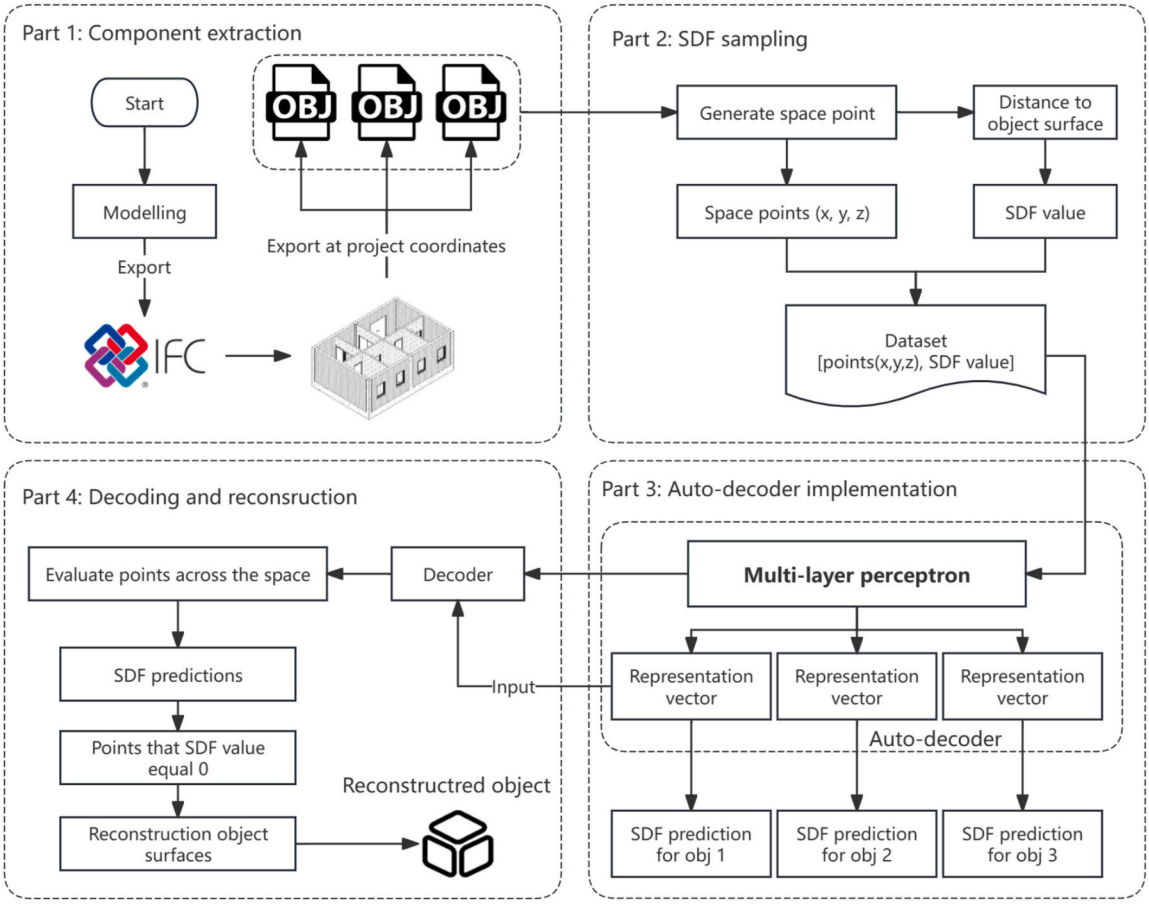

Fig. 1. Architecture of the proposed method.

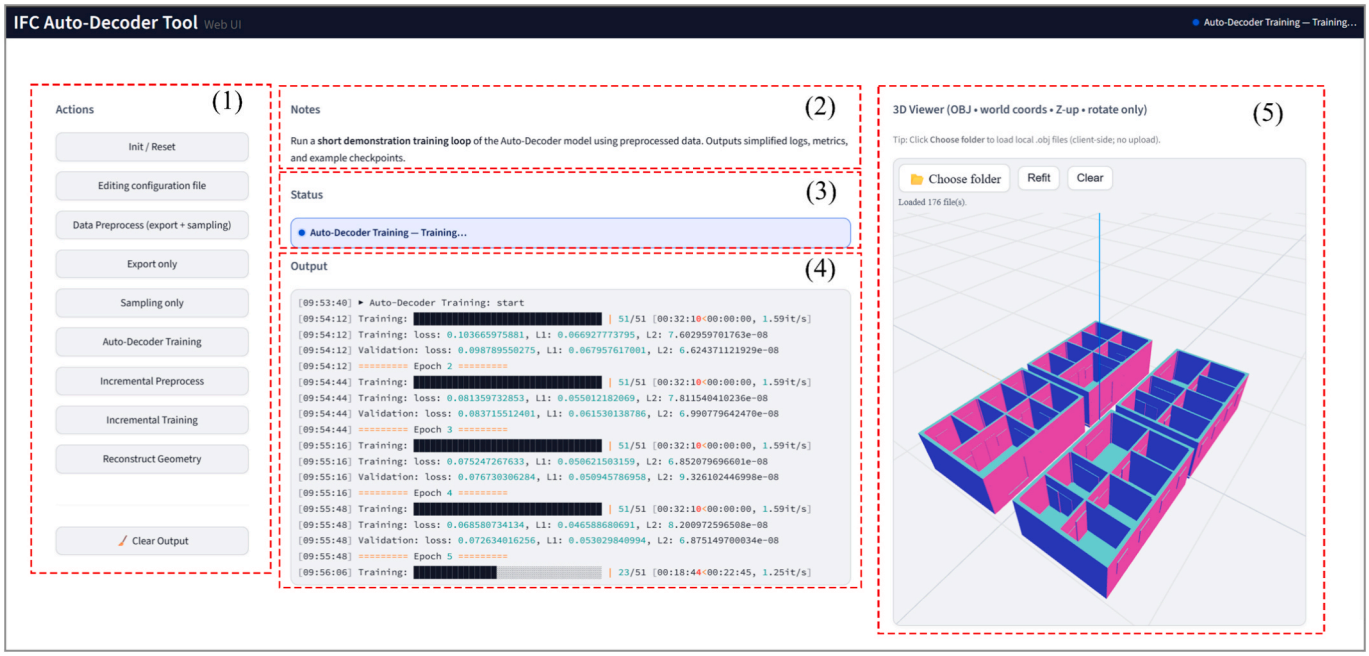

Fig. 2. Interactive web-based tool: (1) command menu listing all available actions, (2) step-by-step instructions for the selected task, (3) live status indicator, (4) program output console, and (5) standalone OBJ viewer.

3.1. Component extraction

The proposed methodology begins with extracting and preprocessing IFC components to ensure compatibility with SDF-based neural representation learning. This preprocessing addresses the fact that SDF computation requires watertight, closed geometric surfaces, while typical IFC models contain complex multi-part assemblies and non-manifold geometries. In addition, IFC files originating from real-world construction projects are typically produced with different authoring tools and heterogeneous modelling practices, leading to substantial variability in geometric representations. Even for nominally similar components, differences in modelling strategies can result in inconsistent topological structures, non-manifold or non-watertight solids, and fragmented or redundant geometric definitions. These characteristics directly affect the usability of IFC-derived geometry for data-driven workflows, as many downstream operations implicitly assume clean, closed, and consistently structured solids. In practice, such irregularities can cause mesh extraction, voxelisation, and other geometry-dependent processing steps to fail or produce unstable results. This issue becomes particularly evident when working with large-scale, heterogeneous IFC datasets from completed projects. The component extraction process focuses on six primary IFC entity types: IfcColumn, IfcDoor, IfcBeam, IfcWall, IfcWindow, and IfcSlab. These categories represent the most common structural and architectural elements in modular buildings and provide sufficient geometric diversity for validating the proposed approach. The proposed method generalises to other component categories, including MEP systems. However, MEP instances in our available datasets exhibit project-specific modelling conventions and inconsistent labelling, which would compromise internal validity. This study therefore first establishes results on consistently annotated architectural elements.

Each IFC component is first converted into a watertight mesh suitable for SDF sampling by exporting the IFC model to OBJ format and repairing the resulting geometry where necessary. All meshes are then uniformly downscaled by a factor of 0.1, which acts as a simple linear change of units, so that both coordinates and signed distance values lie in an order-one numerical range. This global rescaling is introduced solely to improve the conditioning of the optimisation problem, keep the SDF sampling domain compact, and avoid excessively large or small gradients during training. In this work, the global scale is therefore treated as a fixed normalisation choice that is coupled with the sampling scheme and optimisation hyperparameters, rather than as a physically meaningful parameter to be tuned separately for components of different sizes. The value 0.1 was selected empirically as a practical compromise that balances numerical precision with training stability across the range of IFC components considered. While other scale factors could be adopted for different datasets or training setups, a systematic sensitivity analysis over multiple scales would require a separate numerical study and is beyond the scope of this work. Because the auto-decoder is trained from scratch on the rescaled data, the network parameters can adapt to the chosen global scale, and reconstruction quality is expected to be largely insensitive to reasonable variations in this factor.

To preserve the spatial semantics of the original IFC model, the original coordinates and orientations of each component in the project coordinate frame are stored alongside the scaled meshes. This ensures that absolute placement and relative spatial relationships between components can be recovered after processing. Specifically, the workflow retains the real-world scale, project-coordinate position, and orientation information derived from the IFC schema for each component. This approach distinguishes the preprocessing from conventional normalisation pipelines that discard global context by mapping each object to an abstract unit cube. The scaling step described above affects only the numerical magnitudes used during subsequent processing and does not alter the stored spatial reference frame or the relationships between components.

Many IFC components also consist of multiple disjoint geometric parts. For example, window assemblies often contain separate solids for frames, panes, and hardware, and door objects may include distinct elements for panels, frames, and handles. To handle these heterogeneous representations and ensure watertightness for SDF sampling, the workflow employs geometric proxy substitution: windows and doors are replaced by simplified cuboid proxies that preserve their spatial bounds and functional volumes, and wall constructions are restricted to single-layer representations to eliminate internal cavities. These preprocessing steps, combined with global scaling, yield consistent, watertight meshes that are numerically well-conditioned for SDF-based learning while preserving the essential spatial structure encoded in the original IFC model.

Because the IFC schema is intentionally permissive, many components in real projects are modelled as assemblies of separate sub-elements that often follow non-standard, project-specific conventions. In the proposed framework, the simplification step merges such multi-part assemblies, for example, the frame, pane and hardware of a window or the panel and frame of a door, into a single watertight proxy that preserves the overall envelope and functional volume of the object. This reduces the number of solids without altering the essential geometry at the level relevant to modular design, so it does not negatively affect SDF sampling or subsequent model training. Designing robust proxy substitution rules for the whole variety of such non-standard assemblies, however, requires additional engineering effort and coordination with project modelling standards and lies outside the scope of the present study. The implementation utilises Python-based tools, including IfcOpenShell for IFC parsing and Trimesh for geometric processing and validation. Table 2 presents the algorithmic workflow for the IFC-to-OBJ conversion process.

3.2. Sampling and SDF generation

The auto-decoder requires training data in 3D samples annotated with signed distance values. Following the non-normalised approach established in preprocessing, real-world scale and spatial orientation are preserved to maintain modular building design context integrity.

Point sampling employs a two-stage, surface-guided strategy. The first stage involves uniform surface sampling with density and minimum distance thresholds to ensure adequate surface coverage. The second stage applies local jittering around surface points to provide comprehensive near-field and far-field spatial coverage. This approach is crucial because insufficient far-field sampling can cause the auto-decoder to mispredict spurious surfaces in distant regions. The jittering distances

Table 2
Algorithm A: pseudocode for component extraction.

1.	**Algorithm A: IFC-to-OBJ with proxy substitution (IfcOpenShell-based)**
2.	**Input:** IFC model M
3.	**Output**: Set of OBJ files {OBJ(g)} named by GUID g and Manifest table T
4.	Elements ← SelectElements (M, classes = {Walls, Windows, Doors, ...})
5.	Initialize T ← ∅
6.	For each element e in Elements:
7.	g ← GetGUID(e)
8.	C ← GetIFCClass(e)
9.	G ← TessellateSurface(e)
10.	If C is Window or Door:
11.	B ← ComputeBoundingBox(G)
12.	G' ← MakeCuboidProxy(B)
13.	*proxy* ← True
14.	Else:
15.	G' ← SimplifyAndClean(G)
16.	*proxy* ← False
17.	G' ← EnsureWatertight(G')
18.	If IsValid(G'):
19.	p ← ExportOBJ (G', filename = g + ".obj")
20.	Add (g, C, p, *proxy*, BoundingBox(G')) to T
21.	Return {OBJ(g)}, T

are calibrated according to specifications in Appendix A, Table A.3. The maximum sampling distance is 10 m in real-world coordinates, corresponding to 1.0 in the rescaled SDF space after applying the global scale factor of 0.1. In the SDF field, each sampled point stores its signed distance to the nearest surface. By drawing sample points up to this maximum distance and supervising the network with their signed distances, the model is explicitly informed that no geometry exists within this region. In particular, points within the effective 10 m radius around the object are consistently labelled as free space, thereby discouraging the network from hallucinating spurious surfaces in this zone. This sampling radius, therefore, keeps the 10 m neighbourhood around each component free of false surfaces while providing sufficient coverage for modular building design cases. Each sampled point is paired with its corresponding signed distance value to form the training dataset. This explicit pairing yields dense supervision over both occupied and unoccupied regions, ensuring that near-surface geometry and far-field free space are learned consistently. Consequently, the auto-decoder captures not only the detailed shape of each component but also the surrounding free-space envelope up to a 10 m radius.

The implementation utilises Trimesh for geometric processing, NumPy for numerical operations, and point-cloud-utils for point cloud manipulation. Table 3 presents the algorithmic workflow, illustrating how uniform surface sampling and local jittering are combined to construct training data.

3.3. Auto-decoder implementation

The auto-decoder implementation differs from traditional encoder-decoder architectures by directly optimising representation vectors alongside network parameters during training. This eliminates the need for an explicit encoding network. Each IFC component is assigned a learnable representation vector that combines with spatial coordinates to enable the decoder network to reconstruct the component's signed distance field. Both the decoder weights and the representation vectors are optimised through gradient-based training.

The core architecture consists of a multilayer perceptron that maps a combination of shape-specific representation vectors and spatial coordinates to signed distance values. Specifically, the network takes as input a vector $z \in R^k$ representing object-specific geometric characteristics and a 3D spatial coordinate $x \in R^3$ representing a query point in space. The network function f_θ is mathematically defined as:

$$f_\theta(z, x) \approx \mathrm{SDF}(x) \tag{1}$$

where θ represents the learnable network parameters. The training objective optimises both the network parameters θ and the vector z through a reconstruction loss that minimises the discrepancy between the predicted and ground-truth signed distances. The optimisation problem is formulated as:

$$\min_{\theta, \{z_i\}} = \sum_{i-1}^{N} \sum_{j-1}^{M} \| f_\theta(z_i, x_{ij}) - d_{ij} \|^2 \tag{2}$$

where N represents the total number of objects in the training dataset, M denotes the number of sampled points per object, x_{ij} is the j-th sampled point from the i-th object, and d_{ij} corresponds to its ground truth signed distance value. Fig. 3 illustrates the architecture of the auto-decoder.

The training strategy employs a shared network architecture across modular component types. This prevents instance-specific memorisation and forces the model to encode component-specific geometric variants within the representation vector space. The approach promotes generalisation to unseen objects within the same component family. The auto-decoder is implemented using PyTorch, a widely adopted deep learning framework that provides robust automatic differentiation and GPU acceleration capabilities.

Table 3
Algorithm B: pseudocode for sampling and SDF generation.

1.	**Algorithm B: Sampling and SDF generation for BIM-adapted DeepSDF**
2.	**Input:** Watertight meshes {Oi} from surface extraction; Sampling parameters: *surface_density*, *min_surface_samples*, *jitter_radii*, *scale_factor* = 0.1
3.	**Output**: Training dataset D = {(x, y, z, sdf)}
4.	Initialise D ← ∅
5.	For each mesh O in {Oi}:
6.	O_scaled ← UniformScale (O, factor = scale_factor)
7.	S_surface ← SampleSurfacePoints (O_scaled, density = surface_density, min_count = min_surface_samples)
8.	Initialise S_jittered ← ∅
9.	For each point p in S_surface:
10.	For each radius r in jitter_radii:
11.	Q ← JitterPoint (p, radius = r)
12.	Add Q to S_jittered
13.	S_all ← S_surface ∪ S_jittered
14.	For each point q in S_all:
15.	d ← ComputeSignedDistance (q, O_scaled)
16.	Add (q_x, q_y, q_z, d) to D
17.	Return D

3.4. Reconstruction with decoder

The final stage involves reconstructing explicit 3D surfaces from learned representation vectors to enable bidirectional conversion between IFC geometry and neural representations. This process validates the neural approximation quality and provides a practical interface for downstream AI applications.

The reconstruction approach leverages the mathematical property of SDFs where the zero-level set defines the object's surface boundary. The trained neural network is queried at spatial coordinates throughout a defined domain to identify locations where the predicted signed distance value approaches zero. These zero-crossing points collectively constitute the reconstructed object surface. However, the reconstruction process presents significant computational challenges. The primary challenge stems from the decoupled nature of reconstruction and training data. During training, sample points are generated around known component locations. During reconstruction, the decoder operates without prior knowledge of the component's spatial location and must exhaustively search through potentially large spatial domains. The second challenge involves balancing reconstruction accuracy with computational efficiency. High-resolution surface extraction requires dense spatial sampling, which scales cubically with resolution, resulting in prohibitive computational costs.

To address these constraints, the methodology employs an adaptive octree refinement strategy. The algorithm begins with coarse spatial subdivision and refines the octree by depth, where depth represents the level of recursive subdivision. Cells with SDF values near the surface are subdivided, as illustrated in Fig. 4 from depths 1 to 5.

Areas with SDF magnitudes significantly above a predefined threshold are pruned from further subdivision. This hierarchical

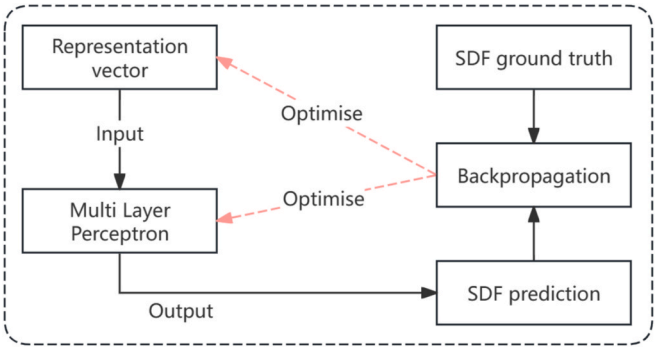

Fig. 3. Auto-decoder schematic.

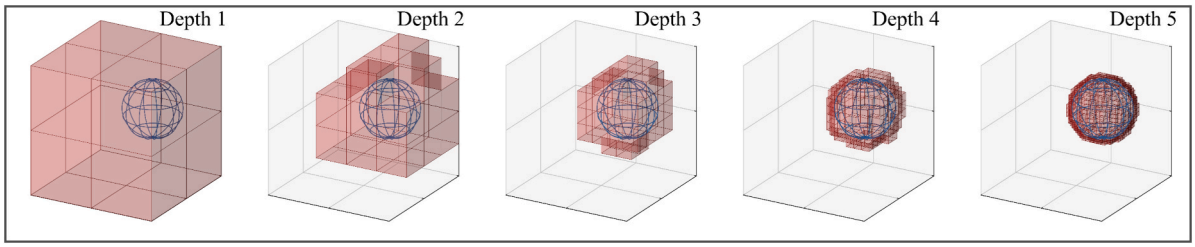

Fig. 4. Conceptual illustration of the octree-based reconstruction strategy. Recursive subdivision from depth 1 to 5 enables efficient capture of object boundaries. Red cells indicate the subdivided regions identified for further refinement near the surface.

approach substantially reduces memory consumption and computational time while accurately capturing surface details.

The reconstruction pipeline combines specialised computational tools. Surface extraction utilises the Marching Cubes algorithm to produce triangulated meshes from the SDF zero-level set. SDF value computation is performed using the trained PyTorch auto-decoder with GPU acceleration for efficient batch processing. Post-processing operations are handled through the Trimesh library. This method enables seamless integration of neural SDF representations with conventional geometric modelling workflows.

4. Experiments and results

This study designed five experiments based on real modular scenarios to evaluate key aspects of the proposed method: computational feasibility under modular unit augmentation, geometric reconstruction accuracy for families of similar yet contextually varied components, characteristics sufficient to support component variant generation, comparative experiment with explicit method, and a downstream AI demonstration. All experiments were conducted using the deployable interactive web-based tool developed as part of this study. The tool provides an interactive interface that guides users through dataset preparation, model training, and reconstruction, with automatic parameter configuration and progress monitoring as shown in Fig. 2.

4.1. Experiment 1: computational feasibility under modular unit augmentation

Capturing spatial context in modular building design requires learning from multi-module scenes rather than from individual modules in isolation. Such multi-module scenes preserve the spatial relationships and contextual dependencies between modules, but they also introduce significant computational challenges. Therefore, this study designed this experiment to verify the computational feasibility of the proposed method under modular unit augmentation, evaluating whether the system can efficiently process increasing numbers of modules while maintaining acceptable performance and quality.

A concrete modular unit was obtained from a real modular project as the base unit, and three progressively augmented datasets were created with increasing quantities of modules. Dataset d1 contains a single unit, d2 contains four units, and d3 contains sixteen units. To study the spatial relationships between modules, the fundamental building components, including walls, columns, doors, and windows were retained. Using identical units for augmentation was intended to minimise interference from varying component quantities across different modules. The augmented datasets were constructed through both horizontal and vertical stacking configurations, simulating common scenarios encountered in modular building design practice. Dataset specifications are summarised in Table 4. Fig. 5 illustrates perspective and plan views of the three datasets in Revit, showing their geometry consistency across different scales.

Training employs a neural auto-decoder with 128-dimensional representation vectors and an eight-layer multilayer perceptron (MLP) with

Table 4
Dataset composition used in Experiment 1.

Dataset	d1	d2	d3
Modular units	1	4	16
Walls	13	52	208
Columns	6	24	96
Beams	7	28	112
Doors	7	28	112
Windows	9	36	144
Floors	2	8	32
Total objects	44	176	704

512 hidden units per layer, with inputs maintained in project coordinates. Experiments were conducted on Ubuntu 24.04.2 LTS with an Intel i5-13400 CPU, NVIDIA RTX 4060 Ti 16 GB GPU, and 128 GB RAM. Detailed system setup is provided in Appendix A, Tables A.2–A.5. Wall-clock time and throughput were recorded for each processing stage. Fig. 6 visualises the Experiment 1 reconstructions from representation vectors for dataset variants d1–d3. Per-stage runtimes and throughput are reported in Table 5, with the corresponding absolute and normalised comparisons shown in Fig. 7. The training dynamics are illustrated in Fig. 8, which shows the convergence behaviour for four metrics: training loss, validation loss, reconstruction loss, and latent loss. The left column displays the whole training progression (0–1000 steps), while the right column focuses on the early stage (0–200 steps). Solid lines indicate raw loss values, and dashed lines show Exponential Moving Average (EMA) smoothed curves (decay $\alpha = 0.15$). The EMA smoothing acts as a filter that removes random fluctuations, producing cleaner curves that make it easier to observe whether the losses are steadily decreasing, plateauing, or oscillating.

The results reveal several key computational characteristics. Model training dominates the end-to-end runtime, accounting for over 60% of total processing time, while preprocessing stages contribute negligibly. Counterintuitively, larger datasets demonstrate improved efficiency: average training time per 10k points decreases from 28.5 seconds (d1) to 18.4 seconds (d3), as shown in Fig. 7. This stems from reduced CPU-GPU coordination overhead with larger batch processing.

The training dynamics in Fig. 8 reveal significant differences in convergence behaviour. The small dataset (d1) exhibits unstable training with oscillating losses, requiring approximately 600 steps to stabilise. In contrast, larger datasets (d2, d3) achieve rapid and stable convergence within approximately 500 steps, suggesting training time could be reduced by 40-50% without sacrificing performance. This improved stability indicates that dataset diversity enhances training robustness more effectively than denser geometric sampling of individual objects. This finding has important implications for data preparation strategies.

These results demonstrate the method's scalability for modular building projects, where standardised components are replicated throughout. The efficiency gains with larger datasets align well with the industrialised nature of modular construction, making the approach increasingly practical as project scale increases.

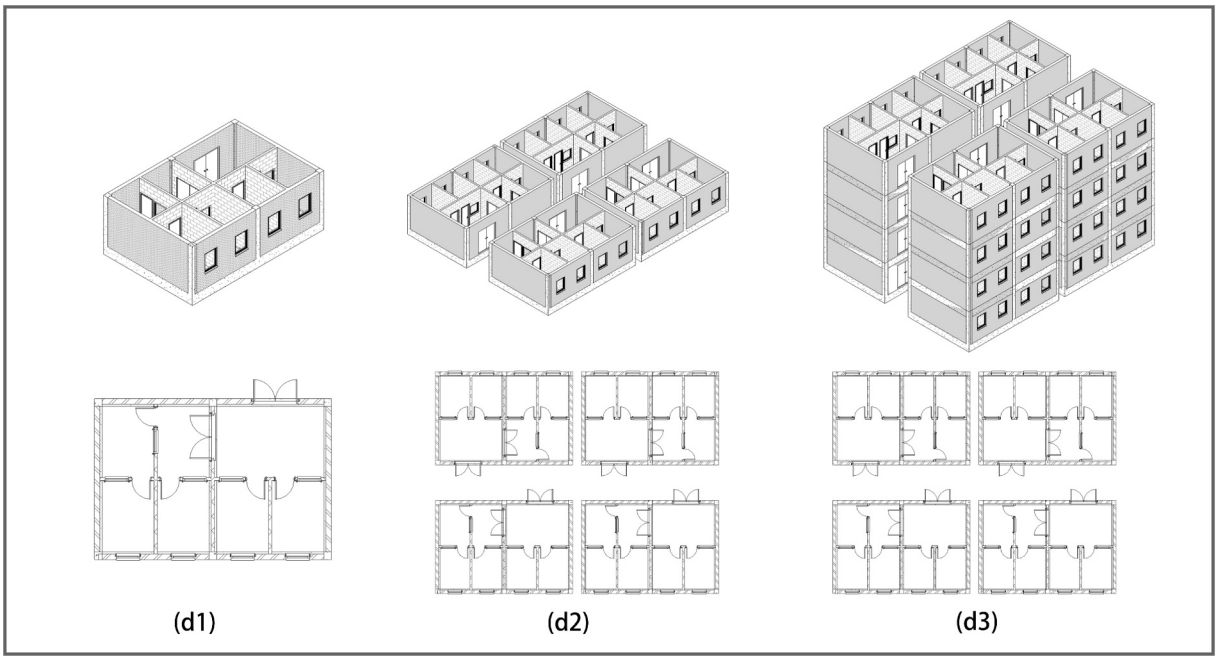

Fig. 5. Example views of the modular building dataset for Experiment 1 in Autodesk Revit, showing perspective 3D representations (top row) and corresponding floor plans (bottom row) for (d1) a single modular unit, (d2) horizontal modular unit replication, and (d3) vertical modular unit stacking.

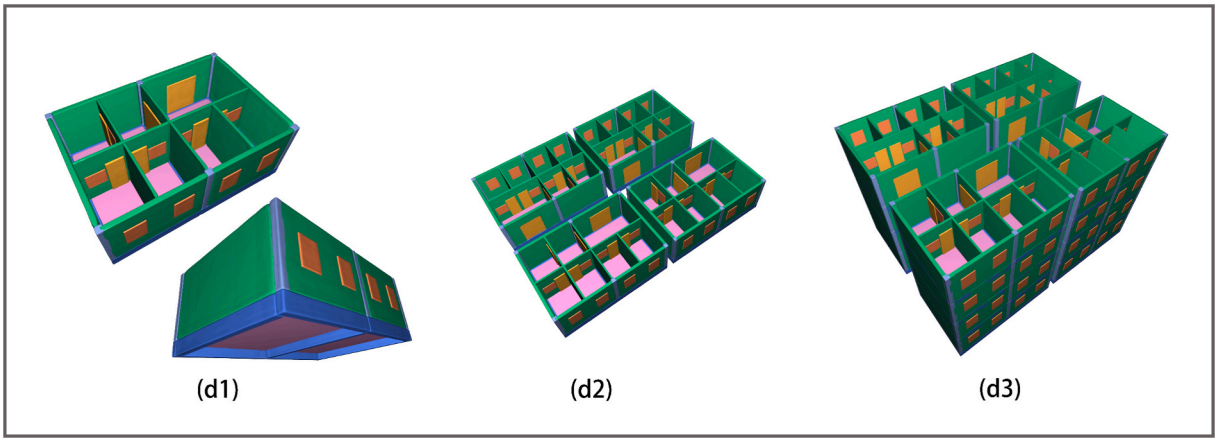

Fig. 6. Experiment 1 reconstructions for dataset variants d1–d3 shown in perspective views. Walls, floors, windows, doors, columns, slabs, and beams are coloured green, teal, orange, yellow, light blue, pink, and blue, respectively, for visual distinction only and not to indicate semantics.

Table 5
Processing times across stages for datasets d1–d3 (seconds; hours in parentheses).

Dataset	d1	d2	d3
Total components	44	176	704
Total sample points	2,911,584	11,364,496	46,536,512
Export time (s)	1.47	5.77	24.00
SDF sampling time (s)	0.89	3.47	13.36
Total training time (s)	8284.47 (2.3 h)	22384.38 (6.22 h)	85740.54 (23.82 h)
Total reconstruction time (s)	3294.06 (0.91 h)	13591.01 (3.76 h)	58518.46 (16.26 h)

4.2. Experiment 2: geometric reconstruction accuracy for component families

Modular buildings consist of numerous similar units, generating a large number of components that share similarities but differ in specific details based on their functional requirements and relationships with adjacent modules. The geometric variants of these components represent responses to surrounding spatial conditions and semantic relationships, making them critical for AI pattern recognition. These subtle geometric differences encode important design logic that must be preserved for meaningful analysis and generation. Therefore, the study examines the geometric reconstruction accuracy for component families to verify whether the representation vectors produced by the proposed method retain sufficient geometric information to capture both the commonalities and variations within component families.

The study selected prefabricated wall components with windows, one of the most common elements in modular building design, as the experimental dataset. The dataset comprises 72 wall components of identical overall dimensions arranged across two levels, with 36 segments per floor. Each component is rotated in 10° increments and contains three windows to introduce geometric complexity. Window sizes are drawn from five predefined types and assigned randomly using a Revit Dynamo script. This configuration simulates the typical window

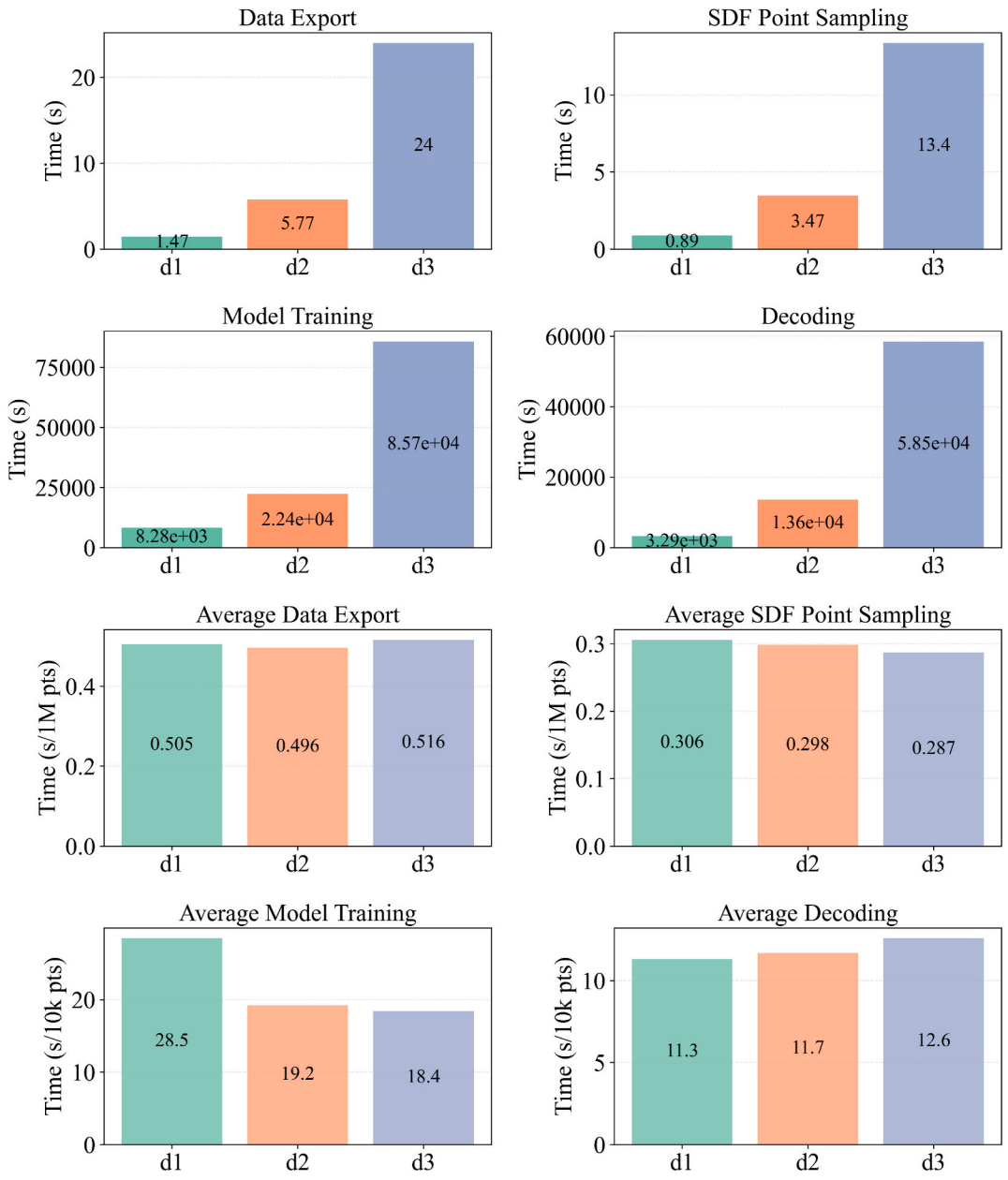

Fig. 7. Stage-wise runtime profiling across dataset variants d1–d3. The top two rows show absolute times for each stage, and the bottom two rows show averages normalised by workload (export/sampling in seconds per million points, training/decoding in seconds per 10k points).

size variation scenarios commonly encountered in modular building design practice, where standardised wall panels accommodate different fenestration requirements. Through this process, the study generated 72 wall components with identical overall dimensions but distinct window opening configurations. To validate the importance of spatial coordinates in preserving geometric information, the study prepared an ablation dataset with identical content but processed with conventional normalisation to a canonical unit space, which removes absolute positional information. Dataset parameters are summarised in Table 6, with the layout shown in Fig. 9.

The study evaluates reconstruction quality using three metrics: Chamfer distance measures average geometric fidelity, Hausdorff distance captures maximum deviation for structural tolerance assessment, and surface normals provide supplementary quality indicators. Two scenarios are tested: (c1) reconstruction with preserved project coordinates and (c2) reconstruction from normalised models (ablation). Table 7 reports component-level errors for c1 compared to the ground truth (GT), and Table 8 reports c2 compared to GT, along with a direct comparison between c1 and c2. Fig. 10 provides qualitative visualisations of the reconstructed model c1. Fig. 11 demonstrates the t-SNE comparison of c1 and c2 as per Table 8. Fig. 12 visualises the metrics in Table 7 and reveals consistent patterns across component families. Figs. 13 - 14 provide Hausdorff distance heatmaps for representative wall components and simplified cuboid reconstructions of windows, respectively. System setup details are provided in Appendix A, Tables A.2–A.5.

The coordinate-preserving approach (c1) achieves reconstruction accuracy comparable to construction tolerance levels, with a Chamfer distance of 14.57 mm and a Hausdorff distance of 51.94 mm. Error distribution reveals three distinct patterns. First, geometric discontinuities concentrate the primary errors, as shown in Figs. 13 - 14. The SDF's inherent continuous nature produces smooth, rounded reconstructions at surface intersections: convex edges appear inward-displaced while concave edges, e.g., window openings, exhibit outward displacement.

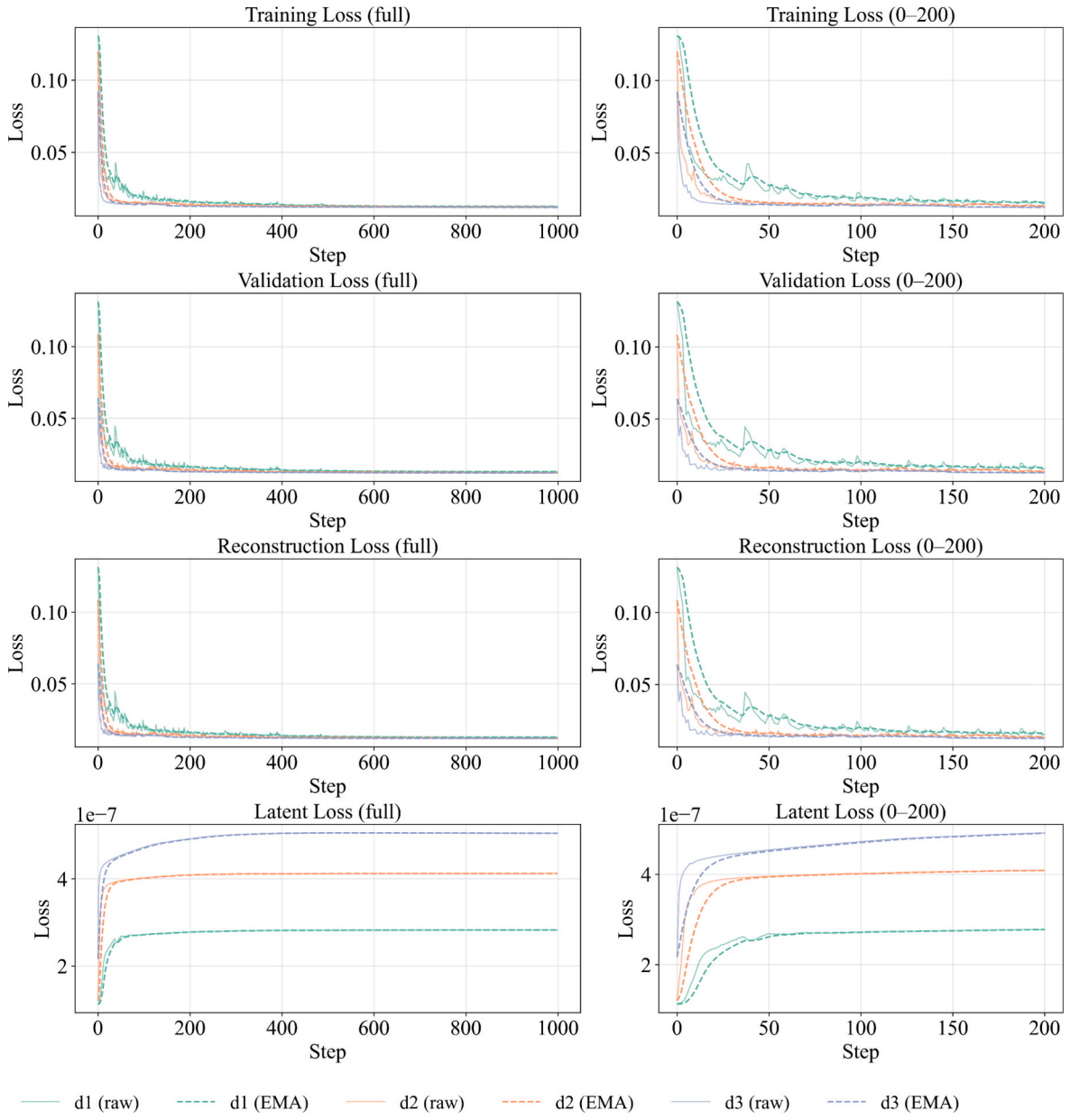

Fig. 8. Convergence of loss functions for datasets d1–d3.

Table 6
Dataset composition used in Experiment 2.

Element	Quantity	Width (mm)	Height (mm)	Thickness (mm)
Wall	72	6000	3000	300
Window type 1	48	600	600	300
Window type 2	42	600	900	300
Window type 3	40	900	900	300
Window type 4	42	900	1200	300
Window type 5	44	1200	1200	300

Second, large planar surfaces maintain high fidelity, demonstrating effectiveness for continuous regions. Third, reconstruction errors correlate positively with object volume, an important consideration for engineering applications.

Current accuracy reflects two implementation factors rather than fundamental limitations. The approximately 19 mm resolution represents a practical balance, as higher resolutions would incur cubic computational costs. Additionally, the octree grid's orthogonal alignment creates reconstruction patterns visible in Fig. 10. Walls parallel to coordinate axes show smooth surfaces, while angled walls display stepped artefacts. These characteristics suggest that application-specific optimisations could further enhance precision.

The ablation study (c2) demonstrates the critical importance of preserving spatial coordinates. Conventional normalisation degrades accuracy by orders of magnitude, with Chamfer distance from 14.57 mm to 5574.75 mm and Hausdorff distance from 51.94 mm to 7201.39 mm. This substantial difference validates that normalised datasets eliminate spatial relationships essential for architectural reasoning, explaining their inadequacy for automated AI design applications. The achieved reconstruction accuracy provides a practical foundation suitable for generative design tasks in architectural workflows. In Fig. 11, the t-SNE plots further show that the latent space has indeed captured spatial relations rather than merely reproducing type labels or metadata. When trained with absolute coordinates, the latent codes for IfcWall and IfcWindow not only form clearly separated clusters but also organise into regular ring-like structures that reflect consistent relative placements of windows along walls. In contrast, training on normalised geometry

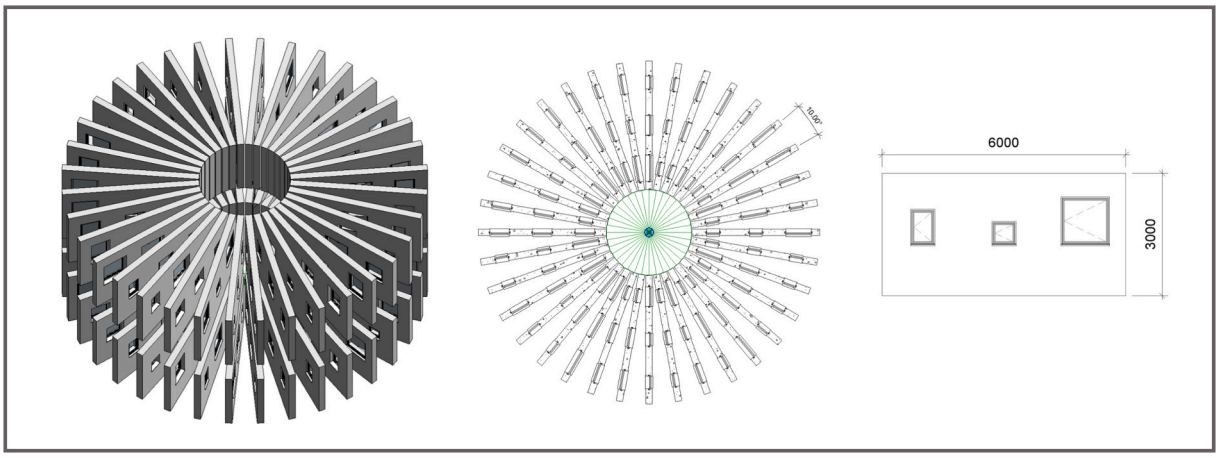

Fig. 9. Example views of the dataset for Experiment 2 in Autodesk Revit, showing perspective 3D representations (left), corresponding floor plans (middle), and an elevation view of a wall component (right).

Table 7
c1 reconstruction errors vs. GT at real-world scale. Metrics are reported per class. Lower is better.

Element	Chamfer distance (mm, mean)	Hausdorff distance (mm, mean)	Surface normal error (deg, mean)
All elements	14.57	51.94	4.40
Wall	31.25	110.80	5.55
Window type 1	6.88	25.98	4.29
Window type 2	7.98	28.72	4.24
Window type 3	9.04	32.09	3.86
Window type 4	10.13	35.72	3.89
Window type 5	11.24	39.66	3.76

Table 8
Aggregate comparison of c1 (Experiment 2, project coordinates) and c2 (ablation, normalised coordinates) reconstruction errors vs. GT. Lower is better.

Scenario	c1	c2
Chamfer Distance (mm, mean)	14.57	5574.75
Hausdorff Distance (mm, mean)	51.94	7201.39
Surface Normal Error (deg, mean)	4.40	67.71

without global coordinates produces a more scattered cloud in which the two classes are mixed, and spatial patterns largely disappeared. Taken together, these quantitative ablation results and the t-SNE visualisation indicate that the IFC-aligned SDF with coordinates learns an AI-usable encoding of spatial layout and adjacency, providing direct access to spatial relationships in the latent space and offering a practical foundation for generative design tasks in architectural workflows.

4.3. Experiment 3: support for component variant generation

Generating component variants is a fundamental task in modular building design. For AI-driven component variant generation, the dataset must explicitly encode the spatial and semantic context in which components exist and establish clear correspondences with target geometries. Consider door openings as an example: the same door family generates multiple variants depending on corridor width, wall thickness, distances to adjacent module interfaces, opening direction, and spatial typology. Suppose data records these contextual factors in direct correspondence with geometric parameters and forms. In that case, the model should be able to map contextual differences to smooth geometric modifications within a unified representation. Consequently, when performing latent space interpolation between representations of doors from the same family, the expected results would be intermediate door forms that transition smoothly along semantic dimensions while maintaining structural and constraint consistency. This continuous transition capability would indicate that the representation is viable and reliable for generating component variants. Therefore, this study designed this interpolation experiment to validate the capability of supporting component variant generation.

To examine interpolation capability under controlled conditions, this study employed a customised component dataset that represents three distinct contextual factors: size, shape, and position. The dataset comprises four distinct column configurations arranged in a systematic 7×7 grid pattern, providing 49 instances of each type within consistent coordinate frameworks. The configurations include square columns with cross-sectional dimensions of 300 mm and 1000 mm, as well as circular columns with diameters of 300 mm and 1000 mm. This study obtained representation vectors for all components using the proposed method, then performed geometric interpolation between them. The interpolation procedure calculates the mathematical difference between two known vectors, divides this difference into six equal segments, and sequentially adds each segment back to the original vector. This process generates five intermediate transition vectors that the model has not encountered during training. These interpolated vectors are then reconstructed to evaluate their transitional states and assess whether the representation supports smooth, semantically meaningful component variants. Dataset parameters are summarised in Table 9, with the layout shown in Fig. 15. This experiment uses the same system setup as in Experiments 1 and 2, with details provided in Appendix A, Tables A.2–A.5.

Seven sequences of progressive transitions are examined. The first sequence (S1) examines dimensional scaling while maintaining consistent positioning and cross-sectional shape. As shown in Fig. 16, the top row illustrates a transition from a large square column (1000 mm × 1000 mm) to a smaller square column (300 mm × 300 mm), while the bottom row displays the corresponding scale change for circular columns (1000 mm to 300 mm diameter). Intermediate steps illustrate smooth dimensional transitions between training samples.

The second sequence (S2) investigates cross-sectional geometry transitions while preserving dimensional and positional parameters. As presented in Fig. 17, the top row shows a transition from a square column (1000 mm × 1000 mm) to a circular column (1000 mm diameter), while the bottom row shows the same transition at a smaller scale (300

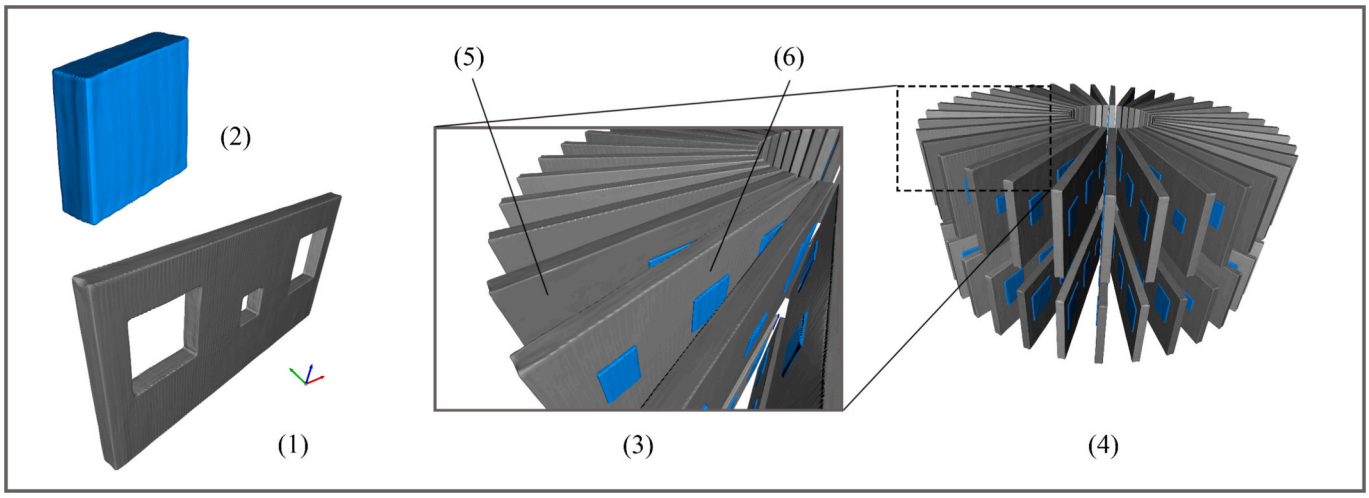

Fig. 10. Qualitative comparison of reconstruction scenarios (c1 vs. c2). (1) a wall component; (2) a window component shown as a simplified cuboid; (3) magnified region from (4) highlighting local surface quality; (4) all reconstructed models rendered together; (5) smooth surface; (6) striped or ribbed surface. Colours are for visual distinction only.

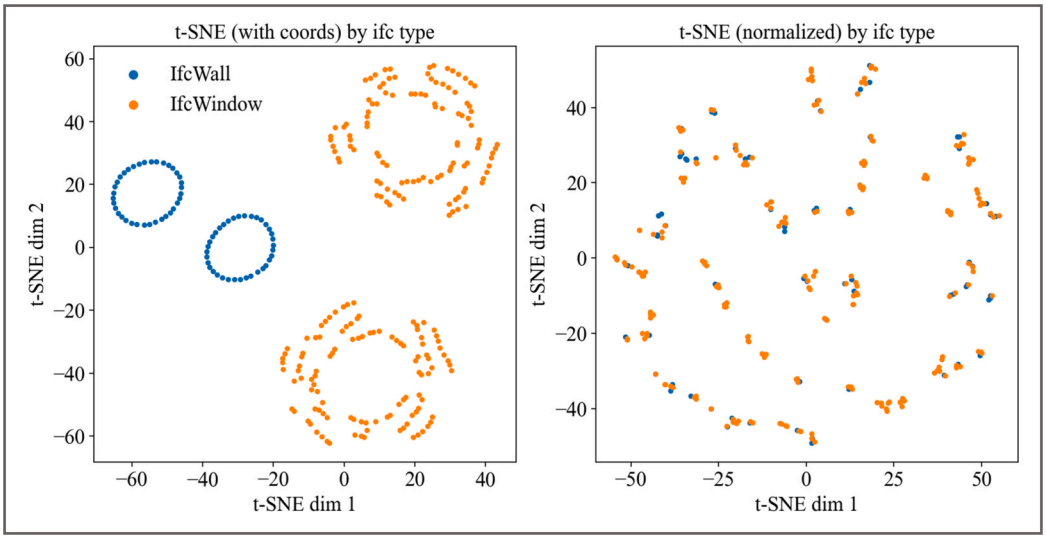

Fig. 11. t-SNE visualisation of latent spaces for two training configurations. Left: latent codes learned with absolute coordinates (with coords) form clearly separated clusters for IfcWall and IfcWindow instances. Right: latent codes learned from normalised geometry (normalised) mix the two classes, indicating reduced class-specific separation in the latent space.

mm × 300 mm to 300 mm diameter). Intermediate steps illustrate smooth geometric morphing, demonstrating the model's ability to generate continuous shape variants between training samples.

The third sequence (S3) combines dimensional and geometric variations while maintaining consistent positioning. As presented in Fig. 18, the top row shows a transition from a square column (1000 mm × 1000 mm) to a circular column (300 mm diameter), while the bottom row illustrates the inverse direction (small square to large circular), which demonstrates smooth geometric morphing from one cross-section type to another.

The fourth sequence (S4) explores positional variations while maintaining consistent dimensions and cross-sectional properties. As presented in Fig. 19, the top row shows a square column (1000 mm × 1000 mm) at (0, 0) smoothly transitioning to (4, 4). The bottom row illustrates intermediate steps, demonstrating the ability to interpolate smoothly in location.

The fifth sequence (S5) examines combined positional and geometric transitions while preserving dimensional parameters. As presented in Fig. 20, the top row shows large square columns (1000 mm × 1000 mm) at (0, 0) that are interpolated into large circular columns (1000 mm diameter) at (4, 4). The bottom row illustrates intermediate steps, demonstrating the ability to interpolate smoothly in both location and shape.

The sixth sequence (S6) examines combined positional and dimensional scaling variations while preserving cross-sectional properties. As presented in Fig. 21, the top row shows a square column (1000 mm × 1000 mm) at (0, 0) smoothly transitioning to a square column (300 mm × 300 mm) at (4, 4). The bottom row illustrates intermediate steps where both geometry and spatial position undergo continuous changes.

The seventh sequence (S7) provides a comprehensive evaluation through simultaneous variation of dimensional, positional, and geometric parameters. As presented in Fig. 22, the top row shows a square column (1000 mm × 1000 mm) at (0, 0) smoothly transitioning to a circular column (300 mm diameter) at (4, 4). The bottom row illustrates intermediate steps where both geometry and spatial position change continuously.

To provide quantitative validation of the visually smooth interpolation results, this study evaluates interpolation quality using three

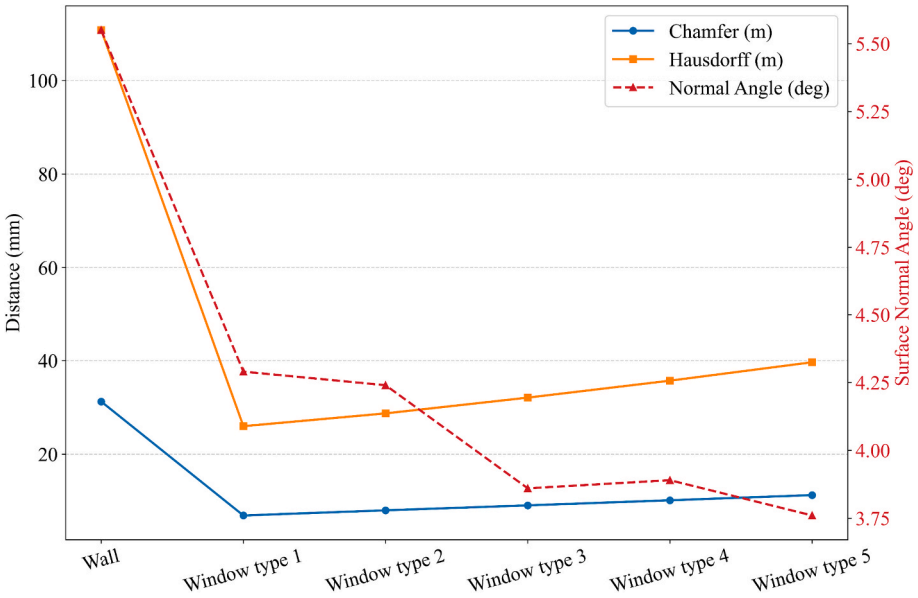

Fig. 12. Graphical summary of c1 reconstruction accuracy across component families.

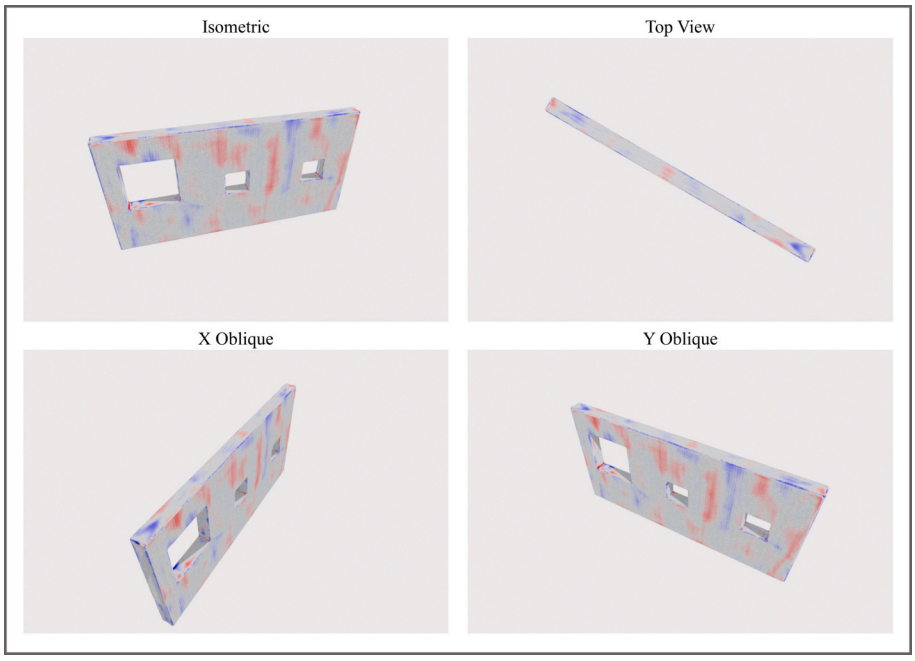

Fig. 13. Hausdorff-error heatmap for a reconstructed wall component, shown from multiple views. Red indicates regions protruding relative to the ground-truth geometry, while blue indicates recessed regions.

deformation-centric metrics. In a learned latent space, there is no observable ground-truth geodesic path between two shapes, because the geometry of the space is defined implicitly by the encoder-decoder, and an analytical or labelled "true" interpolation trajectory is not available. The evaluation, therefore, focuses on how regularly and stably the model distributes deformation along the interpolation path it produces. The stepwise coefficient of variation of pure deformation magnitudes measures the relative dispersion of per-step pure deformation after removing global translation, rotation, and uniform scaling. The cumulative R-squared of pure deformation versus normalised sequence position quantifies how closely cumulative pure deformation increases linearly with normalised sequence position, with values closer to 1 indicating more uniform per-step deformation. The Max/Min ratio of per-step pure deformation magnitudes captures the evenness of step sizes, with values closer to 1 representing more uniform deformation. In addition, Area CV (%) reports the coefficient of variation of surface area over the sequence, indicating the relative magnitude of global size changes, and Area R^2 reports the R-squared of surface area versus normalised sequence position, indicating the linearity of global size changes. Together, these five metrics provide a quantitative assessment of the uniformity and stability of shape change along the interpolation path. Results for all seven sequences are reported in Table 10.

Step CV (%): Stepwise coefficient of variation of pure deformation magnitudes after removing global translation, rotation, and uniform scaling. Cum. R^2: cumulative R-squared of pure deformation versus normalised sequence position; values closer to 1 indicate a more constant speed. Max/Min: Max/Min ratio of per-step pure deformation magnitudes; values closer to 1 indicate more even steps. Area R^2: R-

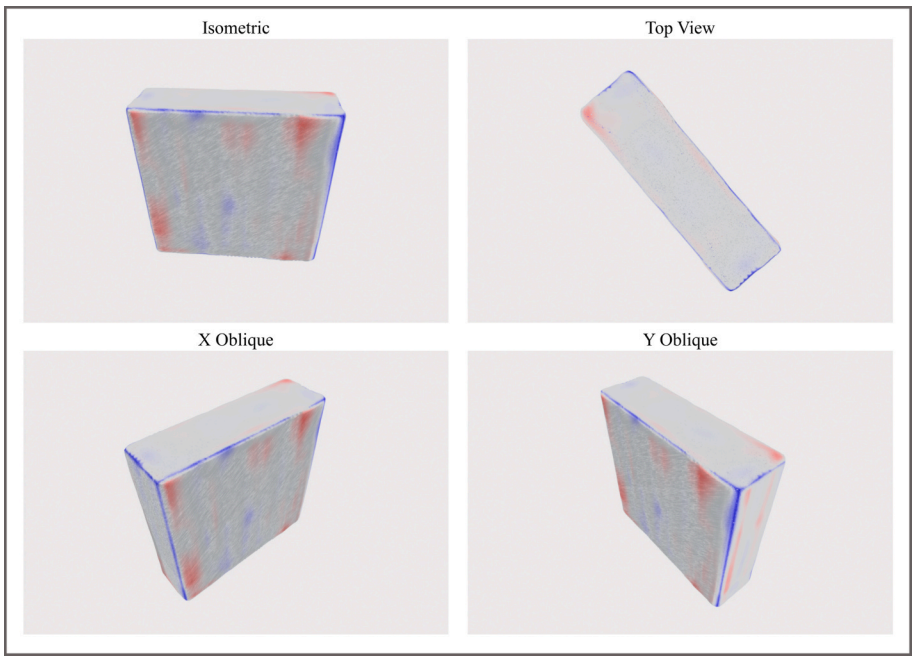

Fig. 14. Hausdorff-error heatmap for a reconstructed window simplified cuboid, shown from multiple views. Red indicates regions protruding relative to the ground-truth geometry, while blue indicates recessed regions.

Table 9
Dataset composition used in Experiment 3.

Elements	Quantity	Parameter	Value (mm)
Square column (large)	49	Side length	1000
Square column (small)	49	Side length	300
Circular column (large)	49	Diameter	1000
Circular column (small)	49	Diameter	300

squared of surface area versus normalised sequence position, indicating the linearity of global size changes. Area CV (%): coefficient of variation of surface area over the sequence, indicating the relative magnitude of global size changes.

Overall, the progression over the sequence is smooth. The cumulative pure deformation follows a normalised sequence position almost linearly, with cumulative linearity between 0.983 and 0.9999. Shape-only interpolation is nearly uniform. Step-size variability ranges from 0.7 to 1.7%. The largest to smallest step ratio ranges from 1.02 to 1.05. When size changes are present, step sizes become less even. The variability ranges from 20 to 28%. The ratio ranges from 1.8 to 2.3. These patterns reflect phase concentration during corner-to-round transitions and coupling between scale and shape. The progression is monotonic rather than jittery. Arc-length reparameterisation can make it uniform. At the surface area level, the interpolation behaves consistently with these deformation-based metrics. When scale changes are modest (S2, S4, S5), the surface area varies only slightly over the sequence (Area CV $\approx$ 0.8–8.1%) and, in S2 and S5, follows the normalised sequence position with a near-linear trend (Area $R^2 \geq 0.99$). For sequences that exhibit more substantial scale changes (S1, S3, S6, S7), the area changes more substantially (Area CV $\approx$ 40–53%, corresponding to max/min factors of approximately 3.6–5.6), yet still evolves smoothly with the sequence index (Area $R^2 \geq 0.98$), indicating gradual rather than oscillatory

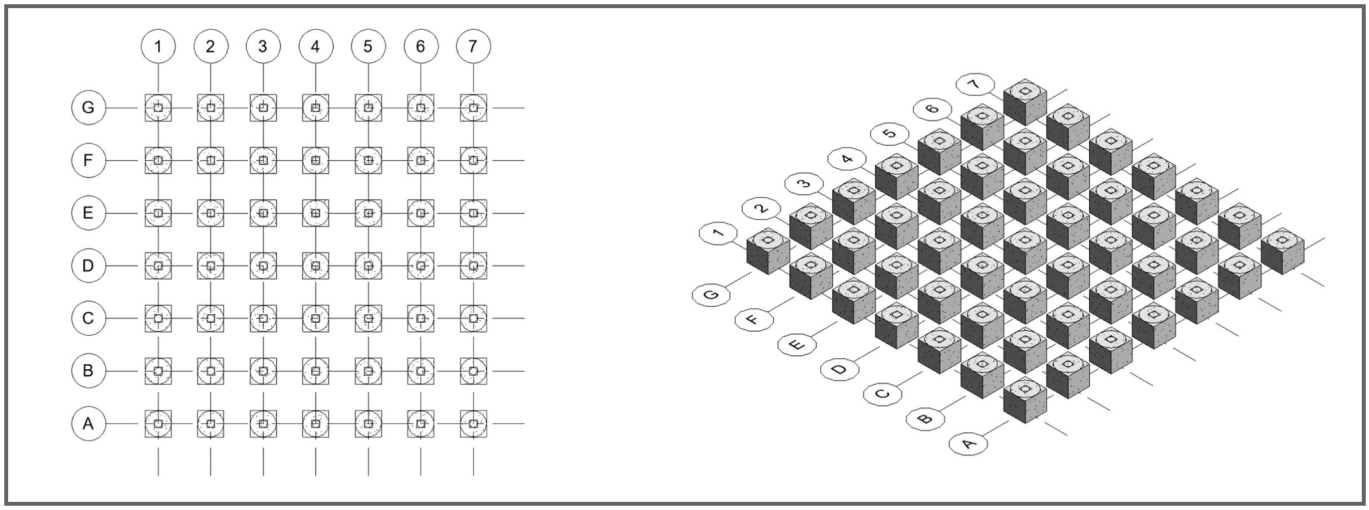

Fig. 15. Example views of the dataset for Experiment 3 in Autodesk Revit, showing floor plans (left) and 3D perspective representations (right). The floor plan grid is arranged at 2 m intervals, providing a consistent spatial reference for column placement.

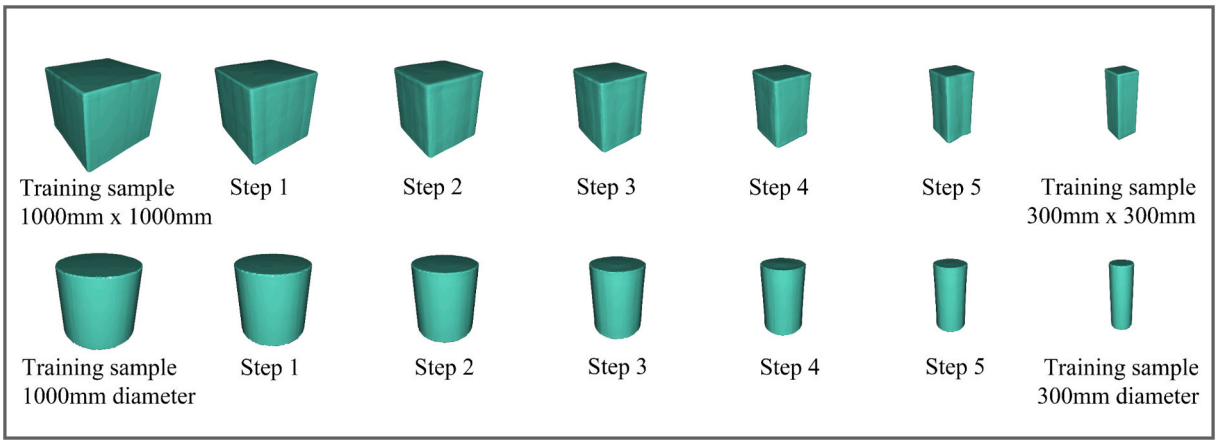

Fig. 16. S1: Size interpolation between square (top) and circular (bottom) columns.

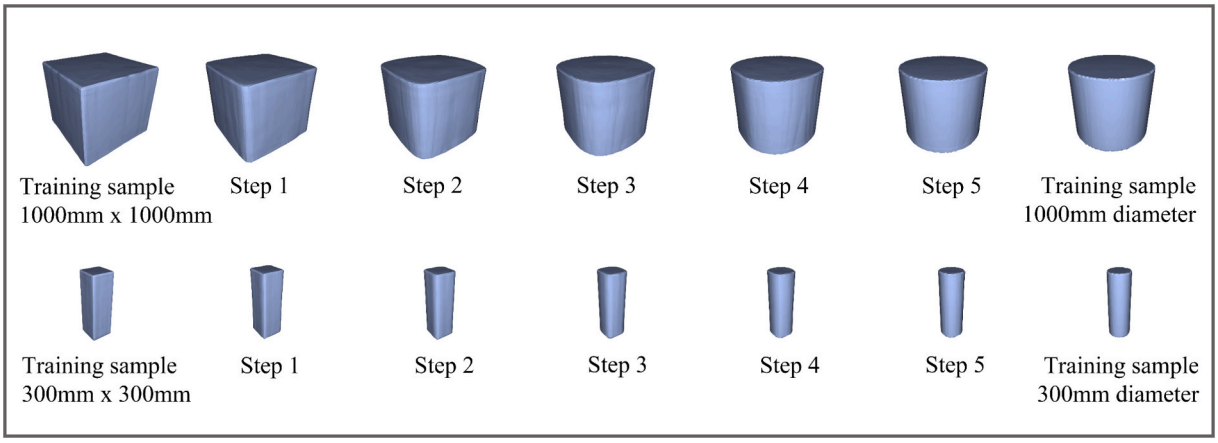

Fig. 17. S2: Shape interpolation between square and circular columns.

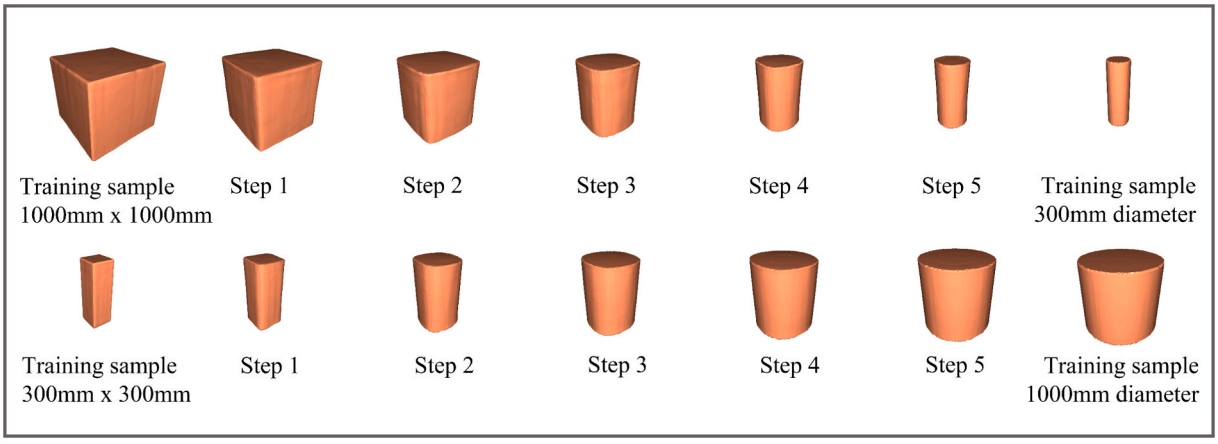

Fig. 18. S3: Shape and size interpolation between square and circular columns.

growth or shrinkage in global size. The apparent outlier is S4: this sequence corresponds to a pure rigid translation, so the surface area is effectively constant (Area CV = 0.84%, Max/Min = 1.02). In this case, there is no meaningful linear trend for the regression to capture, and the resulting low Area R^2 value reflects an almost constant signal rather than irregular interpolation behaviour.

For data-driven modular building design, these interpolation capabilities are particularly valuable. They demonstrate that the obtained vectors not only capture the complex spatial relationships between modular units and components but also map corresponding geometric information in unseen scenarios. In practice, this means that downstream AI models can learn the underlying variation patterns of components across different semantic relationships, thereby establishing a foundation for data-driven component variation generation. The ability to generate meaningful intermediate states through interpolation indicates that the representation space is structured in a way that aligns

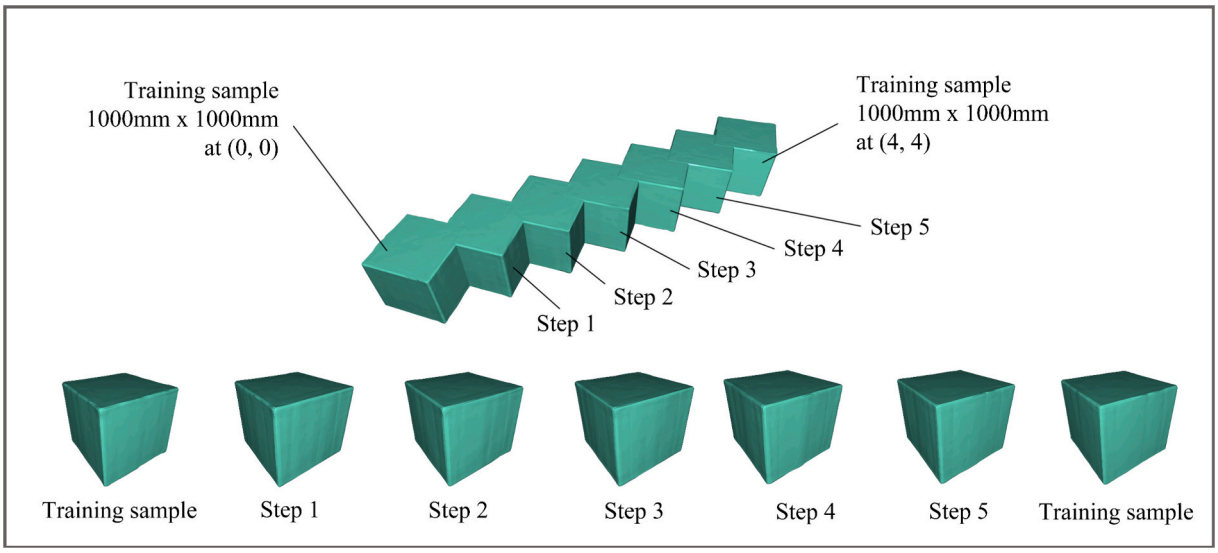

Fig. 19. S4: Position-only interpolation.

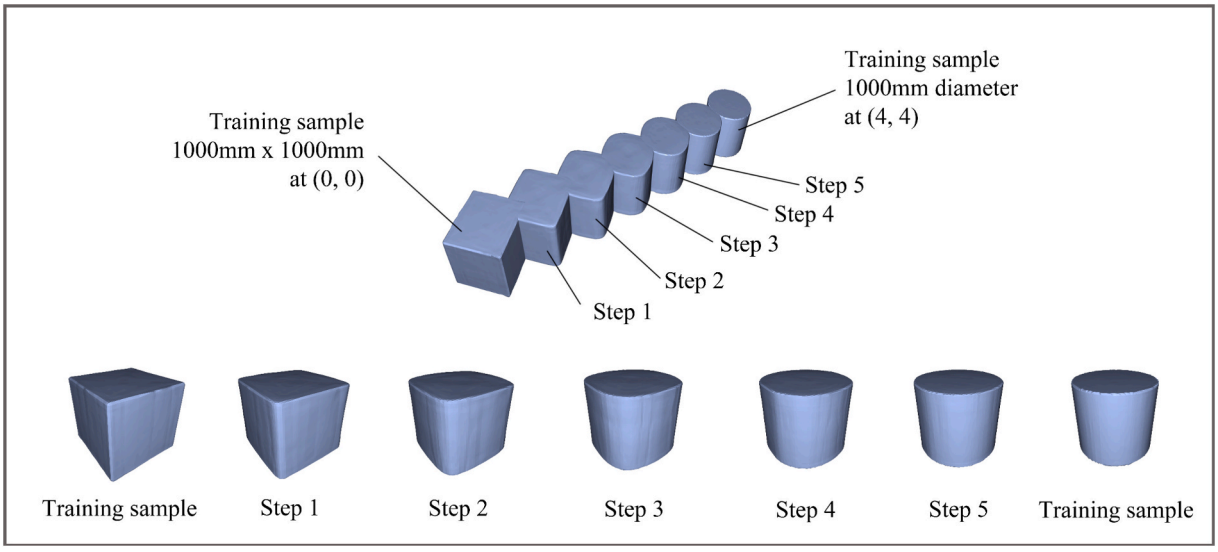

Fig. 20. S5: Position interpolation with combined shape variation.

with architectural design logic, enabling AI systems to reason about and generate contextually appropriate component variations.

4.4. Experiment 4: comparative experiment

To compare implicit and explicit 3D representations, we conducted a preliminary reconstruction experiment using 3D-VAE [54], a 3D variational auto-encoder, as the explicit baseline. We selected 3D-VAE because it is one of the few voxel-based methods that perform both embedding and reconstruction and provides public code for reproducible comparison. To evaluate the generalisability of the proposed approach, we used a subset of component models from the IFCNet dataset [58] as test data. Both methods were evaluated under normalised conditions to ensure a fair comparison. This experiment shows that the proposed implicit auto-decoder achieves higher reconstruction quality than the explicit voxel-based embedding. However, the advantages of the proposed method in preserving spatial relationships and global positioning, which are important for DfMA applications, are not reflected in this normalised single-component comparison.

A total of 60 IfcWall files and 60 IfcSlab files were selected from the IFCNetCore subset of IFCNet. Many IFCNet models contain incomplete or non-watertight solids, so this set represents the portion that can be processed reliably. The meshes were extracted using the same IFC-to-OBJ conversion pipeline as in the earlier experiments. Each object was then translated and scaled into a cube with side lengths of 0.4 metres to provide a consistent normalised input range for both methods. Based on these normalised OBJ files, both the proposed auto-decoder and the 3D-VAE baseline learned embeddings and produced reconstructed meshes. Reconstruction accuracy was measured against the reference meshes using the same protocol as in Experiment 2, which includes Chamfer distance, Hausdorff distance, and surface normal error. The voxelisation step of 3D-VAE proved highly sensitive to its fixed resolution. Many IFCNet components are thin or slender and cannot generate valid occupancy grids, and only 50 components produced successful reconstructions. The quantitative comparison is therefore based on this reduced set. Table 11 summarises the numerical results, and Fig. 23 shows representative reconstructions from both methods.

The quantitative results in Table 11 show that the proposed implicit embedding achieves reconstruction errors that are approximately one order of magnitude lower than those of the explicit 3D-VAE baseline

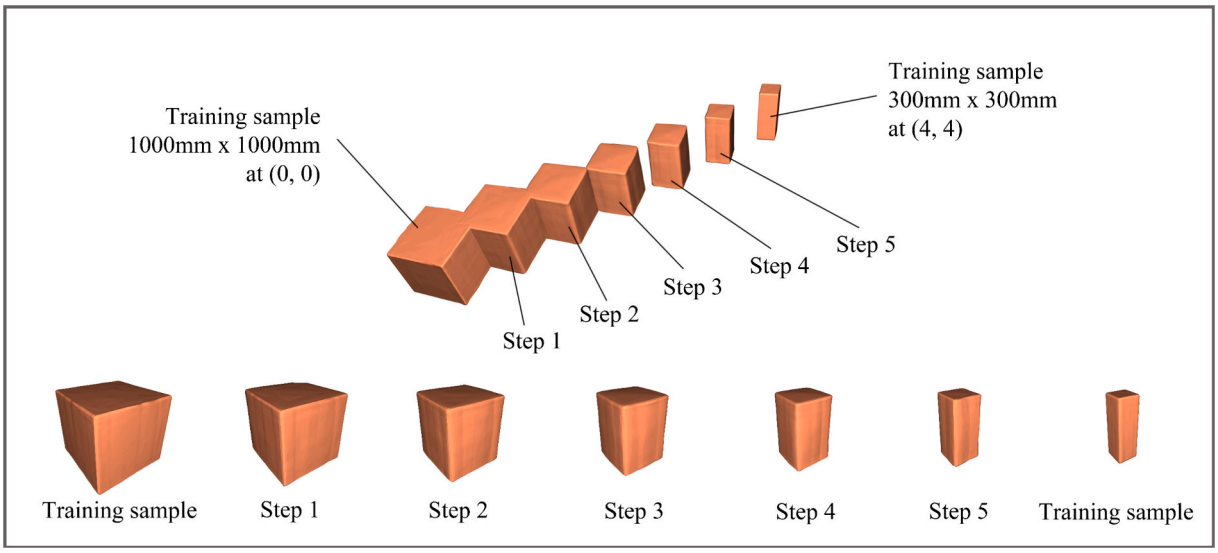

Fig. 21. S6: Position interpolation with combined size variation.

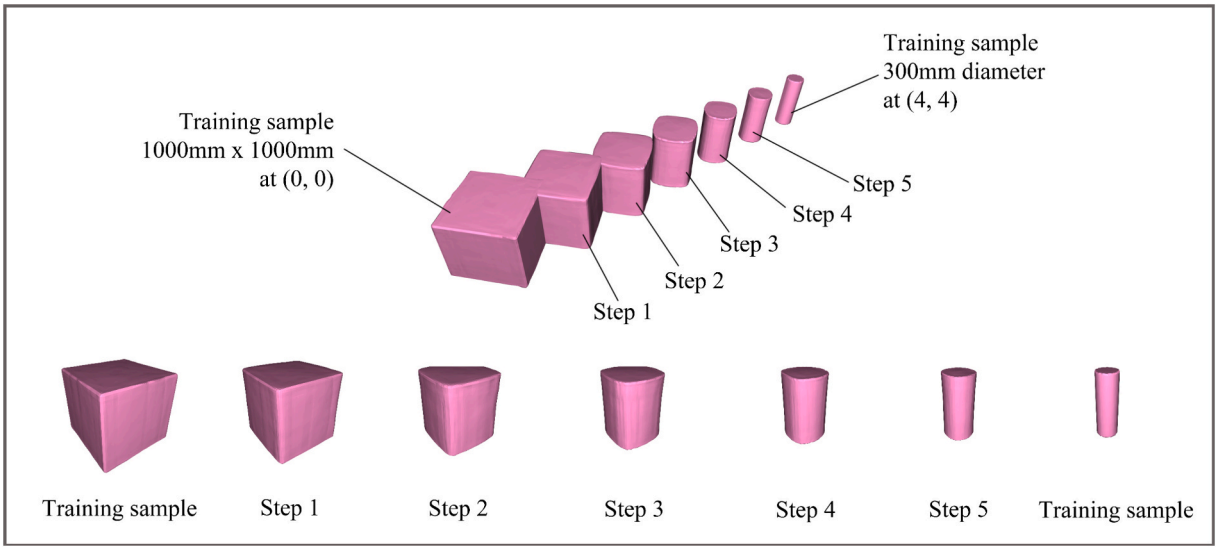

Fig. 22. S7: Combined interpolation of shape, size, and position.

Table 10
Interpolation metrics for Experiment 3.

Sequence	Objects	Step CV (%)	Cum. R^2	Max/ Min	Area R^2	Area CV (%)
S1 top	7	26.55	0.9840	2.1595	0.9938	45.70
S1 bottom	7	25.91	0.9856	2.1666	0.9958	45.02
S2 top	7	1.69	0.9999	1.0507	0.9918	8.22
S2 bottom	7	1.71	0.9999	1.0455	0.9932	7.57
S3 top	7	28.09	0.9830	2.2622	0.9820	52.89
S3 bottom	7	20.41	0.9940	1.8320	0.9951	40.13
S4	7	0.67	0.9999	1.0220	0.0104	0.84
S5	7	1.60	0.9999	1.0470	0.9965	8.08
S6	7	25.97	0.9856	2.1679	0.9928	46.02
S7	7	27.44	0.9843	2.2686	0.9837	52.62

across all three metrics. At the same time, the 3D-VAE retains a clear advantage in runtime: its voxel-based pipeline is substantially faster end-to-end, highlighting that, despite its limited reconstruction fidelity, an explicit embedding can remain a practical choice for downstream AI tasks that require only coarse geometric cues, such as classification.

Table 11
Aggregate per-object reconstruction errors of ours and the 3D-VAE voxel baseline vs. Ground Truth (GT) on the normalised IFCNetCore wall–slab subset. Lower is better.

Scenario	Ours vs GT	3D-VAE vs GT
Chamfer Distance (mm, mean)	2.26	12.45
Hausdorff Distance (mm, mean)	8.17	54.87
Surface Normal Error (deg, mean)	7.65	19.54

Qualitatively, we also observe that the voxel embedding is highly sensitive to its discrete input format: prediction errors often manifest as whole-voxel additions or deletions, resulting in jagged, irregular contours. Increasing the voxel resolution can mitigate these artifacts, but it also increases the data volume cubically, which quickly becomes prohibitive at building scale. By contrast, the proposed implicit representation yields smoother, more faithful surfaces. However, visible defects still occur around sharp or concave features in complex components, for example, at the inner corners of the wall element shown at the bottom row of Fig. 23, indicating that further refinement of the sampling

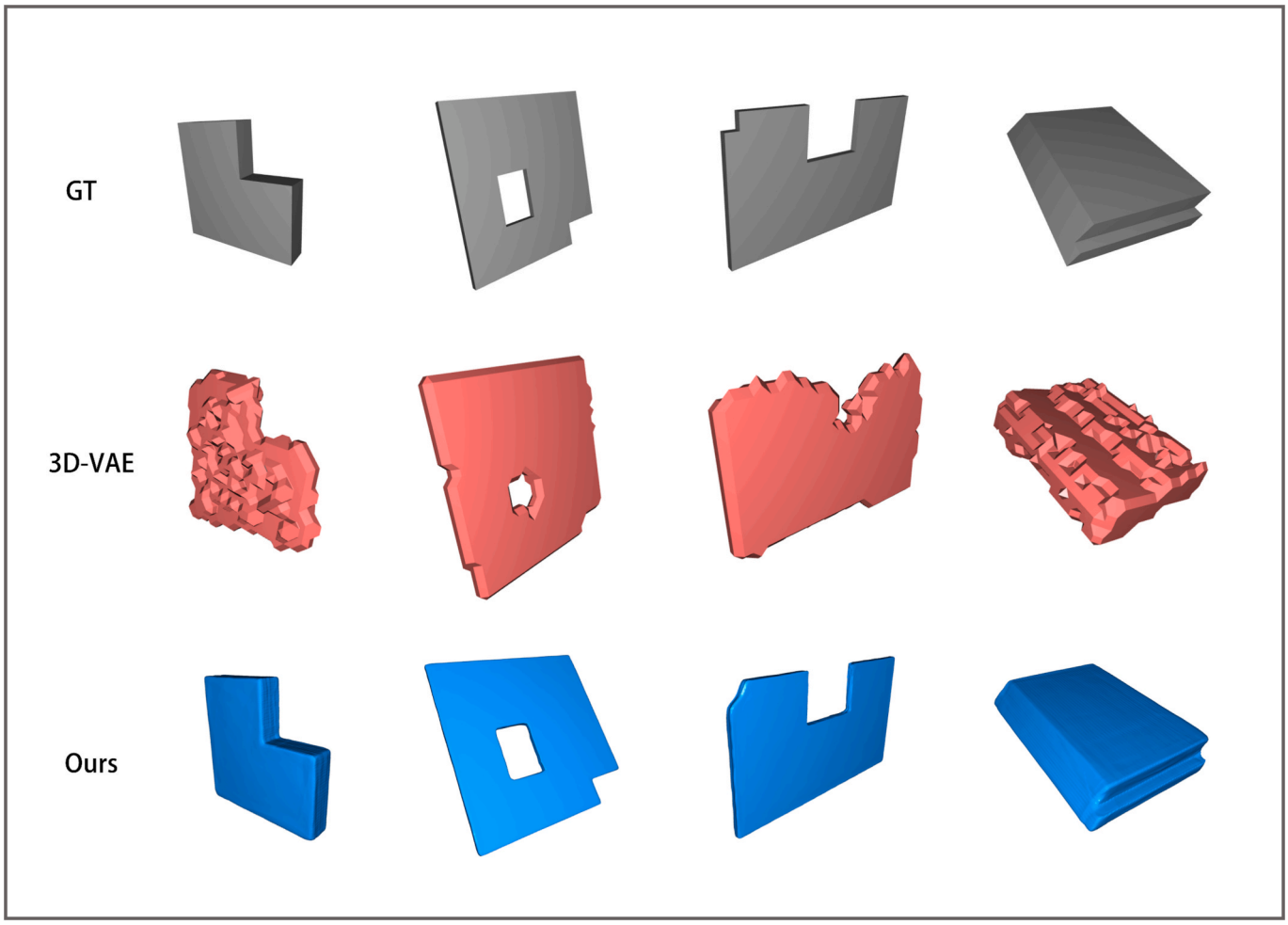

Fig. 23. Qualitative comparison of per-object reconstructions on representative IFCNetCore wall and slab components (normalised to $[-0.2, 0.2]$ m). From top to bottom: ground-truth (GT), the 3D-VAE voxel baseline, and ours illustrating that the proposed implicit representation better preserves sharp profiles and clean surfaces, while the voxel-based baseline exhibits aliasing and boundary artifacts.

strategy is needed.

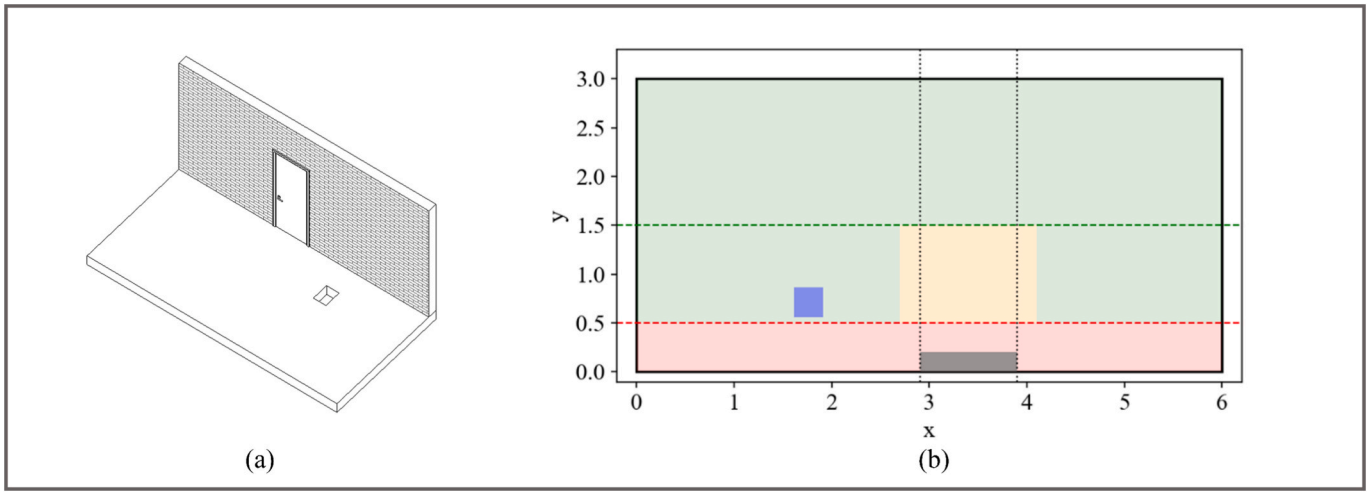

Fig. 24. (a) Revit model of the slab–wall–door configuration used in the experiment. A rectangular vertical opening is placed on the slab surface. (b) Corresponding plan-view representation used for SDF-based rule classification. The red region marks the clash zone (objects with $y < 0.5$ m), the yellow region denotes the near-clearance band around the door in the mid-height range, and the green area represents the safe-far zone. The blue rectangle indicates the position of the opening sampled for this instance.

4.5. Experiment 5: downstream AI application demonstration

To verify that the latent vectors produced by the proposed IFC-to-SDF pipeline can be used directly by downstream AI models, we designed a small demonstration experiment. The objective is to show that a simple neural network can learn implicit design rules solely from latent vectors and perform design checking. The test case is derived from a common situation in modular construction. Vertical service penetrations in the floor slab require slab openings, especially for water pipes. Nearby walls may also contain doors. Door swing and access can conflict with these service openings. Additionally, openings that are too close to the wall base can violate the minimum installation clearance required for lifting and placing the module. The scenario represents a typical cross-component, multi-constraint interaction. Rule-based design checks struggle to exhaustively cover such interactions. However, SDF-based latent vectors enable data-driven learning of these complex relationships.

The illustrative geometry, shown in Fig. 24(a), consists of a single slab, a single wall, and one door. We define three design rules for any given door position on the wall, as illustrated in Fig. 24(b). In the global y direction, we measure the distance from the nearest edge of the floor opening to the wall. When the distance is less than 0.5 m, the configuration is labelled as a module clash case and shown in red. When the distance exceeds 1.5 m, the configuration is labelled as safe. For intermediate cases where the y-direction distance lies between 0.5 m and 1.5 m, an additional horizontal condition is applied. We check whether the horizontal projections of the door and opening overlap. We also measure their separation along the x direction. When the projections do not overlap, and their separation exceeds 0.2 m, the configuration is labelled as safe and shown in green. All remaining cases are labelled as attention required cases and shown in yellow. The yellow region indicates cases where the horizontal projections overlap, or the separation is less than or equal to 0.2 m. Such cases require additional design judgment.

The dataset for the experiment is generated procedurally. The door height is fixed at 2.1 m. The door width is sampled uniformly between 0.8 m and 1.2 m, yielding 50 distinct door variants with random positions along the wall. The floor opening size ranges from 0.2 m $\times$ 0.2 m to 0.6 m $\times$ 0.6 m. We sampled 50 distinct opening variants at random locations on the slab. All 50 door variants are combined with all 50 opening variants, yielding 2,500 unique door and opening configurations. Each configuration is labelled automatically according to the three design rules described above, producing three classes of outcomes: clash, near-clearance, and safe. The labelled dataset is then split into training, validation, and test subsets with a 0.7/0.15/0.15 ratio.

To test whether the latent vectors capture the relevant spatial relationships, we use a simple MLP classifier. The input consists only of the slab's latent codes, including its opening and the door. We adopt a two-tower architecture in which one MLP branch processes the door latent vector, and the other processes the slab latent vector. The outputs of the two branches are concatenated and passed through additional fully connected layers to predict the three design classes. After training, the classifier achieves high accuracy on the held-out test set. Representative quantitative results are reported in Table 12. Qualitative examples of correctly classified configurations are shown in Fig. 25.

The experimental results indicate that a downstream MLP can learn the spatial relationships encoded in the latent vectors. The model discovers the dynamic coupling between door and slab opening positions without access to explicit geometric coordinates or handcrafted features. The model generalises well to unseen combinations of door and opening geometries and achieves high classification accuracy for new spatial configurations. In practical applications, similar pipelines can be used when as-built modular projects are available as IFC models. The pipeline can encode modules and their penetrations into latent vectors and train AI models that learn from past adjustment decisions made to satisfy on-site installation constraints. Such trained models could then provide data-driven feedback on new modular designs, supporting designers with limited construction experience in checking clearances and avoiding cross-component conflicts.

Table 12
Classification performance of the slab–door rule-learning classifier on the test set.

Label	Total sample	Correct predictions	Accuracy (%)
Safe	273	268	98.2
Near clearance	59	52	88.1
Clash	45	45	100.0
Overall	377	365	96.8

5. Discussion

5.1. Challenges in dataset availability

Although our IFC sources originate from completed modular projects, contractual restrictions and heterogeneous modelling conventions prevent them from constituting a public, standardised benchmark. What is currently scarce is not raw BIM data, but shareable datasets that consistently preserve DfMA-critical semantics for modular construction. These semantics include inter-unit spatial relationships for site assembly, such as alignment frames, interface coordinates, and adjacency graphs. They also include standardised volumetric configurations and manufacturing constraints that shape module families and allowable parameter ranges. Additionally, they capture repetitive patterns and in-family variants reflecting prefabrication economies of scale, as well as connection and assembly metadata that encode constructability through sequence, roles, and tolerances. Since existing public datasets [58–60] provide context-stripped, normalised single objects without these DfMA semantics, benchmarking spatial-semantic preservation against established baselines in a modular setting is not feasible. A task-specific corpus is therefore curated from real projects, and key DfMA fields are normalised to enable modular-aware evaluation. This study also advocates the development of benchmark-grade, DfMA-aware modular datasets with transparent licensing, consistent schemas and task-aligned splits across typologies such as residential towers, student housing and hotels, to support robust AI training for industrialised construction.

5.2. Implications for AI-driven modular building design

This study opens new possibilities for data-driven modular building design. Traditional constraint-based methods, such as parametric configurators and rule-based optimisation, struggle when many parameters, spatial relationships and fabrication requirements interact in ways that are hard to formalise. The proposed project coordinate SDF representation enables AI models to learn from completed modular projects by encoding both module geometry and DfMA-informed spatial context.

The proposed methodology does not require all design rules to be predefined. It allows AI systems to discover regularities in volumetric coordination, connections between units and assembly logic directly from historical IFC models. In a DfMA setting, manufacturability, transport constraints, and on-site tolerances must be satisfied simultaneously. The learned representation works with explicit constraints. Explicit rules enforce regulatory and engineering requirements, and the latent space suggests alternatives similar to successful, constructable precedents. To make this more concrete, three application directions are highlighted below.

- Optimisation of connections and interfaces between units.

The latent vectors preserve spatial relationships between adjacent modules and their interface zones, so AI models can suggest alternative connection details or interface layouts that remain compatible with alignment frames, service corridors and tolerance envelopes.

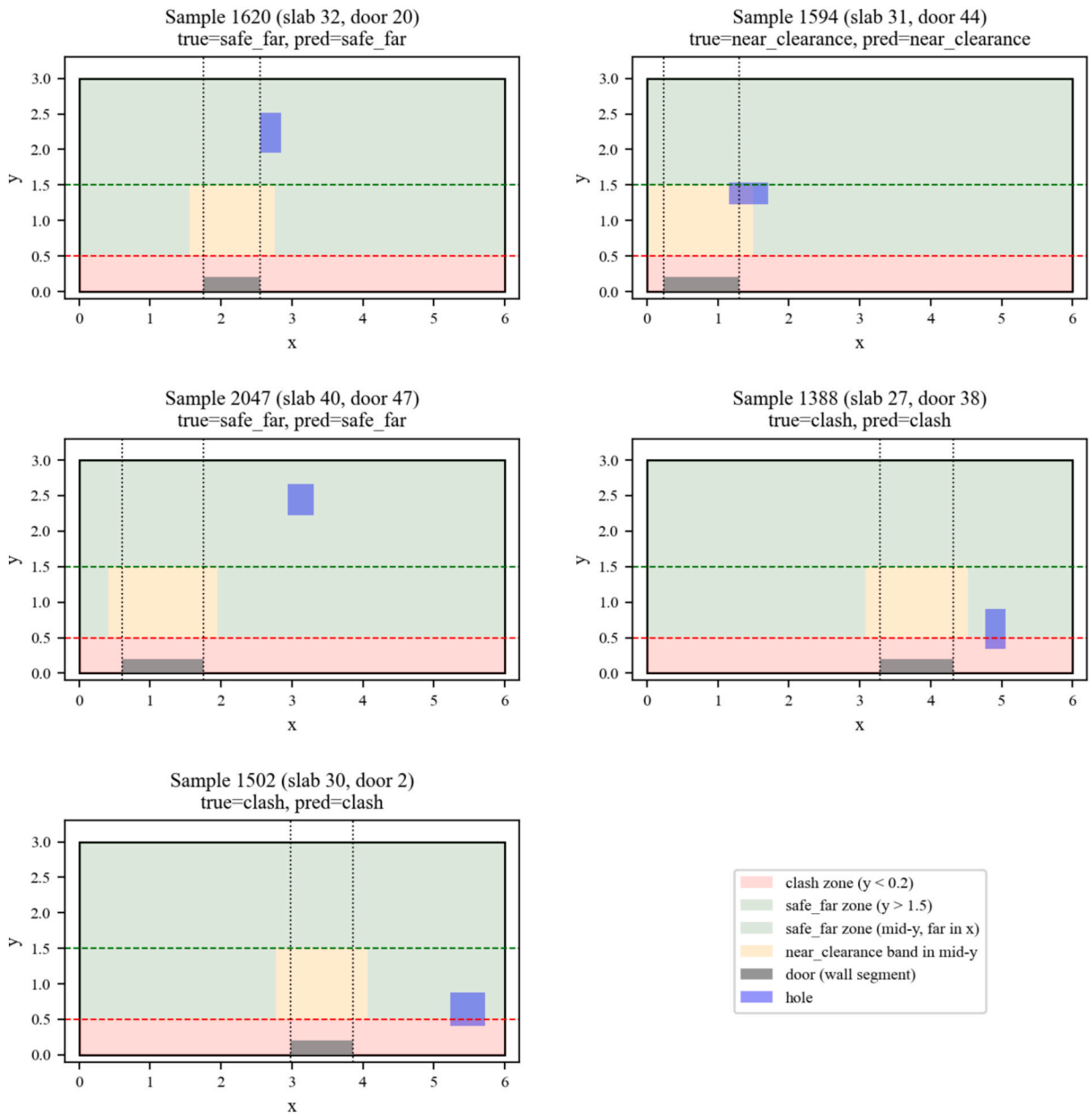

Fig. 25. Representative plan-view samples from the slab–door dataset showing the rule-based labelling and MLP predictions. Each subplot depicts a slab with a vertical hole (blue) and a door segment along the wall (grey). The red band marks the clash zone ($y < 0.5$), the light green region marks the safe_far zone ($y > 1.5$), and the mid-height band is split into a near_clearance region (yellow) and a safe_far region when the door is sufficiently far in x. Vertical dashed lines indicate the x-gap thresholds used in the labelling rules. Titles report the sample index and slab/door IDs, together with the ground-truth label and the model prediction, illustrating that the classifier recovers the intended geometric rules for these examples.

- Adaptive module arrangement and assembly sequencing.

Because modules share the project coordinate frame, the model can reason about how layout changes affect assembly feasibility and can propose stacking or placement schemes that respect lifting constraints, site access and erection sequences.

- Design space exploration for mass customisation.

Smooth interpolation in latent space allows designers to explore families of module variants while preserving DfMA-critical properties, such as standardised volumetric envelopes and repeatable connection patterns, ensuring customised layouts remain close to manufacturable, validated solutions.

5.3. *Limitations and future work*

This study has a few limitations within its current DfMA-oriented scope. First, because typical modular units prioritise planar forms for manufacturing efficiency, the proposed experiments focused on rectangular rooms without curved surfaces or complex geometry. While the proposed method can be extended to these geometric types, such validation was not conducted in this study. Future work will examine modules with curved or non-orthogonal envelopes. Second, large-scale training is restricted by practical implementation challenges. The proposed experiments successfully processed models with up to 704 objects and 46 million sample points, demonstrating scalability for small modular projects. However, real-world engineering applications would require substantially larger datasets, necessitating further investigation

into computational optimisation strategies. Third, our findings are based on single runs that demonstrate feasibility, rather than statistical benchmarking. The study serves as a proof of concept, and we will conduct multi-run evaluations in future work to quantify the variance. Finally, although the proposed tool is readily deployable, integrating it with downstream AI applications still involves environment configuration, compute provisioning, and workflow integration, especially for teams without an established machine learning infrastructure.

Future work will prioritise three key directions aligned with these limitations. First, the development and release of a spatial-context-preserving BIM dataset is planned to enable standardised benchmarking for similar research endeavours. Second, the practical implementation of the method will be enhanced by optimising training procedures to reduce computational time for large-scale datasets to engineering-viable timeframes. Finally, multi-run evaluations will be conducted in subsequent studies to quantify variance and establish statistical robustness.

6. Conclusion

This paper addresses the representation gap between IFC geometry and the data requirements of AI-driven modular building design in a DfMA setting. This study presented an auto-decoder based on SDF that converts IFC component geometry into unified, fixed-length latent vectors while retaining spatial context. To support practical use, the method was further implemented as a web-based, readily deployable tool that operationalises the workflow from IFC export through SDF sampling, training and octree-based reconstruction, making the pipeline accessible to practitioners and researchers.

Experiments demonstrate practical feasibility for small-scale modular projects, with training as the dominant computational cost. By preserving project coordinates that are essential for spatial reasoning about inter-component relationships, the method achieves reconstruction accuracy sufficient for modular building design and DfMA-oriented analysis. An ablation study with conventional normalisation shows that removing spatial context severely degrades accuracy, which confirms the necessity of coordinate preservation. Interpolation experiments demonstrate smooth transitions across size, shape and position. The monotonic progression in deformation confirms that the learned representations support component variant generation rather than merely memorising surfaces and provide a basis for AI-driven generation of contextually appropriate variants for prefabricated modules that remain compatible with assembly and fabrication constraints.

Taken together, these results indicate that the proposed method effectively bridges IFC entities and continuous geometric representations for downstream AI applications in a DfMA setting. In practical modular building design workflows, this enables tasks such as automated layout generation, component variant generation and optimisation of connections and interfaces between units. The study remains limited by its focus on rectangular geometries typical of prefabricated modules, computational challenges for large-scale applications, reliance on single-run validation, and the effort required to integrate the tool with downstream AI systems. Future work will focus on developing spatial-context-preserving BIM datasets for standardised benchmarking, improving computational efficiency to achieve engineering-viable training times, and conducting multi-run evaluations to establish statistical robustness. Additional work will also explore modular-specific functionalities, such as learning assembly rules and connection logic from historical projects.

CRediT authorship contribution statement

Sang Du: Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. **Lei Hou:** Writing – review & editing, Validation, Supervision. **Guomin (Kevin) Zhang:** Project administration, Resources, Supervision. **Yang Zou:** Writing – review & editing. **Haosen Chen:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

Table A.1
. IFC solid geometry representation types [17].

Representation	Parameters	Reconstruction Method:
Swept Solid	2D profile curve:$P(u) \in \mathbb{R}^2$ Extrusion direction: $\vec{d} \in \mathbb{R}^3$ Extrusion height:$h \in \mathbb{R}$ Revolution angle:$\theta \in [0, 2\pi]$	Extrusion : $S(u,t) = T(P(u)) + t \bullet \vec{d}$, $t \in [0, h]$Revolution: $S(u,\theta) = R(\theta) \bullet T(P(u))$
Advanced Swept Solid	2D profile curve:$P(u) \in \mathbb{R}^2$ Directrix curve:C$(t) \in \mathbb{R}^3$ Orientation frame:$R(t) \in SO(3)$ Optional tapering scale:$S(t) \in \mathbb{R}$	$S(u,t) = R(t) \bullet s(t) \bullet P(u) + C(t)$
Brep	Vertices:$\{V_i \in \mathbb{R}^3\}$ Topology:{*faces as loops of* V_i} Euler-valid manifold	S $= \bigcup_i PlanarFace(V_{i1}, V_{i2}, \cdots, V_{in})$
Advanced Brep	Parametric surface:$S(u,v) : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$ Trimming curves:$\gamma(u)$ Control points P_{ij}, knot vectors, weights	$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_i^p(u) M_j^q(v) \omega_{ij} P_{ij}$
CSG	Primitive solids:$A, B, \cdots$ Boolean operation tree T: nodes as $\cap, \cup, -$	$S = A \odot B$, $\odot \in \{\cap, \cup, -\}$, *evaluated recursively*
Clipping	Solid geometry A Half-space $H = x \vert \vec{n} \bullet (x - p) \geq 0$	$S = A \bigcap H$

Table A.2
. Hardware configuration

Component	Specifications
System	Ubuntu 24.04.2 LTS
CPU	Intel Core i5-13400
GPU	NVIDIA RTX 4060 Ti (16GB VRAM)
RAM	128 GB

Table A.3
. SDF sampling parameters

Parameter	Value (after scaling)	Notes
Scale factor	0.1	Uniform scaling before sampling
Minimal sample points per surface	100	Ensures sufficient local coverage
Surface sample density	5000 pts/m^2	
Random points around surface	10 (r = 0.01 m) 8 (r = 0.05 m) 6 (r = 0.20 m) 2 (r = 1.00 m)	For near/far-field SDF sampling

Table A.4
. Auto-decoder model parameters

Parameter	Value
Network depth	8 layers
Hidden dimension	512
Representation vector size	128
Learning rate	0.0007
Sigma	0.001
Training steps	1000

Table A.5
Decoder reconstruction parameters

Parameter	Value (after scaling)
Overall bounding box	((-2.0 m, 2.0 m), (-2.0 m, 2.0 m), (-1.0 m, 3.0 m))
Global grid resolution	256
First-level block resolution	8
Leaf voxel resolution	16
Max depth	5

Data availability

Data will be made available on request.

References

[1] C. Widanage, K.P. Kim, Integrating Design for Manufacture and Assembly (DfMA) with BIM for infrastructure, Autom. Constr. 167 (2024) 105705, https://doi.org/10.1016/j.autcon.2024.105705.

[2] C. Rojas-Herrera, A. Martínez-Soto, C. Avendaño-Vera, R.C. Carrasco, N.R. Barbato, Industrialized construction: a systematic review of its benefits and guidelines for the development of new constructive solutions applied in sustainable projects, Appl. Sci. 15 (2025) 2308, https://doi.org/10.3390/app15052308.

[3] M. Zohourian, A. Pamidimukkala, S. Kermanshachi, D. Almaskati, Modular construction: a comprehensive review, Buildings 15 (2025) 2020, https://doi.org/10.3390/buildings15122020.

[4] A.A. Khan, R. Yu, T. Liu, N. Gu, J. Walsh, Volumetric modular construction risks: a comprehensive review and digital-technology-coupled circular mitigation strategies, Sustainability 15 (2023) 7019, https://doi.org/10.3390/su15087019.

[5] M. Lawson, R. Ogden, C. Goodier, Design in Modular Construction, 1st ed., CRC Press, Boca Raton, FL, 2014. https://doi.org/10.1201/b16607.

[6] B. Llave, I. Iordanova, Digital configuration platforms for residential modular construction projects: A comparative study, in: Int. Conf. Comput. Civ. Build. Eng., Springer, 2024: pp. 298–313. https://doi.org/10.1007/978-3-031-84208-5_24.

[7] L.R. Sokhangoo, M. Nik-Bakht, S. Han, Optimal Space Layout Generation for Modular Offsite Construction for Whole Life Cycle Costs—Gaps and Requirements, in: Can. Soc. Civ. Eng. Annu. Conf., Springer, 2023: pp. 395–411. https://doi.org/10.1007/978-3-031-62170-3_28.

[8] Z. Fan, J. Liu, L. Wang, G. Cheng, M. Liao, P. Liu, Y.F. Chen, Automated layout of modular high-rise residential buildings based on genetic algorithm, Autom. Constr. 152 (2023) 104943, https://doi.org/10.1016/j.autcon.2023.104943.

[9] A.C. Narváez, E.C. Reyes, M.R. Peña, J.S. Ramos, Integrating Modularity into Industrialization and Prefabrication of Sustainable Residential Housing Solutions, in: Int. Conf. Digit. Transform. Graph. Eng., Springer, 2023: pp. 259–269. https://doi.org/10.1007/978-3-031-51623-8_25.

[10] J. Liu, Y. Xue, H. Ni, R. Yu, Z. Zhou, S.X. Huang, Computer-aided layout generation for building design: A review, Comput. Vis, Media 11 (2025) 677–707. https://doi.org/10.26599/CVM.2025.9450484.

[11] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, Cambridge, MA, 2016 http://www.deeplearningbook.org.

[12] H. Abdirad, P. Mathur, Artificial intelligence for BIM content management and delivery: Case study of association rule mining for construction detailing, Adv. Eng. Inform. 50 (2021) 101414, https://doi.org/10.1016/j.aei.2021.101414.

[13] A.B. Saka, L.O. Oyedele, L.A. Akanbi, S.A. Ganiyu, D.W. Chan, S.A. Bello, Conversational artificial intelligence in the AEC industry: A review of present status, challenges and opportunities, Adv. Eng. Inform. 55 (2023) 101869, https://doi.org/10.1016/j.aei.2022.101869%2520%2520Copy%2520to%2520clipboard.

[14] R. Panahi, J. Louis, A. Podder, C. Swanson, S. Pless, Bottleneck detection in modular construction factories using computer vision, Sensors 23 (2023) 3982, https://doi.org/10.3390/s23083982.

[15] P. Debus, J. Valencia, Evaluation of Surface Quality in the Prefabrication of Concrete Panels using Computer Vision, Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 48 (2025) 367–374, https://doi.org/10.5194/isprs-archives-XLVIII-G-2025-367-2025.

[16] S. Du, L. Hou, G. Zhang, Y. Tan, P. Mao, BIM and IFC data readiness for AI integration in the construction industry: a review approach, Buildings 14 (2024) 3305, https://doi.org/10.3390/buildings14103305.
[17] BuildingSMART International, Industry Foundation Classes (IFC4) – ISO 16739-1: 2018, International Organization for Standardization, 2018. https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/.
[18] S. Bakhshi, M.R. Chenaghlou, F.P. Rahimian, D.J. Edwards, N. Dawood, Integrated BIM and DfMA parametric and algorithmic design based collaboration for supporting client engagement within offsite construction, Autom. Constr. 133 (2022) 104015, https://doi.org/10.1016/j.autcon.2021.104015.
[19] E.V. Kalemi, F. Cheung, A.-R.H. Tawil, P. Patlakas, K. Alyania, ifcOWL-DfMA a new ontology for the offsite construction domain, in: LDAC 2020-Proc. 8th Linked Data Archit. Constr. Workshop, 2020. https://www.open-access.bcu.ac.uk/id/eprint/10706.
[20] G. Chen, A. Alsharef, A. Ovid, A. Albert, E. Jaselskis, Meet2Mitigate: An LLM-powered framework for real-time issue identification and mitigation from construction meeting discourse, Adv. Eng. Inform. 64 (2025) 103068, https://doi.org/10.1016/j.aei.2024.103068.
[21] D. Chen, L. Chen, Y. Zhang, S. Lin, M. Ye, S. Sølvsten, Automated fire risk assessment and mitigation in building blueprints using computer vision and deep generative models, Adv. Eng. Inform. 62 (2024) 102614, https://doi.org/10.1016/j.aei.2024.102614.
[22] J.W.L. Shi, W. Solihin, J.K. Yeoh, Fine-tuning a large language model for automated code compliance of building regulations, Adv. Eng. Inform. 68 (2025) 103676, https://doi.org/10.1016/j.aei.2025.103676.
[23] L. Kumi, J. Jeong, J. Jeong, Enhancing construction safety compliance through a blockchain-enabled worker certification management system, Adv. Eng. Inform. 68 (2025) 103784, https://doi.org/10.1016/j.aei.2025.103784.
[24] H. You, T. Zhou, Q. Zhu, Y. Ye, E.J. Du, Embodied AI for dexterity-capable construction Robots: DEXBOT framework, Adv. Eng. Inform. 62 (2024) 102572, https://doi.org/10.1016/j.aei.2024.102572.
[25] D. Lee, S. Lee, N. Masoud, M. Krishnan, V.C. Li, Digital twin-driven deep reinforcement learning for adaptive task allocation in robotic construction, Adv. Eng. Inform. 53 (2022) 101710, https://doi.org/10.1016/j.aei.2022.101710.
[26] J. Liu, Z. Qiu, L. Wang, P. Liu, G. Cheng, Y. Chen, Intelligent floor plan design of modular high-rise residential building based on graph-constrained generative adversarial networks, Autom. Constr. 159 (2024) 105264, https://doi.org/10.1016/j.autcon.2023.105264.
[27] P. Ghannad, Y.-C. Lee, Automated modular housing design using a module configuration algorithm and a coupled generative adversarial network (CoGAN), Autom. Constr. 139 (2022) 104234, https://doi.org/10.1016/j.autcon.2022.104234.
[28] X. Liu, C. Hou, J. Peng, Y. Wang, K.J. Rasmussen, Recent advancements of inter-module connections for modular buildings: An overview and multi-dimensional assessment, Eng. Struct. 335 (2025) 120378, https://doi.org/10.1016/j.engstruct.2025.120378.
[29] Y. Chua, J.R. Liew, S. Pang, Modelling of connections and lateral behavior of high-rise modular steel buildings, J. Constr. Steel Res. 166 (2020) 105901, https://doi.org/10.1016/j.jcsr.2019.105901.
[30] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1798–1828, https://doi.org/10.1109/TPAMI.2013.50.
[31] R. Wu, C. Xiao, C. Zheng, Deepcad: A deep generative network for computer-aided design models, in: Proc. IEEECVF Int. Conf. Comput. Vis., 2021: pp. 6772–6782. https://doi.org/10.1109/ICCV48922.2021.00670.
[32] S. Kim, M. Peavy, P.-C. Huang, K. Kim, Development of BIM-integrated construction robot task planning and simulation system, Autom. Constr. 127 (2021) 103720, https://doi.org/10.1016/j.autcon.2021.103720.
[33] A. Zhu, P. Pauwels, B. De Vries, Component-based robot prefabricated construction simulation using IFC-based building information models, Autom. Constr. 152 (2023) 104899, https://doi.org/10.1016/j.autcon.2023.104899.
[34] O. Poursaeed, M. Fisher, N. Aigerman, V.G. Kim, Coupling explicit and implicit surface representations for generative 3d modeling, in: Eur. Conf. Comput. Vis., Springer, 2020: pp. 667–683. https://doi.org/10.1007/978-3-030-58607-2_39.
[35] Y. Siddiqui, A. Alliegro, A. Artemov, T. Tommasi, D. Sirigatti, V. Rosov, A. Dai, M. Nießner, Meshgpt: Generating triangle meshes with decoder-only transformers, in: Proc. IEEECVF Conf. Comput. Vis. Pattern Recognit., 2024: pp. 19615–19625. https://doi.org/10.1109/CVPR52733.2024.01855.
[36] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017: pp. 652–660. https://doi.org/10.1109/CVPR.2017.16.
[37] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Trans. Graph. TOG 38 (2019) 1–12, https://doi.org/10.1145/3326362.
[38] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, M. Chen, Point-e: A system for generating 3d point clouds from complex prompts, ArXiv Prepr. ArXiv221208751 (2022). https://doi.org/10.48550/arXiv:2212.08751.
[39] D. Maturana, S. Scherer, Voxnet: A 3d convolutional neural network for real-time object recognition, in: IEEERSJ Int. Conf. Intell. Robots Syst. IROS, Ieee, 2015: pp. 922–928. https://doi.org/10.1109/IROS.2015.7353481.
[40] K. Schwarz, A. Sauer, M. Niemeyer, Y. Liao, A. Geiger, Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids, Adv. Neural Inf. Process. Syst. 35 (2022) 33999–34011. https://doi.org/10.48550/arXiv:2206.07695.
[41] E. Sella, G. Fiebelman, P. Hedman, H. Averbuch-Elor, Vox-e: Text-guided voxel editing of 3d objects, in: Proc. IEEECVF Int. Conf. Comput. Vis., 2023: pp. 430–440. https://doi.org/10.1109/ICCV51070.2023.00046.
[42] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, D. Cohen-Or, Meshcnn: a network with an edge, ACM Trans. Graph. TOG 38 (2019) 1–12, https://doi.org/10.1145/3306346.3322959.
[43] Y. Feng, Y. Feng, H. You, X. Zhao, Y. Gao, Meshnet: Mesh neural network for 3d shape representation, in: Proc. AAAI Conf. Artif. Intell., 2019: pp. 8279–8286. https://doi.org/10.1609/aaai.v33i01.33018279.
[44] X. Zhou, C. Ma, M. Wang, M. Guo, Z. Guo, X. Liang, J. Han, BIM product recommendation for intelligent design using style learning, J. Build. Eng. 73 (2023) 106701, https://doi.org/10.1016/j.jobe.2023.106701.
[45] B. Koo, R. Jung, Y. Yu, Automatic classification of wall and door BIM element subtypes using 3D geometric deep neural networks, Adv. Eng. Inform. 47 (2021) 101200, https://doi.org/10.1016/j.aei.2020.101200.
[46] L. Xu, X. Feng, J. Liu, H. Qi, Y. Wang, Automatic defect localization method based on BIM projected image, Adv. Eng. Inform. 69 (2025) 103874, https://doi.org/10.1016/j.aei.2025.103874.
[47] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, W. Wang, Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction, ArXiv Prepr, ArXiv210610689 (2021). https://doi.org/10.48550/arXiv:2106.10689.
[48] J.J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, in: 2019 IEEECVF Conf. Comput. Vis. Pattern Recognit. CVPR, IEEE, Long Beach, CA, USA, 2019: pp. 165–174. https://doi.org/10.1109/CVPR.2019.00025.
[49] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, A. Geiger, Occupancy networks: Learning 3d reconstruction in function space, in: Proc. IEEECVF Conf. Comput. Vis. Pattern Recognit., 2019: pp. 4460–4470. https://doi.org/10.1109/CVPR.2019.00459.
[50] J. Chibane, T. Alldieck, G. Pons-Moll, Implicit functions in feature space for 3d shape reconstruction and completion, in: Proc. IEEECVF Conf. Comput. Vis. Pattern Recognit., 2020: pp. 6970–6981. https://doi.org/10.1109/CVPR42600.2020.00700.
[51] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: Proc. 23rd Annu. Conf. Comput. Graph. Interact. Tech., 1996: pp. 303–312. https://doi.org/10.1145/237170.237269.
[52] A. Shehadeh, O. Alshboul, M.M. Taamneh, A.Q. Jaradat, A.H. Alomari, Enhanced clash detection in building information modeling: Leveraging modified extreme gradient boosting for predictive analytics, Results Eng. 24 (2024) 103439, https://doi.org/10.1016/j.rineng.2024.103439.
[53] L. Huang, R. Pradhan, S. Dutta, Y. Cai, BIM4D-based scheduling for assembling and lifting in precast-enabled construction, Autom. Constr. 133 (2022) 103999, https://doi.org/10.1016/j.autcon.2021.103999.
[54] A. Brock, T. Lim, J.M. Ritchie, N. Weston, Generative and discriminative voxel modeling with convolutional neural networks, ArXiv Prepr. ArXiv160804236 (2016). https://doi.org/10.48550/arXiv.1608.04236.
[55] C. Xu, M. Mielle, A. Laborde, A. Waseem, F. Forest, O. Fink, Exploiting semantic scene reconstruction for estimating building envelope characteristics, Build. Environ. 275 (2025) 112731, https://doi.org/10.1016/j.buildenv.2025.112731.
[56] B. Mildenhall, P.P. Srinivasan, M. Tancik, J.T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis, Commun. ACM 65 (2021) 99–106. https://doi.org/10.48550/arXiv.2003.08934.
[57] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P.P. Srinivasan, J.T. Barron, H. Kretzschmar, Block-nerf: Scalable large scene neural view synthesis, in: Proc. IEEECVF Conf. Comput. Vis. Pattern Recognit., 2022: pp. 8248–8258. https://doi.org/10.1109/CVPR52688.2022.00807.
[58] C. Emunds, N. Pauen, V. Richter, J. Frisch, C. Van Treeck, IFCNet: A benchmark dataset for IFC entity classification, in: EG-ICE 2021 Workshop Intell. Comput. Eng., Universitätsverlag der TU Berlin, 2021. https://doi.org/10.48550/arXiv:2106.09712.
[59] J. Wu, T. Akanbi, J. Zhang, Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification, J. Comput. Civ. Eng. 36 (2022) 04022008, https://doi.org/10.1061/(ASCE)CP.1943-5487.0001012.
[60] J. Wu, J. Zhang, New automated BIM object classification method to support BIM interoperability, J. Comput. Civ. Eng. 33 (2019) 04019033, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000858.