Contents lists available at ScienceDirect

Automation in Construction

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

Fully automated synthetic BIM dataset generation using a deep learning-based framework

Xing Liang [a], Nobuyoshi Yabuki [b,c], Tomohiro Fukuda [a,*]

[a] Division of Sustainable Energy and Environmental Engineering, Graduate School of Engineering, The University of Osaka, Osaka, 565-0871, Japan
[b] Advanced Research Laboratories, Tokyo City University, Tokyo, 158-8557, Japan
[c] The University of Osaka, Osaka, Japan

ARTICLE INFO

Keywords:
2D-to-BIM reconstruction
Floor plan generation
Synthetic BIM dataset
BIM
Automation
Data-driven modeling
Deep learning
Image processing

ABSTRACT

Building information models (BIMs) are essential for efficient building operation, yet most existing buildings only have two-dimensional (2D) drawings, leading to increased interest in 2D-to-BIM reconstruction. To address the data scarcity hindering automated BIM reconstruction and evaluation, this paper presents a deep learning-based fully automated framework for BIM dataset generation. The approach uses image processing to define polygonal boundaries, applies neural networks to generate geometric layouts, and augments semantic information with predefined data for BIM generation via software application programming interfaces (APIs). The resulting Residential unit BIM (ResBIM) is a synthetic dataset comprising over 1000 paired BIMs (RVT format) and their corresponding 2D floor plans automatically annotated via a toolbox, filling a critical gap in BIM data availability. This work provides a scalable automated BIM reconstruction solution and establishes the foundation for future AI-driven BIM automation research.

1. Introduction

The proportion of global energy consumption attributable to the building sector has steadily increased and now accounts for around 40 % [1]. The largest portion of building energy consumption occurs during the operation and maintenance (O&M) stage, which is the longest stage of the building lifecycle [2,3]. Traditional methods of managing buildings during the O&M stage—such as using two-dimensional (2D) drawings or spreadsheets—often result in inefficiencies and coordination challenges due to error-prone and labor-intensive design adjustments on drawings [4,5]. Recently, research interest has increasingly focused on developing three-dimensional (3D) models for existing buildings, particularly building information models (BIMs). Compared to 2D drawings, a BIM provides a more comprehensive representation of a building's physical and functional characteristics, leading to increased work efficiency, better decision-making, and reduced energy consumption [6]. This enhancement is attributed to the capabilities of BIMs, such as real-time data integration and energy simulations, which allow operators to monitor energy usage, detect system faults, and predict when maintenance is required [7–9]. Statistical evidence indicates that implementing BIMs can lead to considerable benefits, including a reduction in energy consumption of up to 35 % in commercial buildings by identifying inefficiencies earlier [10].

Despite the considerable advantages of BIMs, most existing buildings lack these information-rich 3D models, as their design and construction occurred before the widespread adoption of BIM technologies [11,12]. Consequently, project information is often stored and delivered in the form of 2D "as-designed" or "as-built" drawings [13]. These drawings contain valuable semantic information and are readily available, making them an ideal foundation for BIM reconstruction. To address this, numerous studies have proposed methods for reconstructing BIMs from such 2D data (2D-to-BIM), particularly by incorporating artificial intelligence (AI) to enhance reconstruction efficiency [14,15]. Technologies such as optical character recognition (OCR) can automatically extract key features from 2D drawings and efficiently convert them into accurate BIMs [16]. This AI-assisted approach can not only automate the conversion process and reduce human errors but also accelerate the development and application of BIMs. These advancements considerably streamline the otherwise labor-intensive and error-prone manual process of creating BIMs from legacy documents, making it feasible to apply BIMs to a wider range of existing buildings and thus to support better facility management, renovation planning, and operational

* Corresponding author at: Division of Sustainable Energy and Environmental Engineering, Graduate School of Engineering, The University of Osaka, Osaka, 565-0871, Japan.
E-mail addresses: xing.liang@it.see.eng.osaka-u.ac.jp (X. Liang), yabukin@tcu.ac.jp (N. Yabuki), fukuda.tomohiro.see.eng@osaka-u.ac.jp (T. Fukuda).

https://doi.org/10.1016/j.autcon.2025.106584
Received 16 April 2025; Received in revised form 26 September 2025; Accepted 1 October 2025
Available online 18 October 2025

efficiency.

While many prior studies have focused on developing algorithms for 2D-to-BIM reconstruction, a critical challenge remains unresolved: the lack of publicly available datasets containing paired 2D data and reliable reference (RF) BIMs [17]. Specifically, 3D reconstruction is typically evaluated using metrics such as earth mover's distance and intersection-over-union (IoU), both of which require well-defined RF models for quantitative accuracy assessment. However, the objective evaluation and benchmarking of 2D-to-BIM methods, a specialized category of 3D reconstruction, remain challenging due to the scarcity of RF BIM datasets [16].

This issue arises mainly because most existing buildings do not have corresponding 3D models or BIM data, and even when these models exist, confidentiality agreements and intellectual property concerns within the building sector often restrict their sharing and dissemination [18,19]. Consequently, current 2D-to-BIM research is often validated using limited case studies based on researcher-collected data, making cross-method comparisons difficult [20]. Although recent research has introduced metrics that do not require RF data, such as the dense map posterior [21], which evaluates reconstruction quality based on point cloud density, obtaining point clouds remains resource-intensive, involving substantial manual labor, equipment investment, and complex path planning [22]. Given these challenges, there is an urgent need to develop cost-effective methods for creating comprehensive datasets containing BIMs to facilitate robust and objective evaluations of 2D-to-BIM methods.

Motivated by this gap, to address the challenge of BIM data scarcity, this study proposes an automated framework that utilizes deep learning for synthetic BIM dataset generation. The pipeline begins with defining polygonal floor plan boundaries using Python's OpenCV library, followed by employing convolutional neural networks (CNNs) to generate room and wall layouts within these boundaries, thereby producing raster floor plans. Semantic and geometric information is then extracted from these raster images via image processing techniques and combined with predefined semantic attributes and dimensional parameters. These enriched data are then fed into BIM software application programming interfaces (APIs) to automatically generate corresponding BIMs. Through this process, this study has created the Residential unit BIM (ResBIM) dataset, consisting of over 1000 BIMs paired with annotated 2D floor plans, alongside an automatic annotation toolbox for efficient labeling of 2D drawings.

Building on this framework, this research contributes theoretically and methodologically to 2D-to-BIM reconstruction and BIM automation through the establishment of a procedural BIM synthesis paradigm. This paradigm integrates deep learning-based floor plan generation with structured semantic enrichment via explicit mapping between 2D drawing elements and BIM entities, thereby resolving scalability and consistency limitations found in existing BIM dataset construction. ResBIM dataset provides synthetic floor plans with controlled geometric variability and RF BIMs, offering a reproducible resource for objective methodology evaluation. Furthermore, the open-source annotation toolbox formalizes semantic relationships across 2D-to-BIM domains, fostering community-driven refinement of annotation practices and advancing structured knowledge representation in BIM automation research.

The remainder of this paper is organized as follows. Section 2 reviews related datasets and existing AI-driven 2D-to-BIM approaches. Section 3 describes the proposed method for automatic BIM generation. Section 4 presents evaluation experiments and the generation of the ResBIM dataset. Section 5 discusses the research findings and limitations. Finally, Section 6 concludes the paper and outlines future research directions.

2. Related work

2D-to-BIM typically involves two main steps: data extraction and BIM generation. The primary objective of the data extraction phase is to recognize and extract geometric and semantic information from drawings, a process in which AI techniques are frequently employed. To support AI research on 2D-to-BIM tasks, Table 1 summarizes datasets released after 2015 that provide floor plans or BIMs, each with varying levels of accessibility. Specifically, the table provides an overview of each dataset's data source, data format, the presence of 3D data, and its accessibility. In this work, 3D models in the datasets are classified into two categories: ground truth (GT) models and RF models. GT models are derived from real-world buildings measured with high-precision tools and are manually constructed, whereas RF models refer to synthetically or algorithmically generated 3D data.

2.1. *Data extraction and BIM generation*

The data used for 2D-to-BIM reconstruction can be classified based on their data format (vector or raster) and source (real-world or synthetic). This classification yields four distinct types: real-world vector, real-world raster, synthetic vector, and synthetic raster. Each type requires different data extraction methods.

Real-world vector data from CAD software (e.g., DWG, DXF) contain precise geometries and attributes [39,40]. Such data often contain rich implicit information distributed across different layers, including line width, segment spacing, text, and color, which makes rule-based algorithms widely adopted for classification and information extraction. For instance, methods such as automatic layer classification method (ALCM) and ALCM-based elevation detection have been developed to identify hidden layers and infer floor levels [41], while commercial software vendors leverage rule-based algorithms and APIs (e.g., AutoCAD API) to facilitate BIM reconstruction [42]. However, CAD drawings usually contain multiple categories of building elements (e.g., furniture, structural components). Since most existing studies focus on a single category such as architectural floor plans, preprocessing remains necessary—for example, removing furniture and staircases or splitting multiple drawings into independent files [15,43]. Moreover, vector files are not always readily available, especially in renovation projects where digital records are often missing [44].

Real-world raster data (e.g., CFC-FP [23], SydneyHouse [24]) typically consist of scanned drawings or basic digital sketches stored in formats such as PNG. These drawings contain geometric components (e.g., walls, doors) and textual annotations (e.g., dimensions, room names). Data extraction from raster data usually combines CNN-based object detection with OCR techniques [45]; for example, Faster R-CNN with OCR has been used to generate IFC-compliant BIMs [42]. However, many mainstream detection and recognition models were originally trained on natural images and may not perform well on architectural drawings. To address this limitation, recent studies have benchmarked popular detection and recognition models on three different real-world raster floor plan datasets, evaluating their performance in recognizing objects of different sizes (large, medium, small) as well as both handwritten and machine-printed annotations [46]. Similar to real-world vector data, raster drawings also rely on preprocessing to filter out non-target objects; for example, a CNN-based approach has been proposed to detect wall and opening pixels in order to remove furniture and other elements, thereby enabling more accurate reconstruction of vertical, horizontal, and diagonal walls [47].

Both synthetic vector and raster data are algorithmically generated or augmented from real layouts. Although they may not fully capture real-world complexity, they provide standardized geometries and rule-based semantics, which make data extraction more tractable and have therefore been widely adopted for dataset construction and benchmarking [48]. Rule-based and script-driven methods (e.g., Python) are effective for parsing such layouts. For instance, the augmented dataset RPLAN [28] encodes architectural semantics via pixel positions and color values, enabling a coupled generative adversarial network (CoGAN) to learn architectural feature relationships [49].

Table 1
Datasets released after 2015 containing floor plans or building information models (BIMs).*1, *2, *3, *4

Dataset	Year	Data source*1	Format*2	Availability*3	Presence of 3D data*4	Number of data
CVC-FP [23]	2015	RW	R	√	×	122 scanned floor plans
SydneyHouse [24]	2016	RW	R	√	×	174 random houses' floor plans in Sydney
ROBIN [25]	2017	A	R	√	×	510 real-world floor plans
R2V [26]	2017	A	R, V	√	×	100 k + synthetic vector-graphics floor plans
CubiCasa5K [27]	2019	RW	R, V	√	×	5 k floor plans annotated into over 80 floor-plan object categories
RPLAN [28]	2019	A	R, V	○	×	80 k + annotated plans of collected real buildings
HouseExpo [29]	2020	S	R	√	×	35 k + synthetic indoor layouts
Structure3D [30]	2020	A	R	○	RF	3.5 k house designs with 3D wireframe
ZInD [31]	2021	RW	R	○	×	71 k images-derived floor plans
FloorPlanCAD [32]	2022	RW	V	√	×	15 k vector floor plans
Indoor PC/BIM [33]	2023	RW	BIM, PC	√	GT	Point clouds of 5 indoor spaces and corresponding as-built BIMs
MLSTRUCT-FP [34]	2023	A	R	√	×	954 multi-unit floor plans
Tell2Design [35]	2023	A	R	√	×	80 k + residential floor plans for natural language design
SLABIM [36]	2024	RW	BIM, PC	√	GT	1 large BIM model of a university building
MSD [37]	2024	A	R, V	√	×	5.3 k floor plans of building complexes
BIMNet [38]	2025	RW	BIM, PC	○	GT	25 IFC-based BIMs, containing 382 rooms and corresponding point clouds

*1 RW = real world, A = augmented from real-world data but does not directly reflect the real world, S = synthetic.
*2 R = raster, V = vector, PC = point cloud. Multiple formats are available if multiple annotations exist.
*3 Availability: √ = has been published on project website or GitHub, ○ = available on request.
*4 GT = ground-truth model, RF = reference model.

The extracted data serve as input parameters for BIM generation, which typically follows three approaches: IFC-based methods (e.g., IfcOpenShell), parametric design tools (e.g., Dynamo), and BIM software APIs (e.g., Revit API). IFC-based methods provide interoperability but require expertise [50]; parametric tools offer extensible visual workflows but struggle with scalability [17]; APIs allow fine-grained automation but are software-dependent [41].

Overall, the choice of data source directly shapes the complexity and reliability of data extraction in 2D-to-BIM workflows. Real-world inputs are noisy and inconsistent, often requiring preprocessing, intermediate files (e.g., XML, TXT), and manual validation before BIM reconstruction [14,51]. Synthetic data, in contrast, are semantically explicit and scalable, making them ideal for automated pipelines.

2.2. 2D-to-BIM evaluation

Current 2D-to-BIM research typically evaluates two main stages separately: data extraction and BIM generation. For the data extraction stage, the primary focus lies in measuring the accuracy of retrieving semantic and geometric information from 2D drawings, particularly in AI-based approaches. Common evaluation metrics include detection rate, pixel accuracy, and class accuracy, as summarized by Pizarro et al. [20]. Although most studies employ one or more of these metrics, the absence of a standardized evaluation protocol leads to inconsistent metric usage, making cross-comparison between studies difficult, even when the tasks are similar. As a result, existing works remain at the level of fragmented single evaluations rather than standardized benchmarking.

For the BIM generation stage, evaluation typically focuses on BIM fidelity, which requires comparing the generated BIM with a ground-truth model or a RF BIM. Commonly used metrics include geometric IoU from point cloud comparisons or component matching based on IFC outputs [43,52]. However, in the absence of RF models, evaluation often relies on manually created test cases or small proprietary datasets, supplemented by qualitative visual inspection or manual verification [15]. Such datasets are frequently tailored for specific purposes—such as residential building floor plans—introducing potential bias and reducing objectivity [14].

For example, Yang et al. [39] proposed a semi-automatic BIM reconstruction approach based on layer-segmented CAD drawings and rule-based extraction, which was followed by BIM modeling through the Revit Dynamo plugin. Their evaluation involved only two manually created cases and reported execution times of approximately 4500 s and 1800 s, without employing standardized accuracy metrics. Similarly, Zhao et al. [42] introduced a hybrid AI-enhanced approach combining image processing and Faster R-CNN, in which accuracy was judged by comparison with another object detection model (YOLO) rather than a RF BIM.

In addition, most current studies still focus primarily on floor plans, with height and semantic attributes either set to default values or manually specified [15,45], highlighting that 2D-to-BIM remains at an early stage. These methodological limitations are further exacerbated by the scarcity of publicly available paired 2D-3D datasets, constraining objective evaluation [14,40]. Although some public datasets (e.g., Indoor PC/BIM [33], SLABIM [36], BIMNet [38]) provide BIMs paired with scanned point clouds, they lack corresponding 2D drawings and are primarily designed for scan-to-BIM or SLAM applications. Collectively, these challenges highlight a critical research gap: the absence of large-scale paired 2D-BIM datasets, without which the development of standardized evaluation protocols and rigorous benchmarking remains difficult to achieve.

2.3. AI-driven synthetic BIM generation

Data scarcity and the lack of standardized benchmarks have motivated increasing reliance on AI-driven floor plan generation to expand BIM availability. Recent advances in deep learning show considerable potential for creating enriched BIM datasets by producing synthetic floor plans augmented with predefined attributes for 2D-to-3D conversion.

Early approaches typically adopted a two-stage "semantic segmentation plus geometric optimization" pipeline. For instance, the indoor scene synthesis network (ISSNet) [53] leveraged CNNs to produce pixel-level semantic segmentation of room types, with its output subsequently serving as constraints in mixed-integer quadratic programming [54] to perform geometric layout optimization. While this paradigm improved spatial rationality and layout regularity, establishing itself as an important baseline in automatic floor plan generation, the method often suffered from limited global consistency and reduced robustness in handling complex or large-scale scenarios.

To address such limitations, end-to-end frameworks have been proposed, mainly GAN- and CNN-based. GAN-based approaches (e.g., House-GAN++ [55], HouseGanDi [56]) generate symbolic nodes for functional spaces, connected by wall segments to form layouts. In BIM context, Ghannad et al. [49] and Liu et al. [57] applied CoGAN and GC-GAN for modular residential design, exporting outputs into IFC or Revit-based BIMs. Although effective for diversity and conceptual design, GANs often produce irregular geometries, gaps, or mode collapse, undermining their robustness for BIM generation [57–59].

CNN-based methods typically employ cascaded architectures (e.g., combining ResNet-34 [60] and U-Net [61]) to extract geometric and semantic features before generating structured layouts. These pipelines are particularly effective for drawing analysis and rule-constrained tasks owning to their high geometric precision and boundary stability. Wu et al. [28] imposed boundary constraints to produce stable and regular floor plans. This approach aligns with our previous study [62], which introduced a spatial constraint strategy to generate floor plans with greater spatial consistency. The generated plans were exported to structured Excel sheets and further processed for semi-automated BIM conversion. However, similar to earlier works [17,45], the semantic and material diversity remains limited, as generated BIMs continued to rely on default materials and height parameters.

Despite these advances, existing studies have yet to achieve a unified and scalable framework for generating semantically and geometrically valid BIM datasets. A fully automated synthetic BIM dataset pipeline is needed to overcome data scarcity and support standardized evaluation, laying the groundwork for benchmark datasets that enable reproducible cross-method comparison.

3. Methodology

This study presents a deep learning-based framework for automated BIM dataset generation, aimed at supporting the creation of large-scale, semantically rich floor plan–BIM pairs. As illustrated in Fig. 1, the workflow consists of four primary steps: data preparation, data extraction, BIM generation, and annotation. Importantly, each step is designed in modular, allowing for multiple alternative implementations depending on specific research objectives or software environments.

To enable a fully automated workflow, a four-channel intermediate image representation is introduced to separately encode geometric structures, spatial semantics, height information, and material attributes for each floor plan. Specifically, the content of the first and second channels is extracted from the generated floor plan using image processing techniques, while the third and fourth channels contain predefined attributes for height information and material semantics, respectively.

The process begins with the deep learning–based generation of synthetic floor plans, which serve as the foundational layout for BIM modeling. Geometric and semantic data are extracted from the generated floor plans using image processing techniques and stored in the first and second channels of the intermediate file. Since the generated output is inherently two-dimensional, predefined height and material properties are assigned to the third and fourth channels, respectively. This intermediate representation is then used as the input to the Revit API for automated BIM generation. The final output is a multimodal dataset comprising both BIMs and paired annotated drawings. To further enhance the utility of the generated BIMs, corresponding annotation tools are developed for generating 2D drawings in various aspects (e.g., floor plans, elevations), providing essential data to facilitate 2D-to-BIM research.

For demonstration purposes, a modified ResNet-34 [60] and U-Net [61] combination is employed for floor plan generation, trained on RPLAN dataset [28], with subsequent modeling and annotation performed via the Revit API. This architecture is chosen for its strong performance in producing well-structured layouts with stable boundaries, while Revit API-based BIM generation ensures compatibility with industry standards. The chosen configuration achieves a balance between generation quality and practical software integration.

The applicability of the proposed framework is fundamentally determined by the characteristics of the chosen deep learning models and training datasets. In this study, the RPLAN dataset is used to train and validate the deep learning model; consequently, the generated floor plans reflect the structural conventions present in RPLAN—for example, the inclusion of a living room in every floor plan—as detailed in Section 3.1.1. The applicability of the framework can be further extended in future research by employing datasets with broader and more diverse geometric or functional variations.

3.1. Data preparation

Data preparation encompasses the selection of datasets and the configuration of deep learning networks. As outlined in Section 2, recent open-source datasets released since 2015 have been reviewed, with their characteristics analyzed according to data sources and formats. The advantages and limitations of different neural network architectures for floor plan generation have also been discussed. Beyond the dataset–network combination adopted in this framework, alternative models and training datasets can be applied to accommodate varying research objectives or domain-specific requirements.

3.1.1. Training data

The RPLAN dataset [28] serves as the training data in this framework. It is derived from real-world residential building floor plans, with

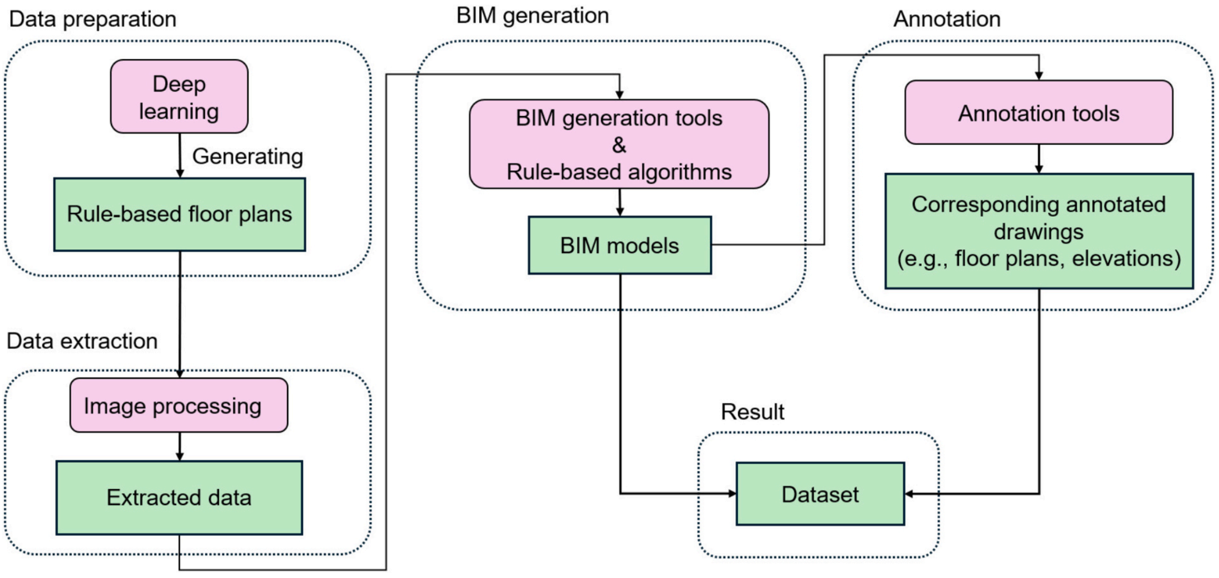


Fig. 1. Overall workflow of proposed framework.

common areas (e.g., elevators, staircases) removed, and each residential unit converted into a rule-based floor plan through algorithmic processing. The dataset contains over 80,000 floor plans, with all data points adhere to the following rules:

- Each data point is in RGBA format (four channels: Red, Green, Blue, and Alpha) in PNG files. The RGB channels represent color information, while the Alpha channel encodes transparency, controlling pixel opacity. In this dataset, pixel values across the four channels are only used for labeling and annotating the images to convey semantic information.
- There must be a living room on the floor plan.
- The total area of each floor plan ranges from 60 m^2 to 120 m^2.
- The number of rooms on a floor plan ranges from three to nine, and the average area of each room is between 10 m^2 and 20 m^2.
- The ratio of the living room's area to the whole area is between 0.25 and 0.55.
- Each image datum is 256 $\times$ 256 pixels in resolution, representing a square region of 18 m $\times$ 18 m.

Table 2 describes the four channels and the semantic information encoded in their pixel values. This dataset enables CNNs to learn the internal spatial relationships and the building boundaries, thereby generating floor plans that are spatially comparable to those designed by human architects.

Fig. 2 illustrates the unprocessed raw view and four channels of an RPLAN data point. In Fig. 2(a), the unprocessed raw view represents the combined RGBA channels. Fig. 2(b) shows the first channel, which depicts floor plan boundaries with the front door highlighted in yellow. Fig. 2(c) presents the second channel, where distinct pixel values represent various room types and elements, each assigned unique classifications. Fig. 2(d) corresponds to the third channel, used to calculate the total number of rooms in the floor plan. Fig. 2(e) displays the fourth channel, identifying and representing the enclosed areas within the defined boundaries.

3.1.2. Deep learning networks

The framework adopts the combination of two complementary CNNs: a modified ResNet-34 and a U-Net. The modified ResNet-34 is designed for predicting the centroid locations of rooms from rasterized floor plan images. To improve spatial feature representation, the standard ResNet-34 is modified to accept four-channel (RGBA) input images of 256 $\times$ 256 pixels, with the additional channel encoding supplementary semantic or structural information beyond standard RGB. The U-Net encoder–decoder is employed to predict pixel-level wall structures between the room centroids identified by the modified ResNet-34. This division allows the framework to address two distinct yet interrelated prediction tasks required for automated floor plan generation, as illustrated in Fig. 3.

3.1.2.1. Room prediction strategy. The room prediction follows a "living-room-first" strategy in which the position of the living room is predicted first. This strategy is informed by three primary considerations derived from the characteristics of the training data: (1) the living room is an essential element of residential units, (2) it is typically situated centrally within the floor plan, and (3) it connects to most other rooms either directly or indirectly. Prioritizing the living room's location improves the accuracy and structural consistency of the generated floor plans.

3.1.2.2. Iterative room prediction with modified ResNet-34. The task of room placement is formulated as an iterative localization process, where the model sequentially predicts the centroid locations of all rooms in the floor plan, as shown from step 1 to step 3 in Fig. 3. The process begins by predicting the location of the living room. At each iteration, the modified ResNet-34 receives a multi-channel (RGBA) floor plan image, updated to reflect the placement of previously predicted rooms. The network processes this contextual input to regress the centroid coordinates and semantic label of the next room to be added. After each prediction, the layout is updated by marking the new room's position and type, and the process continues until a predefined stop criterion is reached. The stop criterion is determined by training data and hyperparameters, as discussed in Section 5.2.

3.1.2.3. Wall prediction with U-Net. The U-Net predicts pixel-level wall structures from a multi-channel image that encodes both the spatial boundaries and the centroid of all rooms. The centroid points serve as spatial anchors that guide the network in inferring the logical partitioning of space. As a result, the output pixels represent walls that spatially separate the areas around each predicted room centroid, producing pixel-accurate wall boundaries that are consistent with the overall room layout, as demonstrated from step 3 to step 4 in Fig. 3.

Table 2
Mapping labels and pixel values in training dataset (RPLAN).

Channel	Pixel value	Semantic information	Description
Channel 1	0	Other	Labeling the boundary of the floor plan
	127	Exterior wall	
	255	Front door	
Channel 2	0	Living room	Labeling the different spaces in the floor plan
	1	Master room	
	2	Kitchen	
	3	Bathroom	
	4	Dining room	
	5	Child room	
	6	Study room	
	7	Second room	
	8	Guest room	
	9	Balcony	
	10	Entrance	
	11	Storage	
	12	Wall-in	
	13	External area	
	14	Exterior wall	
	15	Front door	
	16	Interior wall	
	17	Interior door	
Channel 3	0	Non-room area	Distinguishing different rooms starting from 1; and 0 represents non-room area
	1 to 9	Different rooms	
Channel 4	0	Exterior area	Labeling the areas within the floor plan
	255	Interior area	

Note: the above numbers indicate the corresponding pixel values.

3.2. Image processing for data extraction

Image processing techniques are employed to classify and extract geometric structures and spatial semantics from the floor plans generated by the CNN-based models (input). The extracted results are then encoded into the first and second channels of the intermediate representation. The input data are in raster format and are converted into a three-dimensional matrix (Input) using image processing libraries (e.g., Python's OpenCV), as shown below:

$$Input \in \mathbb{R}^{h\times w\times c}, Intermediate = 0 \in \mathbb{R}^{h\times w\times c}, Mask = 0 \in \mathbb{R}^{h\times w\times c}, \quad (1)$$

where h is the height resolution of the input 2D image, w is the width resolution, and c is the number of channels. Additionally, two matrices—*Intermediate* and *Mask* are initialized as zero matrices of the same dimension as *Input*, where *Intermediate* representing the PNG output in matrix form and *Mask* serving to identify different pixel classes within the image matrix.

Different pixel values in the *Input* matrix represent distinct building-related elements, with specific values determined by the CNNs and training dataset. Each number in the *Input* matrix is evaluated to

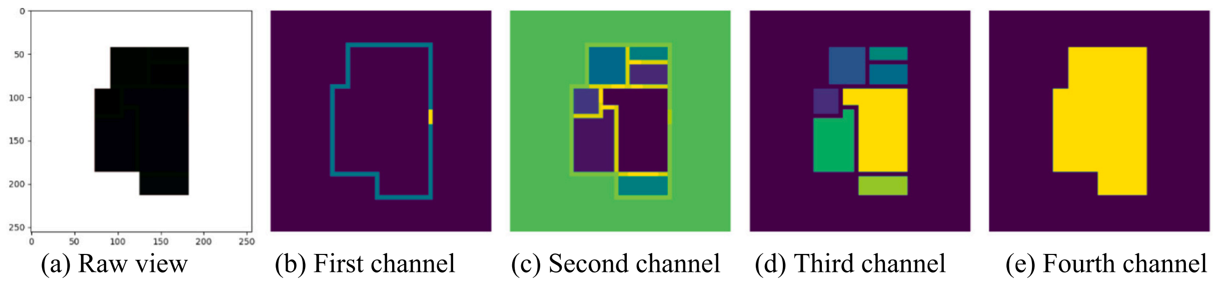


Fig. 2. Different views of sample from RPLAN dataset. (a) superimposed RGBA channels; (b) extracted floor plan boundaries; (c) room segmentation represented by distinct pixel values; (d) mask used for room count calculation; (e) area computation based on enclosed boundaries.

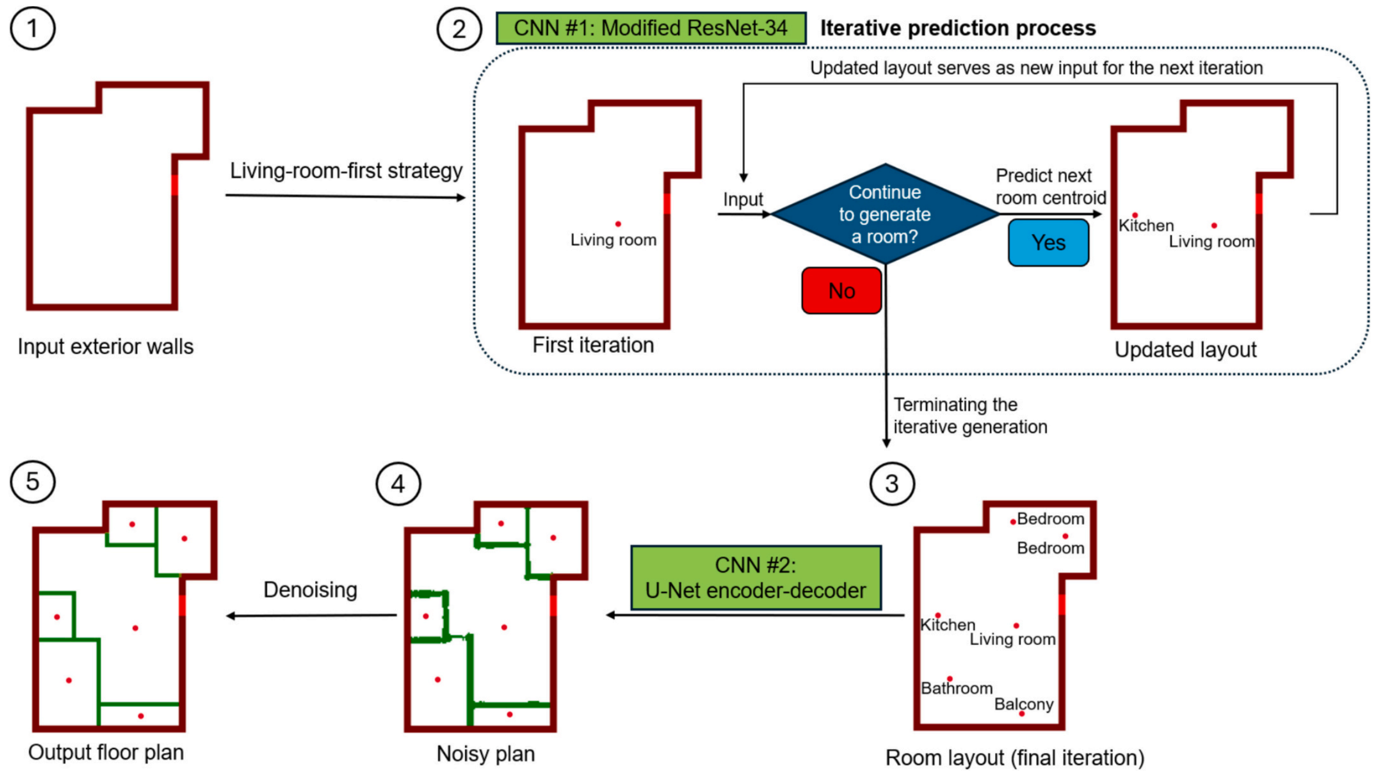

1
2
CNN #1: Modified ResNet-34
Iterative prediction process
Updated layout serves as new input for the next iteration
Living-room-first strategy
Input exterior walls
Living room
First iteration
Input
Continue to generate a room?
Predict next room centroid
Yes
No
Kitchen
Living room
Updated layout
Terminating the iterative generation
5
4
3
Denoising
CNN #2: U-Net encoder-decoder
Bedroom
Bedroom
Kitchen
Living room
Bathroom
Balcony
Output floor plan
Noisy plan
Room layout (final iteration)

Fig. 3. Workflow of combined networks.

determine whether it represents a geometric building component or spatial semantic information. Pixel values identified as geometric elements are assigned to the *Geometry* array, whereas values corresponding to spatial semantics are assigned to the *Spatial* array. The resulting arrays are formalized as follows:

$$Geometry = \{g_1, g_2, \cdots, g_n\}, Spatial = \{s_1, s_2, \cdots, s_m\} \tag{2}$$

where g represents the pixel values classified as geometry, n represents the total number of types of geometry elements, s represents the pixel values classified as spatial semantics, and m represents the total number of types of spatial semantics. In this study, the pixel values for g_n and s_m are provided in Table 2.

Next, the zero-initialized matrix *Mask* at position (h, w, c) is updated based on the classification results from the *Input* matrix. Specifically, if the value at $Input(h, w, c)$ is found in the *Geometry* array, then the corresponding position in *Mask* is set to 1; if it belongs to the *Spatial* array, then it is assigned a value of 2. This process is formalized as follows:

$$Mask(h, w, c) = \begin{cases} 1, & if\ Input(h, w, c) \in Geometry, \\ 2, & if\ Input(h, w, c) \in Spatial, \\ 0, & otherwise. \end{cases} \tag{3}$$

Finally, based on the values in *Mask*, the zero matrix *Intermediate* is updated as follows:

$$Intermediate(h, w, k) = \{Input(h, w, c)\, |Mask(h, w, c) = k\}, k = 1, 2. \tag{4}$$

When $Mask(h, w, c) = 1$, the value at $Input(h, w, c)$ is assigned to the first channel of *Intermediate* $(k = 1)$, representing *Geometry*. Similarly, values belonging to the *Spatial* array are assigned to the second channel $(k = 2)$.

Through Eqs. (1)–(4), all geometric and spatial semantic data extracted from the CNN-generated floor plans are transformed into the first and second channels of the *Intermediate* matrix. An example of these channels is shown in Fig. 4, illustrating the geometric map and spatial semantics, with pixel values derived from the CNN-generated floor plans.

3.3. BIM generation

The BIM generation process takes as input the mathematical parameter set produced in step two (image processing), which describes the building layout in terms of geometric coordinates, semantic labels, height values, and material attributes. These parameters can be stored in

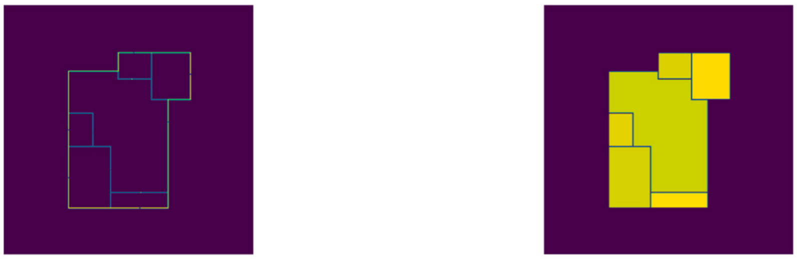

(a) First channel of intermediate PNG (b) Second channel of intermediate PNG

Fig. 4. First and second channels of intermediate PNG, containing geometries and spatial semantics derived from floor plans generated by convolutional neural networks (CNNs).

any structured format (e.g., JSON, CSV/Excel tables, raster channel encodings) and are independent of any specific BIM platform. The parameters are then mapped to building elements through either platform-neutral toolchains, such as generating IFC entities via Python with IfcOpenShell, or through platform-specific APIs (e.g., Autodesk Revit API, ArchiCAD API) for direct model creation.

To demonstrate this concept, a set of rule-based algorithms was developed and utilized alongside Revit API in Revit 2024 [63], implemented using C# in Visual Studio 2019. These algorithms read the intermediate four-channel image as input, decode each channel into its corresponding geometric and semantic attributes, and map them to Revit building components. To achieve full automation, the algorithms and the Revit API procedures were integrated into a single executable script, which was compiled into a dynamic link library (DLL). This DLL is loaded directly within Revit 2024, enabling the BIM generation process to run entirely within the native Revit environment without manual intervention.

3.3.1. *Decision-making strategy for semantic completion*

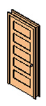
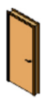
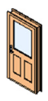
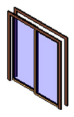
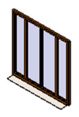
Since the generated floor plans are inherently 2D, supplementary information—such as height values and semantic attributes (e.g., window offsets, wall materials)—must be added before conversion. These attributes are encoded into the third (height) and fourth (material semantics) channels of the intermediate representation.

To overcome the limitations of previous studies in representing semantic and dimensional diversity, a custom component library was developed, with representative entries listed in Tables 3 and 4. The semantic completion strategy is governed by the following global heuristic design rules:

- The first floor of the BIM—referred to as "Level 0"—is set at zero height.
- Elements belonging to the same category within the same floor plan have uniform semantic attributes.
- Interior and exterior walls within the same floor plan are assigned uniform height values.

As an alternative, the material semantics and dimensions presented in Tables 3 and 4 may be expanded or replaced to suit different research objectives by the workflow demonstrated in Fig. 5. For example, when adding a new interior door, the process begins by searching the local library. If the desired instance is unavailable, then the official Revit libraries [64]—offering a comprehensive selection of materials and components for various countries—can be consulted; if the instance remains unavailable, then resources from the Revit community may be utilized, or the instance may be created manually as a last resort. Once the target instance is acquired, its dimensions are verified: if the required dimensions exist, then the instance can be added to the local library for API access; otherwise, a similar existing instance may be modified and saved as a new variant.

Table 3
Semantics and dimensions of windows and doors.

Category	Exterior door	Interior door	Sliding door	Plain window	French window
Legend		 			
Length × height (mm × mm)	810 × 2110 910 × 2110 1010 × 2110	810 × 2010 910 × 2010 810 × 2110 910 × 2110	1700 × 2000 1700 × 2100 1800 × 2000 1800 × 2100	910 × 910 910 × 1210 1360 × 910 1360 × 1210 1810 × 910 1810 × 1210	2100 × 2150
Offset (m)	0	0	0	0.8–1	0

3.3.2. *Height and semantic completion*

The next step is to assign height information to *Door*, *Wall*, and *Window* in the third channel. Eq. (5) determines the type of architectural element represented by each pixel in the first channel by querying its pixel value. Different values are assigned in the third channel representing the corresponding height in the first channel, based on the component library and global heuristic design rules specified in Section 3.3.1:

$$Intermediate(h,w,3)=\begin{cases}0, & Intermediate(h,w,1)\in Door,\\ U(28,30), & Intermediate(h,w,1)\in Wall,\\ 0\cup U(8,10), & Intermediate(h,w,1)\in Window.\end{cases} \tag{5}$$

Because the standard range of pixel values in PNG images is 0 to 255, this study represents height by multiplying the pixel value by 100 mm; for example, a wall with a height of 2950 mm would have a pixel value of 29.5 in the third channel. $U(28,30)$ is a random number between 28 and 30, representing 2.8–3 m in height. The geometry in the third channel is identical to that in the first channel in Fig. 4(a), but the pixel values differ, representing the height of the element.

Revit API can be divided broadly into three main functional modules: geometric modeling, semantic modeling, and filter and query module. The geometric modeling module is responsible for creating the basic geometric shapes of building elements, such as points, lines, and curves. Building on this foundation, the semantic modeling module assigns attributes, properties, and relationships to these geometric shapes,

Table 4
Semantics and dimensions of walls.*1

Category	Legend/Semantic information*1	Width (mm)	Offset (mm)	Height (m)
Exterior wall	Wall-Ext_102Bwk-50Air-45Ins-100DBlk-12P	310		
	Wall-Ext_215Bwk	215	0	2.8–3
	Wall-Ext_102Bwk-75Ins-100LBlk-12P	290		
Interior wall	Wall-Int_12P-100Blk-12P	125		
	Wall-Partn_12P-75Std-12P	100		
	Wall-Partn_30Gwb-70MStd-30Gwb	130		

*1 Ext = Exterior, Int = Interior, Partn = Partition, Bwk = Brickwork, Air = Air cavity, Ins = Insulation, DBlk = Dense blockwork, Blk = Blockwork, P=Plaster, Std = Stud, Gwb = Gypsum wallboard, MStd = Metal stud.

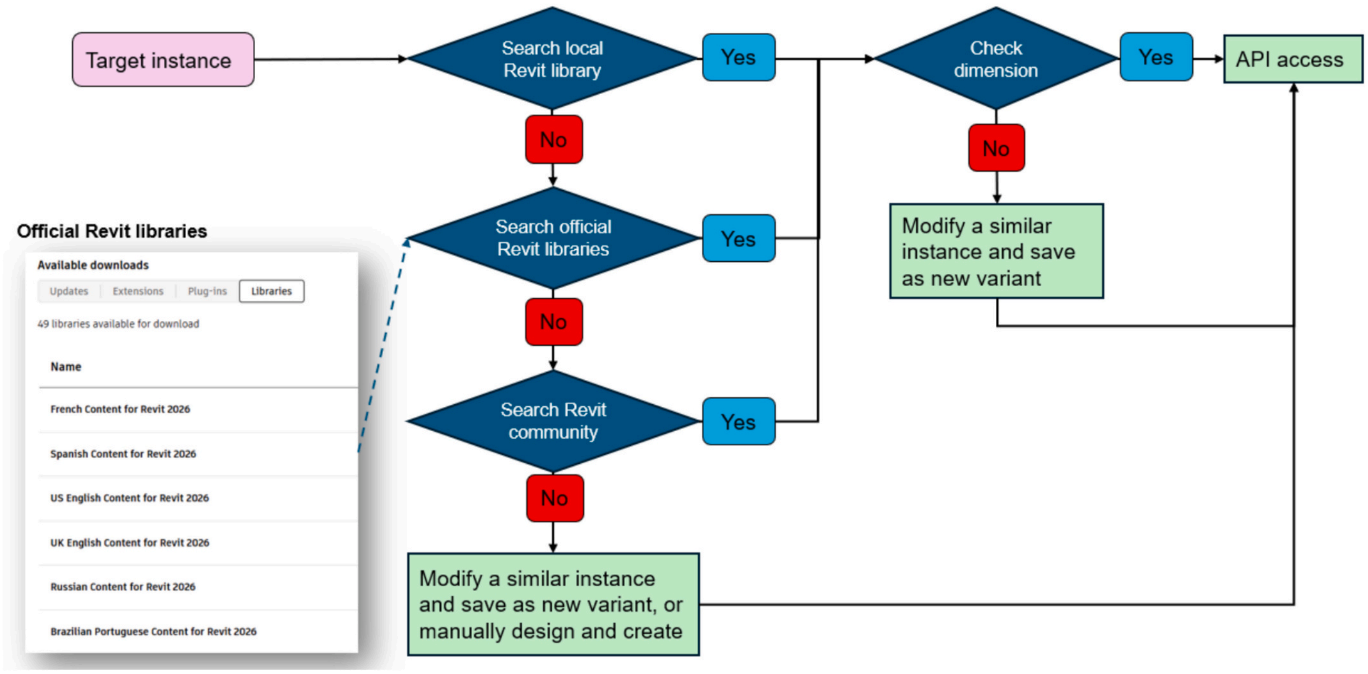

Target instance
Search local Revit library
Yes
No
Check dimension
Yes
API access
No
Modify a similar instance and save as new variant
Official Revit libraries
Available downloads
Updates
Extensions
Plug-ins
Libraries
49 libraries available for download
Name
French Content for Revit 2026
Spanish Content for Revit 2026
US English Content for Revit 2026
UK English Content for Revit 2026
Russian Content for Revit 2026
Brazilian Portuguese Content for Revit 2026
Search official Revit libraries
Yes
No
Search Revit community
Yes
No
Modify a similar instance and save as new variant, or manually design and create

Fig. 5. Workflow for material extension or replacement.

effectively transforming raw geometric models into semantically rich building components. Notably, the assignment of semantic attributes must be performed by selecting components from Revit's library via the filter and query module. Related operations of the component library are discussed in Section 3.3.1.

Fig. 6 illustrates an example of creating a plain window instance listed in Table 3. First, three combined filter conditions (*FamilySymbol*, *BuiltInCategory*, and *OST_Windows*) are applied in the filter and query module (*FilteredElementCollector*) to obtain *windowType*, an array containing all window instances that meet these filter conditions.

The second step uses *NewFamilyInstance* method to create a new instance in Revit by specifying both the element type and a designated coordinate. In this example, the fourth element ($windowType[3]$ = Windows_Sgl_Plain 1360 × 1210 mm) from the query result and a coordinate (*point*) are specified. This creates a 1360 mm × 1210 mm window instance at the designated location *point* in Revit.

The specific material properties of the generated window are built-in in Revit, as illustrated in Fig. 7. Therefore, the pixel value of the fourth channel (material map) of the intermediate file is assigned based on

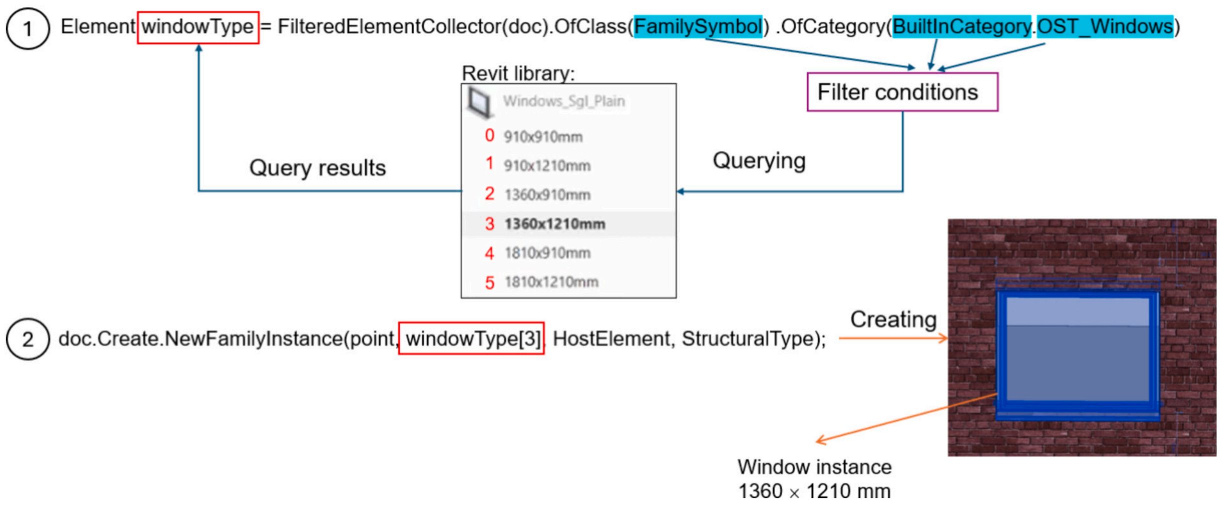
$$Intermediate(h,w,4) = \{s|Intermediate(h,w,1) \in Geometry\}, s \in \{0,1,\cdots,n-1\}, \tag{6}$$

where n is an integer representing the total number of specific building elements in the current Revit library that meet all filtering conditions, and s is the assigned pixel value, which is a random integer between 0 and $n-1$.

3.3.3. From floor plans to BIMs

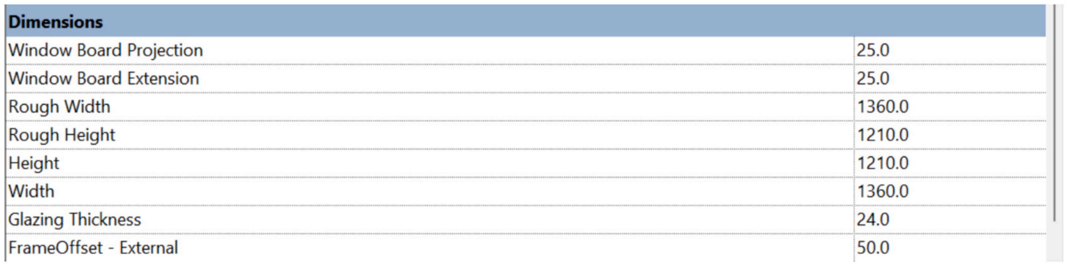
The placement of each BIM element is determined by the geometric coordinates derived from the first channel of the intermediate matrix. However, in the pixel matrix representing the rasterized image, each pixel's coordinates only indicate its position within the matrix without any real-world distance association. To address this, a scaling factor is introduced to convert pixel coordinates into actual distances.

In this study, the training dataset used was RPLAN, which consists of a large collection of real-world floor plans normalized by the dataset's authors to a resolution of 256 × 256, representing an actual area of 18 m × 18 m. This normalization implies that one pixel corresponds to



```
Element windowType = FilteredElementCollector(doc).OfClass(FamilySymbol) .OfCategory(BuiltInCategory.OST_Windows)
doc.Create.NewFamilyInstance(point, windowType[3], HostElement, StructuralType);
```

Fig. 6. Generating window instance by Revit API.



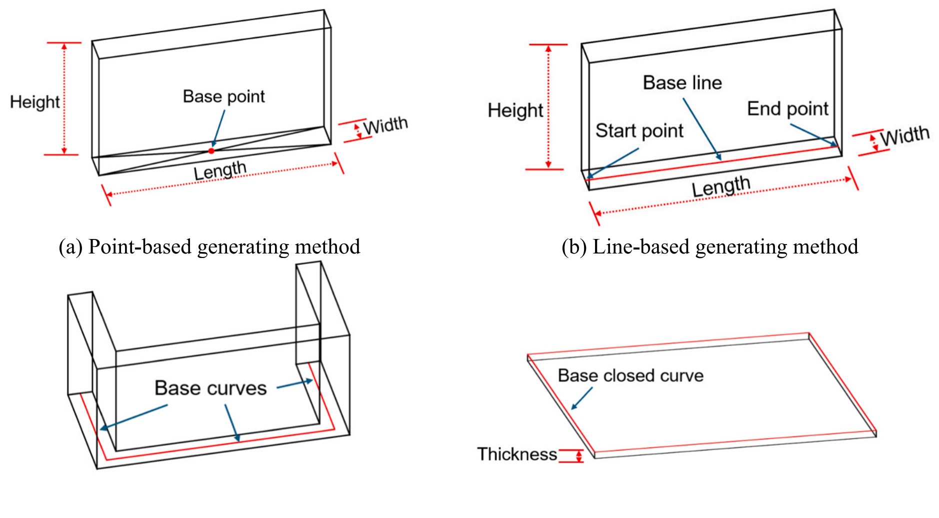
Dimensions	
Window Board Projection	25.0
Window Board Extension	25.0
Rough Width	1360.0
Rough Height	1210.0
Height	1210.0
Width	1360.0
Glazing Thickness	24.0
FrameOffset - External	50.0

Fig. 7. Built-in material properties in Revit.

approximately 70 mm in real-world distance. Because the neural network trained on this dataset generates floor plans adhering to this scale, the generated plans approximately follow the same proportion. To ensure compatibility with the Revit software, which uses millimeters as its default unit of length but its API operates with imperial units (foot/ft), a conversion factor $f = \frac{70}{304.8}$ is applied. This conversion ensures that one pixel in the intermediate file accurately represents 70 mm in Revit, thus maintaining accurate scaling and compatibility between the rasterized image output and the BIM generation process.

The orientation of each element is determined by the *FacingOrientation* parameter, defined as a 3D unit vector and can be modified. For example, when a door opens to the north, its *FacingOrientation* $= (0, 1, 0)$; if this parameter is changed to $(0, -1, 0)$, then the door's opening direction is flipped to face south. In this study, directional orientation is computed by computing the spatial relationship between the center point of the living room and directional elements such as doors; directions that are closer to the center point are defined as "inside," whereas those farther away are defined as "outside."

The geometric coordinates in the pixel coordinate system determine



(a) Point-based generating method (b) Line-based generating method

(c) Open curve-based generating method (d) Closed curve-based generating method

Fig. 8. Four geometry generation methods and determined geometry types.

the corresponding position of an element in Revit, while other properties (such as thickness and width) are defined by the material properties. In the first channel of the intermediate file, pixels with the same value represent the same type of building element. From a geometric perspective, these equal-value pixels can be categorized into four types: a single point, a line segment, a non-closed polyline, and a closed loop. These four types correspond to four Revit API classes for BIM element creation: point-based, line-based, open curve-based, and closed curve-based, respectively, as described below:

- Point-based elements such as windows and doors are generated using input 3D point coordinates as the central coordinates at the base of the geometry, as illustrated in Fig. 8(a). The length, width, and height of these elements are determined by the selected material properties.
- Line-based elements such as interior walls are generated using the input line as the central axis along the length of the geometry's base, as shown in Fig. 8(b). The line is defined by a vector from the 3D start point to the 3D end point. The height and length of the generated element are specified by input parameters, while the width is determined by the selected material properties.
- In Revit API, the CurveLoop class is a collection of one or more lines, curves, or arcs. For elements generated from open curves, each individual component (line, curve, or arc) within CurveLoop is treated as the central axis along the length of the geometry's base, following the same generation method as line-based elements, as depicted in Fig. 8(c).
- For elements generated from closed curves, the input closed curve is extruded upward or downward to create geometry, as illustrated in Fig. 8(d). The thickness of the generated element is determined by the selected material properties.

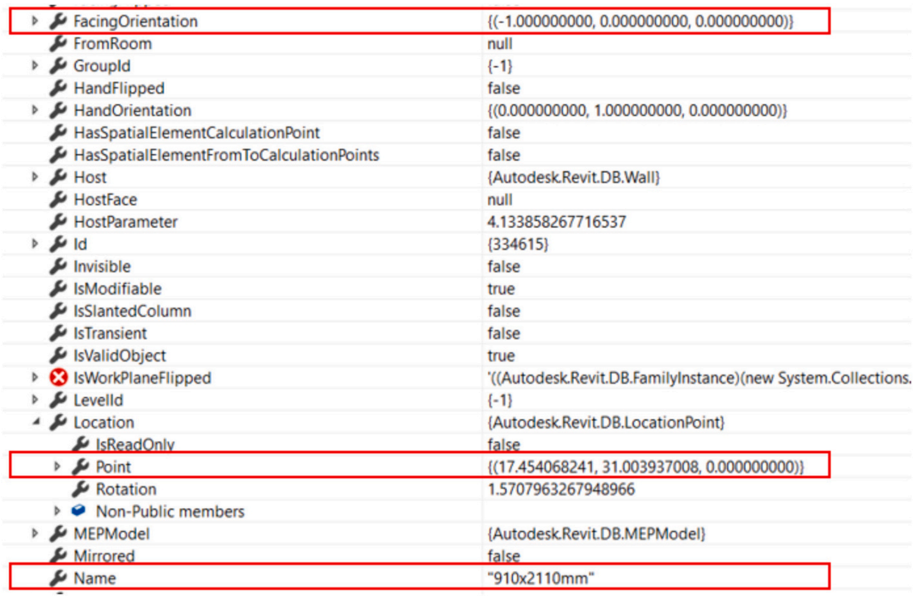
Algorithm 1 is a pseudocode that uses the intermediate PNG as an input parameter and combines the rule-based algorithm and Revit API to generate a BIM. Specifically, Algorithm 1 takes two input parameters: an $h \times w \times 4$ matrix converted from the intermediate image and a dictionary D. The dictionary D is a mapping table with two parameters: a *key* and a *value*: *key* is a pixel value, and *value* is the corresponding element type of this pixel. For example, one data point in D is $\left\{key = 127, value = \text{"ExtWall"}\right\}$, meaning that if the input number 127 is passed to the dictionary, then it returns the string "ExtWall," indicating that this pixel value represents an exterior wall. This dictionary is created manually and converted from the mapping relations presented in Tables 2, 3, and 4. Such a dictionary ensures the scalability of the proposed framework in the case of different datasets used or the addition of new element types by adding a key–value pair to the dictionary.

The algorithm begins by iterating through all the pixel values in the first channel of the matrix. If the value at $Intermediate[x, y, 0]$ is not zero, then this indicates the presence of an architectural element at position $[x, y]$. The corresponding height and material attributes of this element are stored as $Intermediate[x, y, 3]$ and $Intermediate[x, y, 4]$, respectively, while the specific element type is identified using the predefined dictionary D. Next, a 3×3 *mask* is applied around the position $[x, y, 0]$ to search for non-zero-pixel values. This local neighborhood is used to determine whether the pixel belongs to a point, line, open curve, or closed curve geometric classification, as described in Fig. 8.

This process collects all the necessary parameters required by Revit API to generate a complete architectural instance. These include the geometric classification, corresponding geometric coordinates, height information, and material semantic data.

Algorithm 1. BIM generation using Revit API.

```
Input: Intermediate Image $I$ (size $h \times w \times 4$), Dictionary $D$ (key: pixel value, value:
       architectural element type)
Output: Generated BIM Model using Revit API
foreach $(x, y)$ in $I$ do
    $pixel\_value \leftarrow I[x, y, 0]$ // Access the first channel of pixel
    if $pixel\_value \neq 0$ then
        $element\_type \leftarrow D[pixel\_value]$ // Lookup element type from dictionary
        $mask \leftarrow I[x-1 : x+1, y-1 : y+1, 0]$ // Extract 3x3 neighborhood mask
        $neighbors \leftarrow Count(mask == pixel\_value)$ // Count same value pixels
        if $neighbors == 0$ then
            $geometry\_type \leftarrow Point$ ;
        else
            if $neighbors == 2$ then
                $geometry\_type \leftarrow Line$ ;
            end
            else if $neighbors > 2$ and $FormsClosedLoop(mask, pixel\_value)$ then
                $geometry\_type \leftarrow ClosedCurve$ ;
            end
            else
                $geometry\_type \leftarrow OpenCurve$ ;
            end
        end
        $geometry \leftarrow I[x, y, 0]$ // Geometry based on geometry type
        $height \leftarrow I[x, y, 2]$ // Extract height information from channel 3
        $semantic \leftarrow I[x, y, 3]$ // Extract semantic information from channel 4
        Call RevitAPI.CreateElement($geometry, height, semantic$);
    end
end
```



```
FacingOrientation                           {(-1.000000000, 0.000000000, 0.000000000)}
FromRoom                                    null
GroupId                                     {-1}
HandFlipped                                 false
HandOrientation                             {(0.000000000, 1.000000000, 0.000000000)}
HasSpatialElementCalculationPoint           false
HasSpatialElementFromToCalculationPoints    false
Host                                        {Autodesk.Revit.DB.Wall}
HostFace                                    null
HostParameter                               4.133858267716537
Id                                          {334615}
Invisible                                   false
IsModifiable                                true
IsSlantedColumn                             false
IsTransient                                 false
IsValidObject                               true
IsWorkPlaneFlipped                          '((Autodesk.Revit.DB.FamilyInstance)(new System.Collections.
LevelId                                     {-1}
Location                                    {Autodesk.Revit.DB.LocationPoint}
  IsReadOnly                                false
  Point                                     {(17.454068241, 31.003937008, 0.000000000)}
  Rotation                                  1.5707963267948966
  Non-Public members
MEPModel                                    {Autodesk.Revit.DB.MEPModel}
Mirrored                                    false
Name                                        "910x2110mm"
```

Fig. 9. Query results for a door displayed in Visual Studio 2019.

3.4. Automatic annotation tools

The filter and query module of the BIM software API—in combination with custom scripts—enable the extraction of orientation and dimensions for individual building elements. These extracted data are then used to generate dimension annotations in floor plans and elevations.

Various properties of each element within the BIM can be obtained via the filter and query module of Revit API. As shown in Fig. 9, filter and query are used to retrieve the material properties of a door, where the three highlighted data items are *FacingOrientation* (the direction in which the door opens), *Point* (the generation coordinate), and *Name* (the dimensions or identifier). Algorithm 2 is a pseudocode that shows how to use different combination filter conditions to consult the document (BIM model) in the current Revit software and generate annotations for different aspects.

The algorithm begins by using filter and query module to retrieve material properties for a specific type of building element in the current BIM. From the querying results, key material properties are extracted, such as *FacingOrientation*, *Point*, and *Name*. Based on these, a dimension *line* can be created that is consistent with the length and orientation of the element to be annotated.

Next, an offset is specified manually to generate a reference line, *RA*; this is a parallel line that matches the length of *line* but is separated by the specified offset distance. Finally, *RA* is annotated with text representing the length of *line* in millimeters. This approach ensures precise annotations that are aligned with the geometric and material properties of the building elements.

Algorithm 2. Generating annotations.

```
Input: Revit Document doc
Output: Generated annotations on selected BIM elements
Initialize elementFilter ← CreateFilter(MaterialType=TargetMaterial);
elements ← doc.Query(elementFilter) // Elements matching the filter
foreach element in elements do
    materialInfo ← element.Material;
    FacingOrientation ← element.FacingOrientation // Orientation vector
    Point ← element.GetLocationPoint();
    Name ← element.Name // Retrieve name or identifier
    DimensionLine ← CreateLine(point1, point2);
    OffsetVector ← facingOrientation.Normalize() × offset;
    ReferenceArray ← CreateParallelLine(line, offsetVector);
    Annotations ← doc.Create.NewDimension(line, RA);
end
```

3.5. Generality and cross-platform adaptability of the framework

The proposed framework is designed to be platform-independent and modular, comprising four key steps: data preparation, data extraction, BIM generation, and annotation. While the current implementation leverages a specific combination—namely a modified ResNet-34 and U-Net for floor plan generation, and the Autodesk Revit API for BIM modeling and 2D annotation—each step in the workflow supports interchangeable alternatives depending on the dataset, modeling platform, or research objective.

Data preparation relies on data-driven models and is therefore independent of any BIM tool. Data extraction applies image processing methods to convert the generated floor plan into a structured, machine-readable mathematical description of the building layout, consisting of geometric, semantic, height, and material parameters. These parameters can be stored in structured formats—such as raster images, JSON files, or spreadsheets—and serve as the direct input for BIM generation.

Regarding BIM generation, the parameter set can be processed via Python using IfcOpenShell to produce IFC entities or mapped to platform-specific APIs such as the Autodesk Revit API or the ArchiCAD API for model creation. Annotation is based on mathematical view projections of the BIM; using the same platform as in BIM generation helps to avoid format-conversion overhead and semantic mismatch.

4. Experiments

The experiments reported in this study were designed to evaluate the proposed framework's performance in automating the generation of BIMs and paired annotated 2D drawings from generated floor plans. Specifically, the experiments aimed to validate three key aspects: (1) the prediction accuracy of the CNN models, (2) the quality and efficiency of BIM generation using Revit API, and (3) the effectiveness of the automated annotation tool in producing annotated 2D drawings.

4.1. Implementation details

The networks were trained using PyTorch on an NVIDIA GeForce RTX 3090 (24G) GPU. The RPLAN dataset, comprising over 80,000 raster images in PNG format, represents residential unit floor plans, and these images were split into training, validation, and test sets in a 70:15:15 ratio. To improve the training efficiency, the images were converted from PNG to Pickle files, a Python-specific serialized file format that efficiently stores and quickly loads complex data structures. This conversion considerably reduced I/O overhead and enhanced training speed compared to directly loading PNG images. Each Pickle file included masks for room boundaries, categories, indexes, and interior walls or doors, as well as centroid coordinates for individual room types calculated using a custom Python script. Training and testing the two networks took approximately three days.

The first CNN employed was a modified ResNet-34 architecture, adjusted to handle four-channel RGBA images of 256 × 256 pixels. The training settings—including a batch size of 16, an initial learning rate of 0.0001, and a weight decay of 0.0001—followed standard practices from previous CNN-based architectural image analysis studies [58]. Learning rate decay was applied at epochs 30, 60, and 90, based on empirical findings suggesting improved convergence and reduced risk of overfitting. This network was trained for 300 epochs because of the complexity and variability inherent in predicting accurate room centroid locations from raster images, allowing sufficiently many epochs for stable convergence.

The second network utilized an encoder–decoder architecture inspired by U-Net to predict pixel-level wall structures from room centroid locations generated by the first CNN. Similar to the first network, this encoder–decoder architecture used a batch size of 16, an initial learning rate of 0.0001, and weight decay of 0.0001. However, this network was trained for only 100 epochs because pixel-wise segmentation tasks typically converge more quickly because of the direct pixel-level supervision and the structured nature of the wall predictions [61]. The learning rate was progressively adjusted after each epoch to facilitate faster convergence and better model stability.

Both networks were trained independently, and their outputs were integrated sequentially within the proposed BIM generation pipeline, substantially enhancing the automation and accuracy of BIM conversion process.

4.2. Data preparation

Because the output of CNNs is inherently dependent on the training data, the floor plans generated in this study exhibit characteristics

Table 5
Mapping labels and pixel values in CNN-generated floor plans.

Channel	Semantic information	Pixel value
1 (exterior wall)	Exterior wall	127
	Front door	255
2 (interior wall)	Interior wall	127
3 (spatial semantic)	Living room	100
	Master room	101
	Kitchen	102
	Bathroom	103
	Dining room	104
	Child room	105
	Study room	106
	Second room	107
	Guest room	108
	Balcony	109
	Entrance	110
	Storage	111
	Wall-in	112
4 (inside area)	Inside area	255
	Outside area	0

Note: the above numbers indicate the corresponding pixel values.

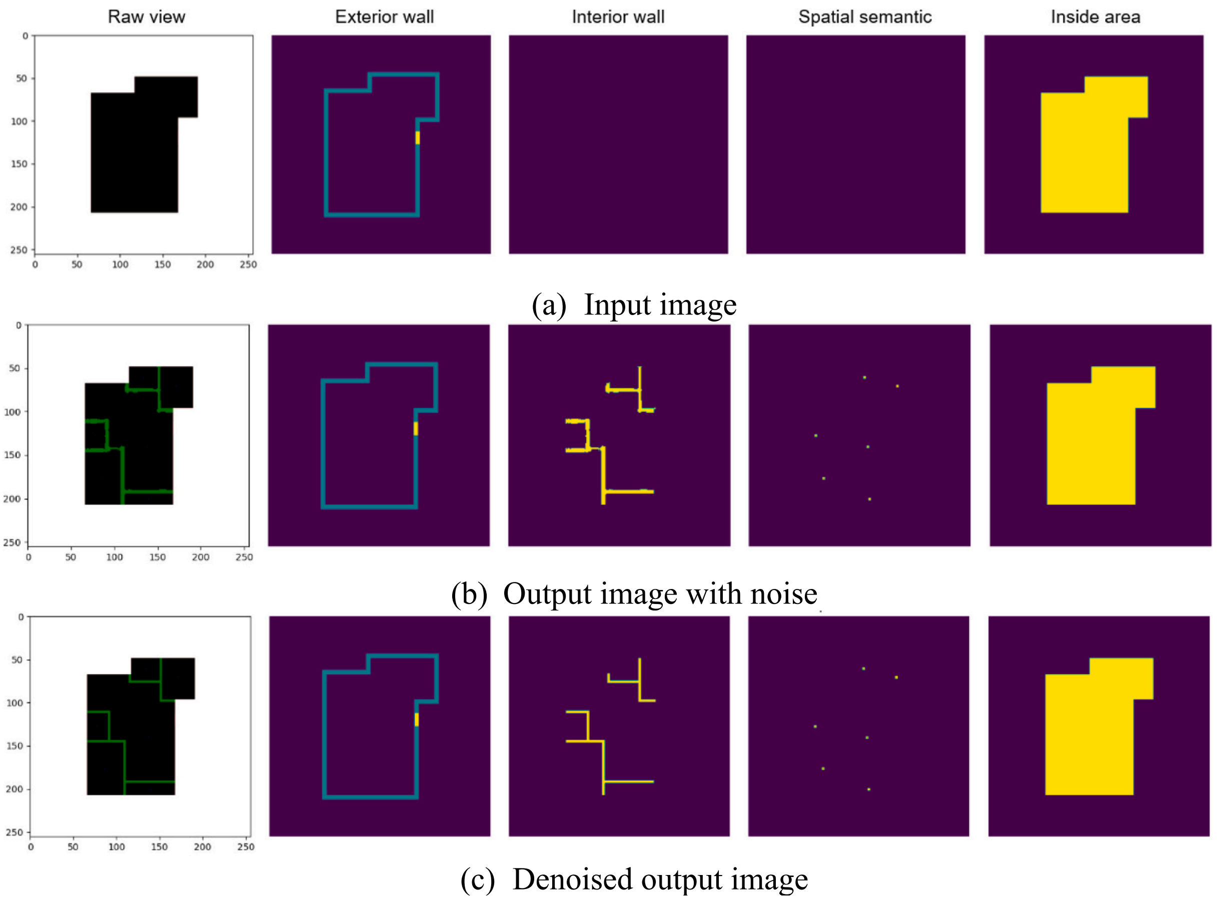


Fig. 10. Image input and output of networks.

similar to those of the RPLAN training dataset. Specifically, RPLAN represents an 18 m × 18 m spatial area using a 256 × 256-pixel map, where each pixel corresponds to approximately 70 mm × 70 mm (about 4.9×10^{-3} m^2). Each floor plan in RPLAN spans 60–120 m^2, equivalent to approximately 12,245–24,490 pixels.

To ensure consistency between the training data and the generated input representations, a Python-based program was developed to generate training-aligned synthetic floor plans. The process begins by initializing a 256 × 256 canvas on which polygonal room boundaries are drawn using OpenCV, ensuring that the enclosed area falls within the target pixel range. Each generated floor plan features a unique spatial configuration and may include non-Manhattan-style boundaries, ensuring both realism and geometric diversity. As shown in Fig. 10(a), the generated boundary and the area that it encloses—used as input for the CNNs—are represented by the first channel (exterior wall) and the fourth channel (inside area), respectively.

In the input image, only the first and fourth channels contain data, while the second and third channels are initially empty, as illustrated in the input data in Fig. 10(a). The first channel represents the building boundaries, including external walls and the front door, while the fourth channel contains the areas enclosed by these boundaries.

The iterative generative network generates spatial semantics in the third channel, and the encoder–decoder network generates internal walls with noise in the second channel, as shown in the output image with noise in Fig. 10(b). Finally, a denoising algorithm is applied to the second channel (interior wall) to remove noise, resulting in the denoised output shown in Fig. 10(c). These three data share the same set of pixel value mapping relations as given in Table 5, where specific semantic information is encoded by pixel values in different channels.

4.3. Data extraction

Fig. 11 illustrates the process of transforming the CNN-generated floor plan into an intermediate PNG through image processing. In Fig. 11(a), the denoised floor plan generated by the CNNs is presented, while Fig. 11(b) shows the synthesized intermediate PNG.

First, the external walls, internal walls, and front doors from the first and second channels of the CNN-generated floor plan are extracted and merged into the first channel of the intermediate image. Next, semantic information from the third and fourth channels is extracted and merged into the second channel of the intermediate image. The third and fourth channels of the intermediate image are completed manually to include height information and material properties, respectively.

4.4. BIM generation

The proposed framework employs rule-based algorithms and Revit API to transfer a CNN-generated floor plan to the corresponding BIM. By leveraging looping scripts, this process can be fully automated, enabling batch conversion of multiple floor plans into BIMs. Fig. 12 shows two CNN-generated floor plans and the corresponding synthetic BIMs.

Furthermore, by tuning the hyperparameters of the CNNs, different floor plan layouts can be generated from the same boundary input, as illustrated in Fig. 13. The adjustment of CNN hyperparameters and their impact on the generation results are discussed in detail in Section 5.2.

Building upon this capability, multi-story buildings can be created with limited manual intervention. Fig. 14 presents a two-story residential unit, where the base level of the first floor is set to zero (level 0) and the base level of the second floor is set to 3000 mm (level 1) using Revit API.

4.5. Annotation tools

This study focuses on three aspects of automatic annotation: grid generation in floor plans, dimensioning in floor plans, and dimensioning in elevation views. The automatic annotation toolbox is built upon the Revit API query and filter functionalities as described in Algorithm 2 in Section 3.4. Additionally, 2D drawings in different styles can be generated by selecting various built-in rendering modes in Revit. The annotated 2D floor plans shown in this section were rendered in the "Realistic" mode.

Fig. 15 depicts an annotated floor plan with grids and grid dimensions. Different annotation fonts and sizes can be customized, as shown in Fig. 16. Fig. 16(a) demonstrates the annotations of grids, windows, and doors using the "Arial" font in different sizes. In addition, the toolbox can generate the dimensions of various elements and annotations for elevation views, as shown in Fig. 16(b).

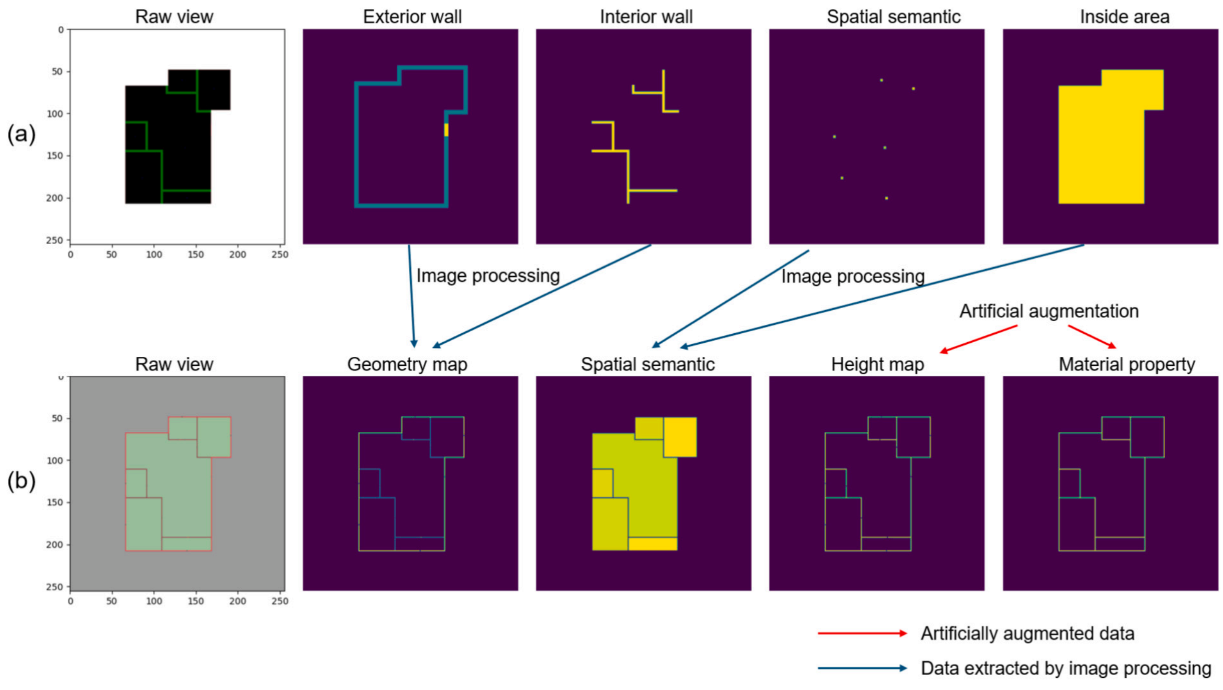


Fig. 11. Creating intermediate image.

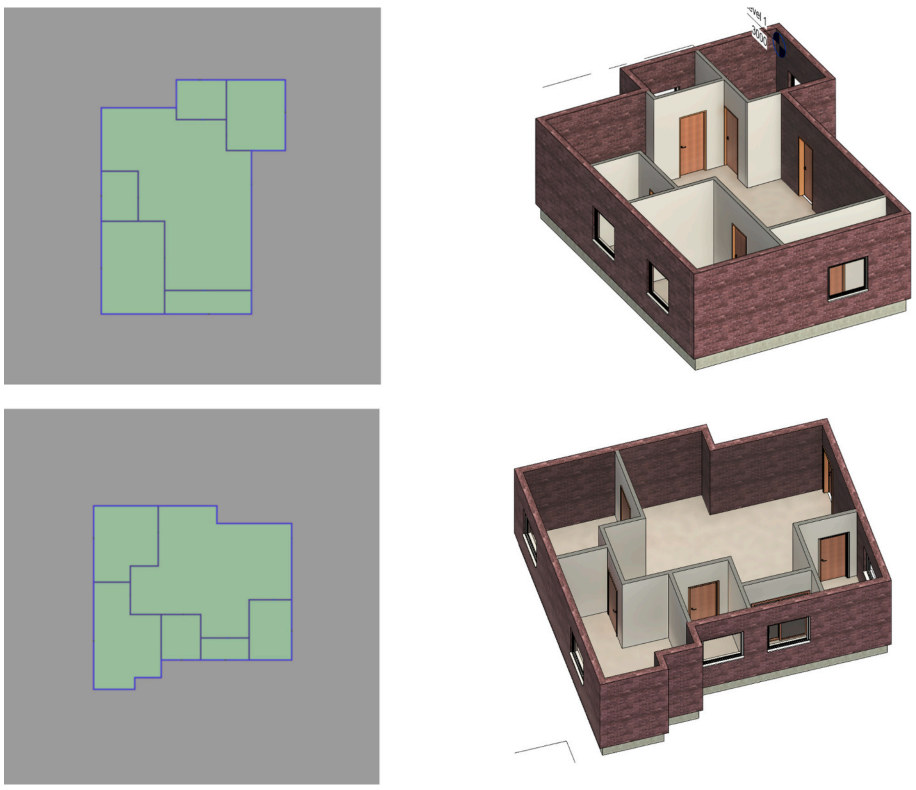


Fig. 12. Two CNN-generated floor plans (left) and corresponding synthetic BIMs (right).

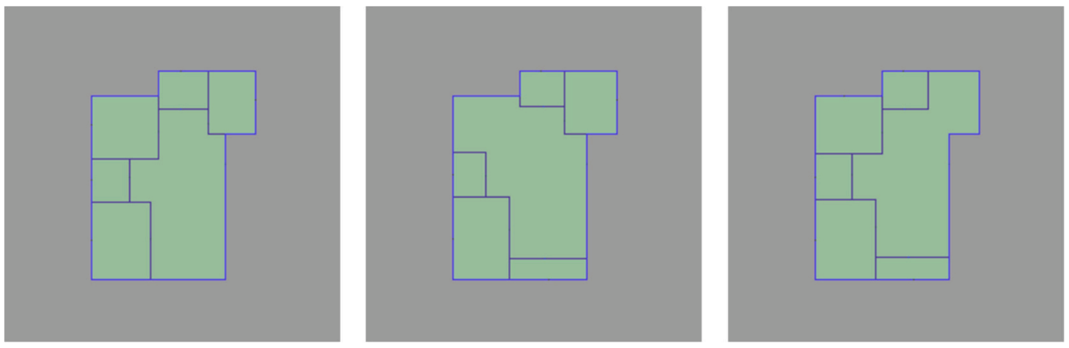


Fig. 13. Three floor plans with identical boundaries but different spatial layouts.

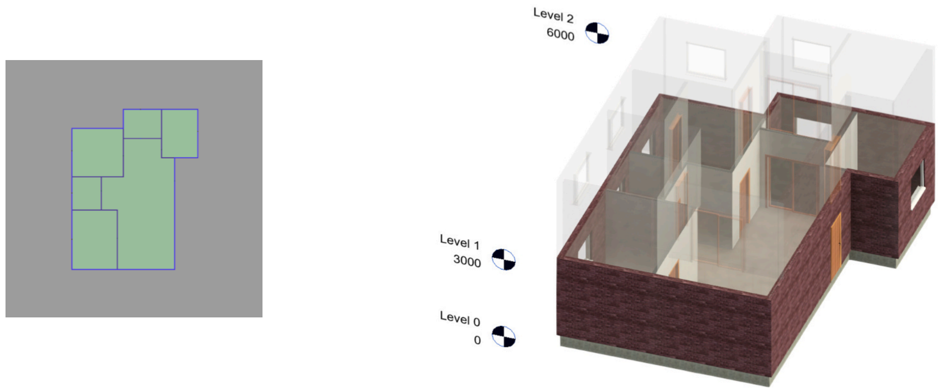
5. Results and discussion

This section provides a comprehensive evaluation of the proposed generation framework and the ResBIM dataset. It begins with quantitative assessments of model accuracy and a study of key hyper-parameters, followed by an examination of BIM generation quality and efficiency. The dataset's characteristics are then analyzed and compared with existing resources, concluding with a discussion that highlights the contributions and limitations of this work.

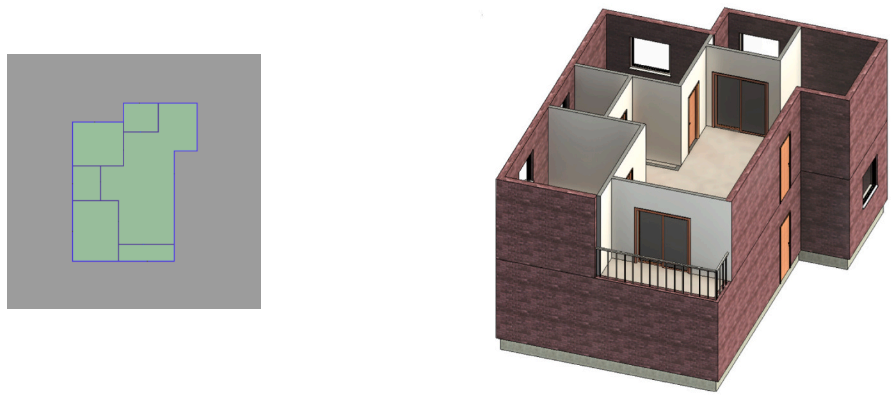
5.1. Quantitative evaluation of model accuracy

For the first network (the modified ResNet-34), the mean absolute error (MAE) was used to evaluate the predicted room centroid positions against the GT centroids, with the error measured in pixels. The trained model achieves an MAE of approximately 5.2 pixels on the test set, along with a room type classification accuracy of 99 %, indicating strong performance in the room localization task. Fig. 17 illustrates the training loss, MAE, and classification accuracy over epochs.

The encoder–decoder network was tasked with pixel-wise wall segmentation, where wall pixels accounted for only about 5 % of the total image area (256 × 256), resulting in a severely imbalanced class distribution. In such cases, commonly used metrics such as IoU tend to underestimate model performance, as even small misclassifications can disproportionately affect the IoU score when the target region is sparse. To address this limitation, the Dice coefficient was used as an evaluation metric [65]. This Dice score provides a more balanced assessment by incorporating both precision and recall, making it particularly suitable for segmentation quality evaluation under extreme class imbalance. The encoder–decoder network achieved 99 % spatial classification accuracy around wall regions, with an average Dice score of approximately 0.52, demonstrating reasonable segmentation capability despite the sparsity of the target class. The progression of training loss, Dice coefficient, and accuracy across epochs is shown in Fig. 18.



(a) Floor plan (left) and corresponding BIM (right) of first floor, with second floor set to 90% transparency



(b) Floor plan (left) and corresponding BIM (right) of second floor

Fig. 14. BIM of two-story residential unit.

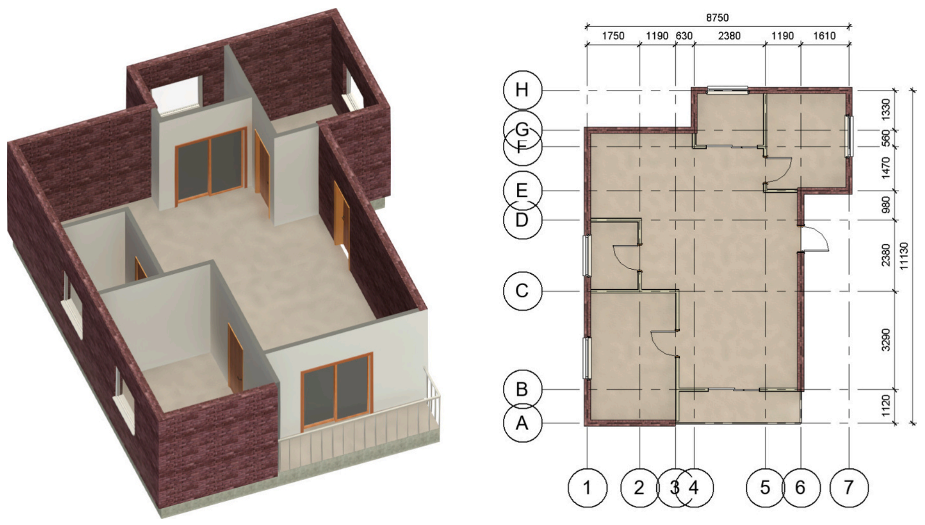
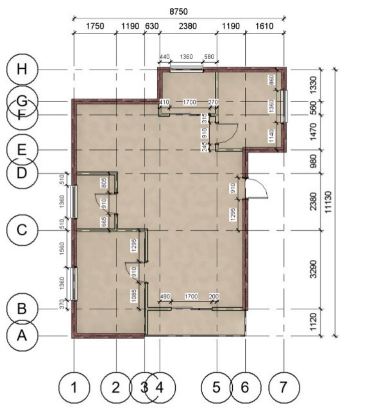


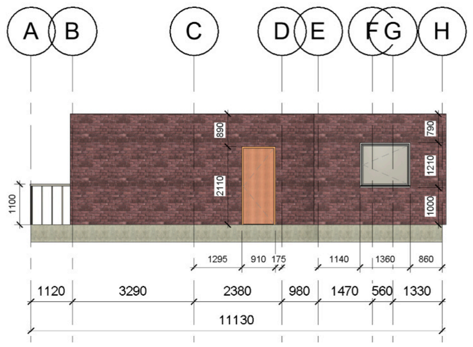
Fig. 15. Synthetic BIM (left) and creation of grids and dimensions in its floor plan (right).

5.2. *Key hyperparameters and their impacts*

The synthetic floor plan generation process relies on several adjustable hyperparameters, as summarized in Table 6. The parameter *mask_size* (M) defines the pixel radius of the local search area used during the feature decoding process; it directly affects the pixel density of the generated floor plan layout. A smaller value of M produces sparser representations with lower pixel density, typically resulting in simpler and more abstract room layouts. In contrast, a larger M expands the search area, enabling the network to capture finer geometric details and produce denser, more complex spatial configurations. Therefore, M serves as a crucial parameter for controlling the granularity of the reconstructed floor plan.

The parameter *room_number* (R) specifies the number of distinct room labels that the modified ResNet-34 architecture attempts to generate, effectively corresponding to the number of rooms in the synthesized floor plan. To ensure consistency with the scale and capacity of the input layout, the network imposes an upper limit on the number of rooms based on the input floor plan's area. If R exceeds this predefined threshold, the actual number of rooms generated is constrained by both


(a) Dimensions of elements in floor plan



(b) Annotations in east elevation view

Fig. 16. Creating other types of annotations in different aspects.

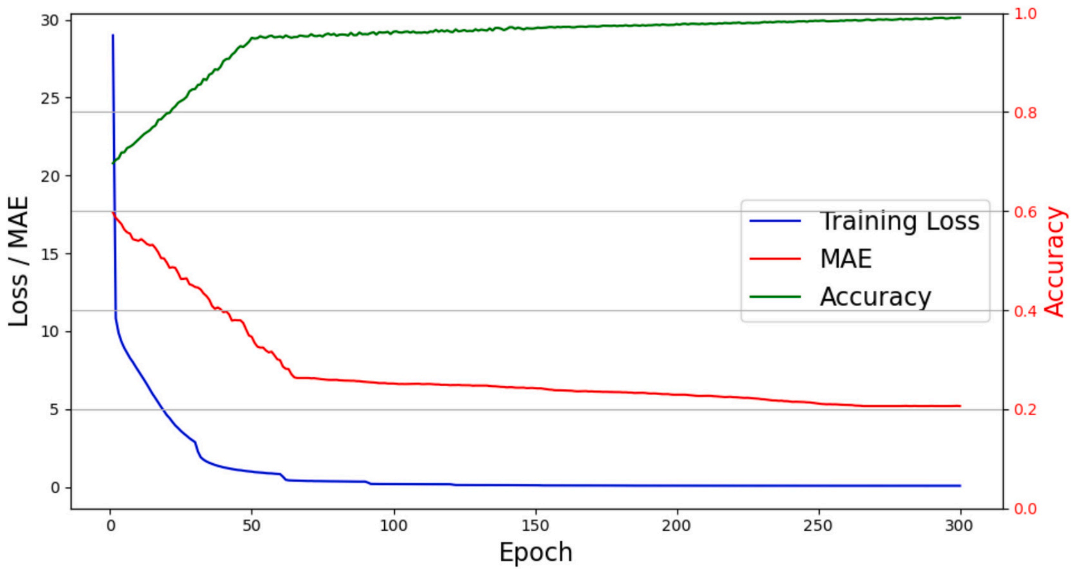


Fig. 17. Training loss, mean absolute error (MAE), and classification accuracy over epochs for modified ResNet-34.

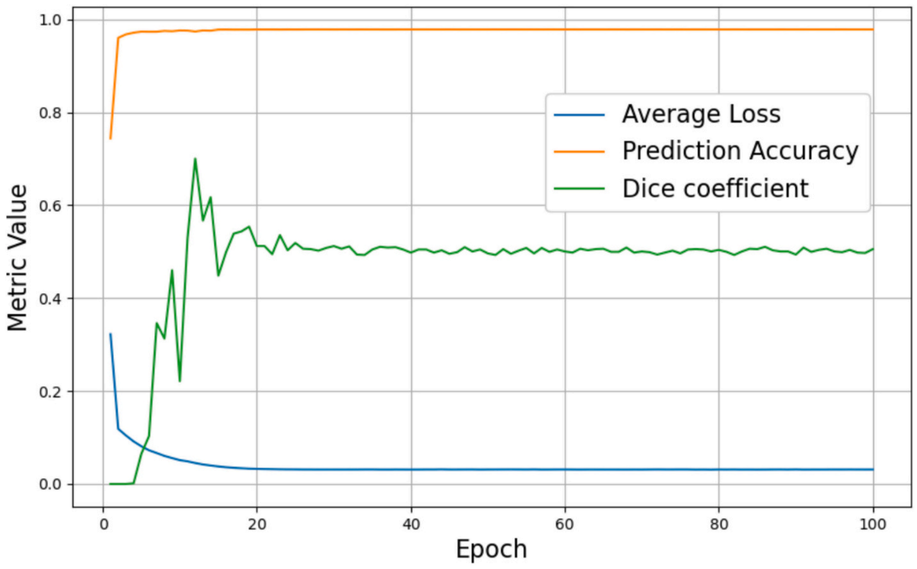


Fig. 18. Training loss, Dice coefficient, and accuracy across epochs for encoder–decoder.

Table 6
Key hyperparameters of CNNs.

Parameter (abbreviation letter)	Data type	Recommended range	Effect on results
mask_size (M)	Integer	6–9	The radius of the local search area. Smaller values lead to sparser layouts; larger values result in more detailed and complex room distributions.
room_num (R)	Integer	2–6	The number of room labels generated by the network represents the number of rooms in the generated floor plan.

Note: This table summarizes the two key hyperparameters. Additional hyperparameters were tested but showed a minimal effect on the model's performance.

the threshold and the effective spatial resolution governed by *M*. Consequently, *R* and *M* jointly regulate the spatial and semantic complexity of the final output.

Fig. 19. illustrates how varying *M* from 2 to 13 affects the generated (non-denoised) floorplans under a fixed floor plan boundary with $R = 10$. In this case, the value of *R* exceeds the threshold imposed by the network, as the maximum number of rooms in the training set is 9. Therefore, the CNNs rely on the value of *M* to determine the number and complexity of rooms. When *M* is between 1 and 6, the generated wall pixels are sparse, resulting in numerous broken and disconnected structures. When *M* ranges from 7 to 10, the model exhibits optimal convergence, producing clear layouts with minimal noise. However, when *M* exceeds 10, the number of wall pixels increases, but many of them are noise.

Fig. 20. presents the effect of increasing *R* from 2 to 6 under the same floor plan boundary, with *M* fixed at 7 and 9, both within the optimal convergence range. When *R* is below the network's room-number threshold, it directly determines the number of rooms generated. Meanwhile, different values of *M* under the same *R* produce variations in room layout and spatial configuration.

Maintaining a consistent boundary is particularly valuable for applications in modular architecture and multi-story building design. By using different combinations of *M* and *R*, the model can generate different floor plan layouts with varying room counts under the same boundary constraints, as demonstrated in Section 4.4.

5.3. Quality and efficiency of BIM generation

To ensure the quality of the generated BIMs, Revit API was utilized in conjunction with Revit's built-in validation tools to check each converted instance systematically according to several key criteria: (1) wall continuity and integrity—verifying that all interior and exterior walls are fully connected, closed, and free from gaps, floating segments, overlaps, or redundant wall sections; (2) opening–host relationship—ensuring that each door or window (opening) is correctly associated with a host wall, with valid position and alignment within the wall boundaries and thickness; (3) spatial constraints—checking for door swings or window placements that might result in collisions with other doors, walls, or windows. During the conversion process, any BIM model failing to meet these requirements triggered a warning window in Revit, as illustrated in Fig. 21(a).

In a task that involved generating 1000 floor plans, the trained networks took an average of 1.2 s to produce one noisy floor plan and 1.6 s to denoise it. On average, the framework generated BIMs at a speed of approximately six models per minute. In this study, 1100 distinct boundaries were generated using Python and OpenCV and served as inputs to the CNNs; correspondingly, the CNNs produced 1100 synthetic floor plans. When these floor plans were converted into BIMs using Revit API, 1027 conversions were successful without any warning windows, resulting in an overall conversion accuracy of 93.4 %. The entire process took around four hours to complete. The 73 failed conversions were caused primarily by dimensional errors, excessive pixel density, and insufficient pixel coverage, as shown in Fig. 21.

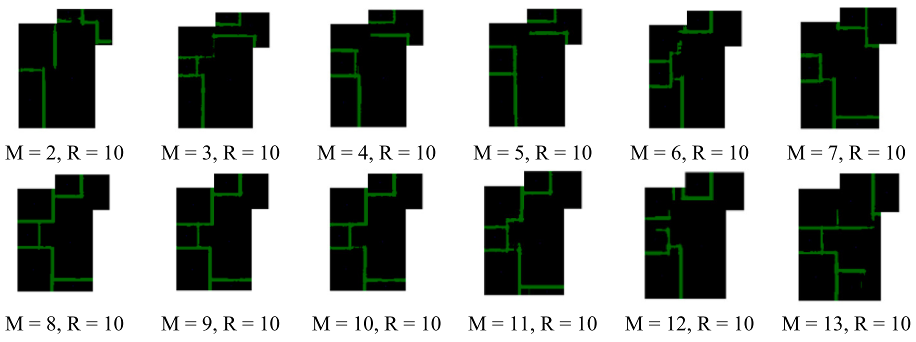


Fig. 19. Effects of increasing mask size (*M*) under a fixed boundary and constant room number (*R*).

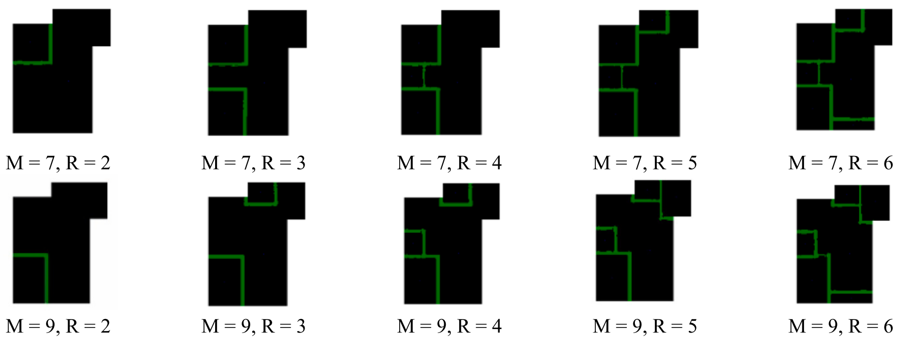
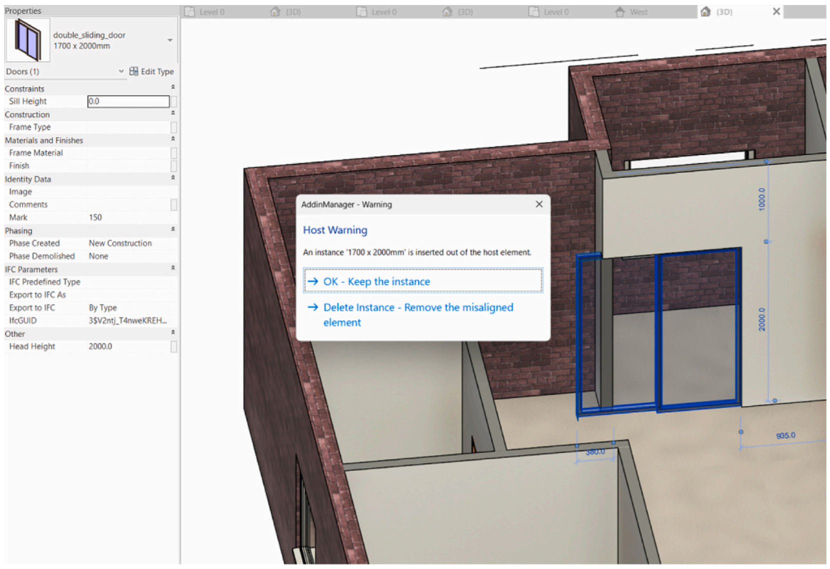


Fig. 20. Effects of increasing *R* from 2 to 6 on floor plan generation under a fixed boundary, with *M* set to 7 and 9 (within the optimal convergence range).



double_sliding_door
1700 x 2000mm

Constraints	
Sill Height	0.0
Construction	
Frame Type	
Materials and Finishes	
Frame Material	
Finish	
Identity Data	
Image	
Comments	
Mark	150
Phasing	
Phase Created	New Construction
Phase Demolished	None
IFC Parameters	
IFC Predefined Type	
Export to IFC As	
Export to IFC	By Type
IfcGUID	3$V2ntj_T4nweKREH...
Other	
Head Height	2000.0

AddinManager - Warning

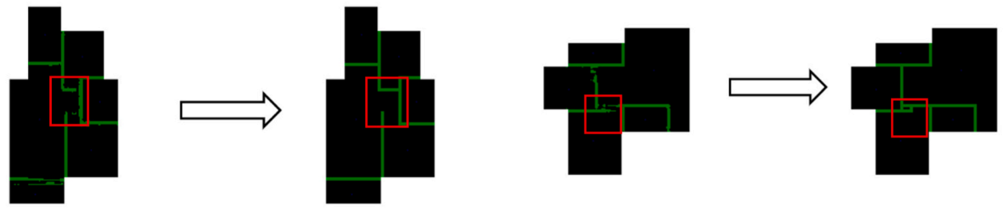
Host Warning

An instance '1700 x 2000mm' is inserted out of the host element.

→ OK - Keep the instance

→ Delete Instance - Remove the misaligned element

(a) Case of insufficient wall length



(b) Insufficient pixels (c) Disconnected walls after denoising (d) Excessive pixels (e) Additional space after denoising

Fig. 21. Three types of failure cases: (a) dimensional error, (b) disconnected walls resulting from small ***mask_size*** and (c) corresponding denoised output, and (d) redundant spaces caused by large ***mask_size*** and (e) denoised result.

Dimensional errors were encountered during the conversion of CNN-generated raster floor plans to BIMs. Because these floor plans are pixel-based, it is necessary to map pixel values to real-world distances; to maintain scale consistency with the training dataset, in this study each pixel represents 70 mm in the real world. However, some generated floor plans included walls with fewer than 10 pixels; this resulted in the wall length being less than 700 mm, which is smaller than the standard minimum width of a sliding door in the default Revit library. When attempting to insert a door into a wall narrower than the door width, Revit generates an error, preventing proper placement of the door, as illustrated in Fig. 21(a).

Failures were also observed in the encoder–decoder network. The encoder–decoder generates walls in the form of discrete pixels, which—despite containing noise—serve to partition space and define individual rooms. As analyzed in Section 5.2, if the *mask_size* hyperparameter is set too small, then the number of wall pixels becomes insufficient, resulting in broken or disconnected walls after denoising, as highlighted in Fig. 21(b) and (c). Conversely, if *mask_size* is too large, then too many wall pixels are produced, which leads to redundant wall structures after denoising, as highlighted in Fig. 21(d) and (e).

5.4. Characteristics and usage of dataset

ResBIM is a synthetic dataset designed specifically for research in BIM automation and 2D-to-BIM reconstruction. The dataset comprises 1027 paired samples, each consisting of a fully parametric 3D BIM (RVT

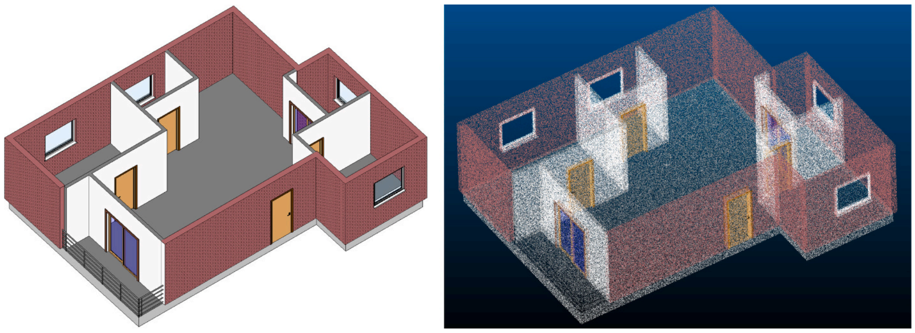


Fig. 22. BIM in the ResBIM dataset (left) and corresponding RGB point cloud (right) in CloudCompare.

format) and its corresponding annotated 2D floor plan. All data were generated and validated using automated workflows in Autodesk Revit to ensure both consistency and scalability. ResBIM is intended for a wide range of applications, including multi-platform BIM interoperability, serving as a foundation for benchmarking 2D-to-BIM algorithms, and the development of computer vision models for architectural analysis. The dataset supports a wide range of use cases, including but not limited to the following.

- *Multi-platform compatibility.* The BIM generation in this study utilizes Revit API to produce RVT-format models within Autodesk Revit. Although RVT is a proprietary format with software dependency, Revit—as one of the mainstream BIM software solutions in the architecture, engineering, and construction (AEC) domain—provides various built-in tools for format conversion and export. Each BIM in the dataset can be exported seamlessly to the IFC format, enabling compatibility with multiple platforms.
- *2D-to-BIM evaluation and benchmarking support.* ResBIM supports evaluation across the two major stages of the 2D-to-BIM workflow. In the data extraction stage, each annotated floor plan provides semantic categories, textual labels, and geometric elements, enabling the measurement of detection rate, pixel- and class-level accuracy, and OCR accuracy. In the BIM generation stage, each paired BIM serves as reference data, allowing reconstruction fidelity to be assessed using widely adopted metrics such as point cloud IoU and IFC-based component matching. Specifically, since BIMs are semantically rich 3D representations, they can be directly converted into IFC files, RGB-preserving point clouds (e.g., PLY), or meshes (e.g., OBJ). Fig. 22 illustrates an RGB point cloud converted from a BIM in the ResBIM dataset, used to compute IoU for 2D-to-BIM algorithms. Beyond these stage-wise assessments, ResBIM also supports end-to-end evaluation—from floor plan recognition to the generation of the resulted BIM—enabling comprehensive benchmarking of accuracy, runtime efficiency, and overall system performance.
- *Obtaining various types of 2D drawings.* Because BIMs are semantically rich 3D representations, all BIMs in the dataset can be converted into various 2D views such as floor plans, elevations, and cross-sections. Combined with the provided annotation toolbox, these views can be exported conveniently from Revit with annotations. The resulting 2D drawings serve multiple research purposes, such as training OCR models to recognize different font styles or training CNNs to detect openings in elevation views. As illustrated in Fig. 16 (a) and (b), the paired floor plan and east elevation view are directly derived from the BIM model. This also addresses a common limitation highlighted in previous studies: the lack of annotated elevation drawings with height information, which has largely restricted 2D-to-BIM data extraction to planar geometry instead of multi-view representations [16,42].

5.5. Dataset comparison

To better evaluate the contribution of the proposed ResBIM dataset, two comparative analyses are presented: (1) against existing floor plan datasets, and (2) against BIM-based datasets. Numerous large-scale publicly available point cloud datasets based on real-world building scans, such as ScanNet [66] and S3DIS [67], have played an important role in scan-to-BIM and SLAM research. However, because these datasets provide only point cloud data without corresponding 2D floor plans or BIMs, they are not applicable to the present 2D-to-BIM task and are therefore excluded from the dataset comparison.

Table 7 summarizes a comparison between ResBIM and representative floor plan datasets. Among these datasets, real-world floor plans are often sourced from housing rental companies for marketing purposes. These drawings typically focus on layout and frequently suffer from incomplete information or distorted proportions. Fig. 23(a) shows a real-world plan from the CVC-FP [23] dataset, where room dimensions are annotated using area values (e.g., 26.18 m^2). As a result, the exact spatial dimensions (e.g., length and width) of each room are difficult to determine. Fig. 23(b) displays a real-world plan from the SydneyHouse [24] dataset, where the red-highlighted annotation indicates the living room size as 6.8 × 4.2 (assuming a rectangular shape). However, the actual room shape is irregular, and certain rooms such as the laundry and bathroom lack dimensional annotations altogether.

In many augmented datasets, original elements present in real-world drawings—such as dimension annotations and furniture—are typically removed during the augmentation process. As shown in Fig. 23(c), a floor plan from the Tell2Design [35] dataset illustrates this characteristic. Such augmented floor plans are often created for classification tasks in computer vision and usually lack textual annotations and rarely include RF 3D models.

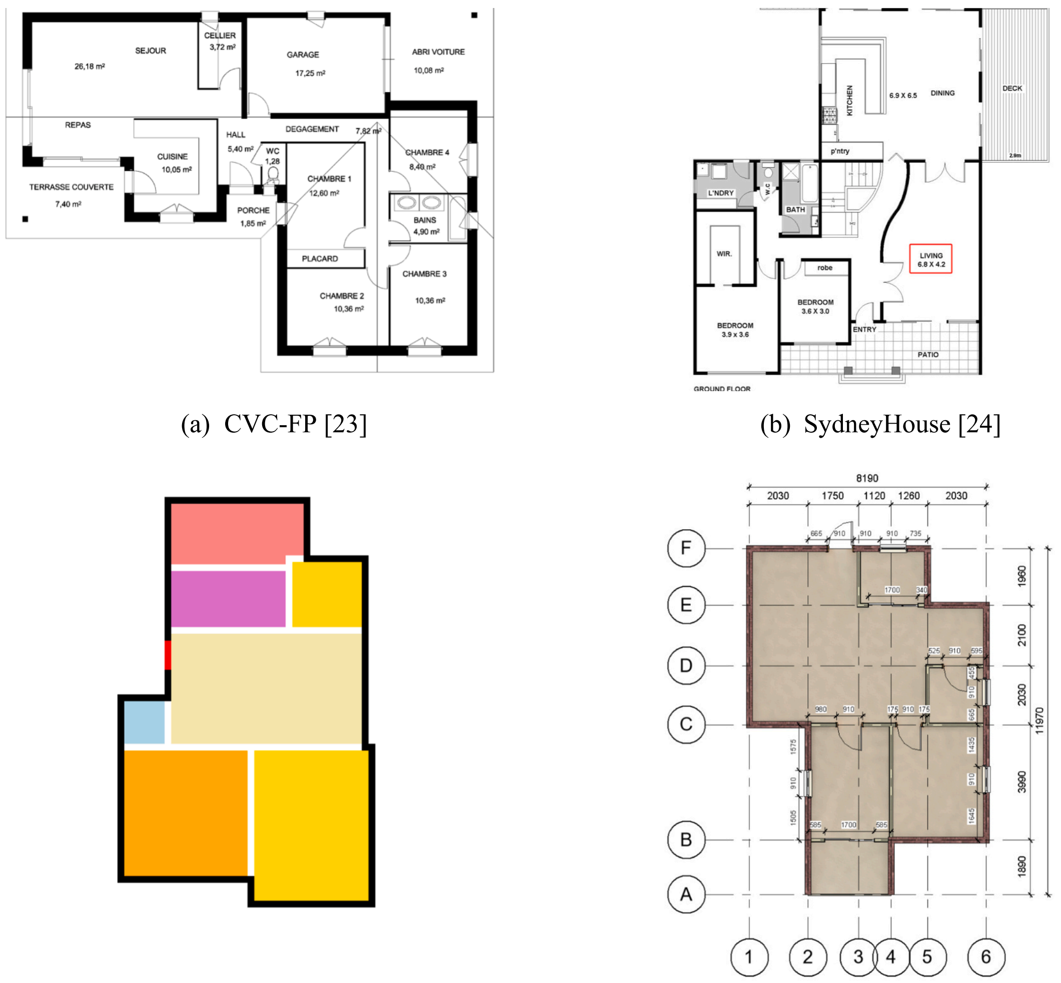
In contrast, ResBIM is generated within Autodesk Revit, which supports customizable dimension annotations at varying levels of detail, as shown in Fig. 23(d). Because of the 3D nature of a BIM, in addition to generating floor plans, it can also produce elevation views as shown in Fig. 16(b)—a data type that remains rare in existing datasets. The RF BIMs in the dataset allow researchers to evaluate the accuracy and generalizability of their algorithms using standard metrics such as point-to-point distance and IoU.

Table 8 compares ResBIM with existing datasets that provide BIMs. Prior datasets in this category are typically derived from laser-scanned point clouds of real-world buildings and are designed primarily for scan-to-BIM tasks. While these datasets offer higher geometric and semantic complexity because of their real-world origin, they often lack

Table 7
Related datasets containing floor plans.*1, *2

Dataset	Year	Data source*^{1}	3D data*^{2}	Annotation type	Dimension	Furniture	Height information
CVC-FP [23]	2015	RW	×	Text	√	√	×
SydneyHouse [24]	2016	RW	×	Text	√	√	×
ROBIN [25]	2017	A	×	×	×	√	×
R2V [26]	2017	A	×	Text, RGB	×	√	×
CubiCasa5K [27]	2019	RW	×	Text	×	√	×
RPLAN [28]	2019	A	×	RGB	×	×	×
HouseExpo [29]	2020	S	×	×	×	×	×
Structure3D [30]	2020	A	RF	RGB	×	×	√
ZInD [31]	2021	RW	×	Text	√	×	×
FloorPlanCAD [32]	2022	RW	×	RGB	√	√	×
MLSTRUCT-FP [34]	2023	A	×	×	×	×	×
Tell2Design [35]	2023	A	×	RGB	×	×	×
MSD [37]	2024	A	×	RGB	×	×	×
ResBIM	2025	S	RF	Text, RGB	√	×	√

*^{1} RW = real-world data, A = augmented based on real-world data, S = synthetic data.
*^{2} RF = reference model.



(a) CVC-FP [23]

(b) SydneyHouse [24]

(c) Tell2Design [35]

(d) ResBIM

Fig. 23. Comparison of 2D drawings in different datasets.

Table 8
Related datasets containing BIMs.*1, *2

Dataset	Year	Number of BIMs (areas)	Data source*1	3D data*2	2D drawing availability	Scalability and extensibility	Target task
Indoor PC/BIM [33]	2023	5 (No mention)	RW	GT	×	Low	Scan-to-BIM Scan-vs-BIM
BIMNet [38]	2025	25 (8710 m^2)	RW	GT	×	Low	Scan-to-BIM
ResBIM	2025	1 k (12,000 m^2)	S	RF	√	High	2D-to-BIM

*1 RW = real-world data, **S** = synthetic data.
*2 GT = ground-truth model, RF = reference model.

paired 2D annotations, limiting their applicability for 2D-to-BIM research. In contrast, ResBIM is designed specifically for the 2D-to-BIM task and includes labeled 2D floor plans that are consistent with the associated BIMs. As a synthetic dataset, ResBIM offers high scalability, allowing the data volume to be expanded with minimal manual effort—an essential property for training large-scale models. While ResBIM offers high scalability and precise 2D-3D correspondences, it does not fully replicate the complexity and diversity of real-world architectural data, a limitation discussed in Section 5.6.

5.6. Discussion

Existing public datasets for 2D-to-BIM research are fragmented. Real-world floor plan datasets are mainly designed for computer vision tasks such as image classification or pixel-level parsing and thus lack 3D models or the annotations needed in AEC workflows. Conversely, BIM-containing datasets are largely built for scan-to-BIM research and generally do not provide corresponding 2D drawings. Attempts to generate synthetic floor plans and convert them into BIMs have been hindered by the instability of generated layouts, which prevents large-scale automated production. In addition, semantic and height information in current 2D-to-BIM workflows is usually assigned through defaults or manual configuration, restricting dimensional and material diversity. The absence of annotated elevation views has further confined most extraction methods to planar geometry, creating a significant gap for tasks that require accurate and large-scale mapping between 2D and 3D representations.

ResBIM dataset directly addresses this gap by providing over 1000 paired floor plans and fully parametric BIMs, generated through automated Revit workflows. These pairs enable not only training of advanced AI models but also reproducible evaluation of 2D-to-BIM algorithms using metrics reliant on RF BIMs. With the accompanying annotation toolbox, BIMs can be exported into multiple 2D views—including floor plans, elevations, and sections—supporting research in OCR, vision-based recognition, and multi-view reconstruction. In doing so, ResBIM tackles two critical limitations: the lack of paired 2D-3D datasets and the absence of annotated elevation views.

Beyond the dataset, the proposed four-step framework introduces methodological advances. Boundary constraints during floor plan generation stabilize large-scale automated workflows, overcoming the instability issues reported in earlier data-driven methods (e.g., Section 2.3). The extensible semantic material library enriches variability in materials, dimensions, and components, moving beyond reliance on defaults and enabling more representative scenarios. Together, these design choices make the framework scalable, platform-compatible, and adaptable across diverse data sources.

By filling a critical data gap, this work lays the groundwork for future research in automated BIM reconstruction and supports scaling to more diverse and complex architectural scenarios. Furthermore, the proposed generation pipeline demonstrates considerable cost-effectiveness in time and labor. It is capable of generating, denoising, and converting over 1000 floor plans into BIMs within approximately four hours while maintaining a success rate exceeding 93 %, the framework substantially reduces manual workloads and modeling time relative to conventional BIM workflows. As the dataset continues to evolve, including the integration of a wider range of architectural styles and elements, the proposed framework has the potential to further drive AI innovation in the AEC domain.

5.6.1.1. Limitations and positioning. Despite these advantages, certain limitations remain. The dataset primarily represents residential layouts with straight walls, which limits applicability to industrial or public facilities. Elements such as staircases, MEP systems, and façades are also absent, restricting its use in research targeting these features. Moreover, synthetic datasets cannot fully capture the drafting irregularities or noise of real-world drawings, which may affect model generalization. Nonetheless, as shown in related domains (e.g., RPLAN [28], SUNCG [68]), synthetic datasets with controlled assumptions can serve as useful early-stage resources. Accordingly, ResBIM should be viewed as a supplementary dataset rather than a replacement for real data, aiming to: (1) expand data availability in early-stage 2D-to-BIM research, (2) provide a controlled baseline for method comparison, and (3) promote standardized benchmarking protocols for real or hybrid datasets.

6. Conclusions

This paper addressed a critical gap in 2D-to-BIM research by introducing a procedural BIM synthesis framework that integrates deep learning–based floor plan generation with structured 2D-3D mapping. Alongside the framework, the open-source ResBIM dataset and its annotation toolbox provide the AEC community with scalable and reproducible resources for diverse applications. The key contributions are as follows:

- *Unified procedural framework.* A scalable pipeline that formalizes the mapping from 2D drawings to BIMs, integrating deep learning with rule-based synthesis for large-scale automation.
- *Paired 2D-3D dataset.* ResBIM offers paired 2D-3D data with detailed annotations, filling a gap in the literature and providing a dataset that supports reproducible evaluation of 2D-to-BIM algorithms.
- *Open-source tools for reproducibility.* Annotation utilities and dataset design principles that support semantically consistent benchmarks and foster community-driven research in the AEC domain.

Limitations remain, particularly the dataset's focus on residential layouts with rectilinear walls and the synthetic nature of the data, which cannot fully capture the drafting inconsistencies of real-world drawings. These constraints may affect model generalization across broader building typologies and more complex architectural features.

Future work should focus on expanding building typologies, incorporating complex architectural elements, and integrating more advanced generative models capable of balancing diversity with geometric stability. Broader efforts to establish large-scale annotated datasets—both synthetic and real—and standardized evaluation protocols will be essential to ensure objective comparison and accelerate the adoption of automated BIM generation in research and industry.

CRediT authorship contribution statement

Xing Liang: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Nobuyoshi Yabuki:** Writing – review & editing, Supervision, Project administration. **Tomohiro Fukuda:** Writing – review & editing, Supervision, Project administration.

Declaration of generative AI and AI-assisted technologies in the writing process

In the process of preparing this work, the authors used ChatGPT to improve the readability and language of the text. Following the use of this tool, the authors reviewed and edited the content as necessary and took full responsibility for the paper's contents.

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.

Data availability

The data has been already published at https://github.com/RogerLiang0725/ResBIM.

References

[1] H. El Hafdaoui, A. Khallaayoun, K. Ouazzani, Activity and efficiency of the building sector in Morocco: a review of status and measures in Ifrane, AIMS Energy (2023) 454–485, https://doi.org/10.3934/energy.2023024.

[2] N.Z. Che-Ghani, N.E. Myeda, A.S. Ali, Efficient operation and maintenance (O&M) framework in managing stratified residential properties, J. Facil. Manag. (2023) 609–634, https://doi.org/10.1108/JFM-10-2021-0124.

[3] F. Prideaux, R.H. Crawford, K. Allacker, A. Stephan, Approaches for assessing embodied environmental effects during the building design process, in: IOP Conf Ser Earth Environ Sci, Institute of Physics, 2023 12053, https://doi.org/10.1088/1755-1315/1196/1/012053.

[4] M. Deng, C.C. Menassa, V.R. Kamat, From BIM to digital twins: a systematic review of the evolution of intelligent building representations in the AEC-FM industry, J. Inf. Technol. Constr. (2021) 58–83, https://doi.org/10.36680/J.ITCON.2021.005.

[5] Y. Cao, S.N. Kamaruzzaman, N.M. Aziz, Building information Modeling (BIM) capabilities in the operation and maintenance phase of green buildings: a systematic review, Buildings (2022) 830, https://doi.org/10.3390/buildings12060830.

[6] P. Schönfelder, A. Aziz, B. Faltin, M. König, Automating the retrospective generation of As-is BIM models using machine learning, Autom. Constr. (2023), https://doi.org/10.1016/j.autcon.2023.104937, 104937.

[7] Q. Lu, X. Xie, J. Heaton, A.K. Parlikad, J. Schooling, From BIM towards digital twin: Strategy and future development for smart asset management, in: Studies in Computational Intelligence, Springer Verlag, 2020, pp. 392–404, https://doi.org/10.1007/978-3-030-27477-1_30.

[8] S.A. Adekunle, C. Aigbavboa, O.A. Ejohwomu, SCAN TO BIM: a systematic literature review network analysis, in: IOP Conference Series: Materials Science and Engineering, 2022, p. 12057, https://doi.org/10.1088/1757-899X/1218/1/012057.

[9] Q. Tushar, M.A. Bhuiyan, G. Zhang, T. Maqsood, An integrated approach of BIM-enabled LCA and energy simulation: the optimized solution towards sustainable development, J. Clean. Prod. (2021) 125622, https://doi.org/10.1016/j.jclepro.2020.125622.

[10] M. Kozlovska, S. Petkanic, F. Vranay, D. Vranay, Enhancing energy efficiency and building performance through BEMS-BIM integration, Energies (2023) 6237, https://doi.org/10.3390/en16176327.

[11] L. Gimenez, J.L. Hippolyte, S. Robert, F. Suard, K. Zreik, Review: reconstruction of 3D building information models from 2D scanned plans, J. Build. Eng. (2015) 24–35, https://doi.org/10.1016/j.jobe.2015.04.002.

[12] Q. Lu, S. Lee, Image-based technologies for constructing as-is building information models for existing buildings, J. Comput. Civ. Eng. (2017), https://doi.org/10.1061/(ASCE)CP.1943-5487.0000652, 4017005.

[13] L. Klein, N. Li, B. Becerik-Gerber, Imaged-based verification of as-built documentation of operational buildings, Autom. Constr. (2012) 161–171, https://doi.org/10.1016/j.autcon.2011.05.023.

[14] M. Urbieta, M. Urbieta, T. Laborde, G. Villarreal, G. Rossi, Generating BIM model from structural and architectural plans using artificial intelligence, J. Build. Eng. (2023) 107672, https://doi.org/10.1016/j.jobe.2023.107672.

[15] J. Lim, P. Janssen, R. Stouffs, Automated generation of BIM models from 2D CAD drawings, in: CAADRIA 2018 - 23rd International Conference on Computer-Aided Architectural Design Research in Asia: Learning, Prototyping and Adapting, The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), 2018, pp. 61–70, https://doi.org/10.52842/conf.caadria.2018.2.061.

[16] C. Zhang, Y. Zou, J. Dimyadi, A systematic review of automated BIM modelling for existing buildings from 2D documentation, in: ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, 2021, pp. 220–226, https://doi.org/10.22260/ISARC2021/0032.

[17] B. Bortoluzzi, I. Efremov, C. Medina, D. Sobieraj, J.J. McArthur, Automating the creation of building information models for existing buildings, Autom. Constr. (2019), https://doi.org/10.1016/j.autcon.2019.102838, 102838.

[18] M.K. Dixit, V. Venkatraj, M. Ostadalimakhmalbaf, F. Pariafsai, S. Lavy, Integration of facility management and building information modeling (BIM), Facilities (2019) 455–483, https://doi.org/10.1108/F-03-2018-0043.

[19] R. Volk, J. Stengel, F. Schultmann, Building information Modeling (BIM) for existing buildings - literature review and future needs, Autom. Constr. (2014) 109–127, https://doi.org/10.1016/j.autcon.2013.10.023.

[20] P.N. Pizarro, N. Hitschfeld, I. Sipiran, J.M. Saavedra, Automatic floor plan analysis and recognition, Autom. Constr. (2022), https://doi.org/10.1016/j.autcon.2022.104348, 104348.

[21] G. Zhang, Y.Q. Chen, A metric for evaluating 3d reconstruction and mapping performance with no ground truthing, in: International Conference on Image Processing, ICIP, IEEE Computer Society, 2021, pp. 3178–3182, https://doi.org/10.1109/ICIP42928.2021.9506329.

[22] N. Abreu, A. Pinto, A. Matos, M. Pires, Procedural point cloud modelling in scan-to-BIM and scan-vs-BIM applications: a review, ISPRS Int. J. Geo Inf. (2023) 260, https://doi.org/10.3390/ijgi12070260.

[23] L.-P. de las Heras, O.R. Terrades, S. Robles, G. Sánchez, CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool, Int. J. Doc. Anal. Recognit. (IJDAR) (2015) 15–30, https://doi.org/10.1007/s10032-014-0236-5.

[24] H. Chu, S. Wang, R. Urtasun, S. Fidler, Housecraft: Building houses from rental ads and street views, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11–14, 2016, Proceedings, Part VI 14, Springer, Cham, 2016, pp. 500–516, https://doi.org/10.1007/978-3-319-46466-4_30.

[25] D. Sharma, N. Gupta, C. Chattopadhyay, S. Mehta, Daniel: a deep architecture for automatic analysis and retrieval of building floor plans, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017, pp. 420–425, https://doi.org/10.1109/ICDAR.2017.76.

[26] C. Liu, J. Wu, P. Kohli, Y. Furukawa, Raster-to-vector: revisiting floorplan transformation, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2195–2203. https://openaccess.thecvf.com/content_ICCV_2017/papers/Liu_Raster-To-Vector_Revisiting_Floorplan_ICCV_2017_paper.pdf (accessed June 9, 2025).

[27] A. Kalervo, J. Ylioinas, M. Häikiö, A. Karhu, J. Kannala, Cubicasa5k: a dataset and an improved multi-task model for floorplan image analysis, in: Image Analysis: 21st Scandinavian Conference, SCIA 2019, Norrköping, Sweden, June 11–13, 2019, Proceedings 21, Springer, Cham, 2019, pp. 28–40, https://doi.org/10.1007/978-3-030-20205-7_3.

[28] W. Wu, X.M. Fu, R. Tang, Y. Wang, Y.H. Qi, L. Liu, Data-driven interior plan generation for residential buildings, ACM Trans. Graph. (2019) 1–12, https://doi.org/10.1145/3355089.3356556.

[29] T. Li, D. Ho, C. Li, D. Zhu, C. Wang, M.Q.-H. Meng, Houseexpo: a large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 5839–5846, https://doi.org/10.1109/IROS45743.2020.9341284.

[30] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao, Z. Zhou, Structured3d: a large photo-realistic dataset for structured 3d modeling, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16, Springer, Cham, 2020, pp. 519–535, https://doi.org/10.1007/978-3-030-58545-7_30.

[31] S. Cruz, W. Hutchcroft, Y. Li, N. Khosravan, I. Boyadzhiev, S. Bing Kang, Zillow indoor dataset: annotated floor plans with 360 o panoramas and 3d room layouts, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2133–2143. https://openaccess.thecvf.com/content/CVPR2021/papers/Cruz_Zillow_Indoor_Dataset_Annotated_Floor_Plans_With_360deg_Panoramas_and_CVPR_2021_paper.pdf (accessed June 9, 2025).

[32] Z. Fan, L. Zhu, H. Li, X. Chen, S. Zhu, P. Tan, Floorplancad: a large-scale cad drawing dataset for panoptic symbol spotting, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10128–10137. https://openaccess.thecvf.com/content/ICCV2021/supplemental/Fan_FloorPlanCAD_A_Large-Scale_ICCV_2021_supplemental.pdf (accessed June 10, 2025).

[33] N. Abreu, R. Souza, A. Pinto, A. Matos, M. Pires, Labelled indoor point cloud dataset for BIM related applications, Data (2023) 101, https://doi.org/10.3390/data8060101.

[34] P.N. Pizarro, N. Hitschfeld, I. Sipiran, Large-scale multi-unit floor plan dataset for architectural plan analysis and recognition, Autom. Constr. (2023), https://doi.org/10.1016/j.autcon.2023.105132, 105132.

[35] S. Leng, Y. Zhou, M.H. Dupty, W.S. Lee, S.C. Joyce, W. Lu, Tell2Design: a dataset for language-guided floor plan generation, ArXiv (2023), https://doi.org/10.48550/arXiv.2311.15941. Preprint.

[36] H. Huang, Z. Qiao, Z. Yu, C. Liu, S. Shen, F. Zhang, H. Yin, SLABIM: a SLAM-BIM coupled dataset in HKUST main building, ArXiv (2025), https://doi.org/10.48550/arXiv.2502.16856. Preprint.

[37] C. van Engelenburg, F. Mostafavi, E. Kuhn, Y. Jeon, M. Franzen, M. Standfest, J. van Gemert, S. Khademi, MSD: A benchmark dataset for floor plan generation of building complexes, in: European Conference on Computer Vision, Springer, Cham, 2024, pp. 60–75, https://doi.org/10.1007/978-3-031-73636-0_4.

[38] Y. Liu, H. Huang, G. Gao, Z. Ke, S. Li, M. Gu, Dataset and benchmark for as-built BIM reconstruction from real-world point cloud, Autom. Constr. (2025), https://doi.org/10.1016/j.autcon.2025.106096, 106096.

[39] B. Yang, B. Liu, D. Zhu, B. Zhang, Z. Wang, K. Lei, Semiautomatic structural BIM-model generation methodology using CAD construction drawings, J. Comput. Civ. Eng. (2020), https://doi.org/10.1061/(asce)cp.1943-5487.0000885, 04020006.

[40] R. Tang, Y. Wang, D. Cosker, W. Li, Automatic structural scene digitalization, PLoS One (2017), https://doi.org/10.1371/journal.pone.0187513 e0187513.

[41] M. Yin, L. Tang, T. Zhou, Y. Wen, R. Xu, W. Deng, Automatic layer classification method-based elevation recognition in architectural drawings for reconstruction of 3D BIM models, Autom. Constr. (2020) 103082, https://doi.org/10.1016/j.autcon.2020.103082.

[42] Y. Zhao, X. Deng, H. Lai, Reconstructing BIM from 2D structural drawings for existing buildings, Autom. Constr. (2021), https://doi.org/10.1016/j.autcon.2021.103750, 103750.

[43] Y. Byun, B.S. Sohn, ABGS: a system for the automatic generation of building information models from two-dimensional CAD drawings, Sustainability (2020) 6713, https://doi.org/10.3390/SU12176713.

[44] Q. Lu, L. Chen, S. Li, M. Pitt, Semi-automatic geometric digital twinning for existing buildings based on images and CAD drawings, Autom. Constr. (2020), https://doi.org/10.1016/j.autcon.2020.103183, 103183.

[45] S. Feist, L. Jacques de Sousa, L. Sanhudo, J. Poças Martins, Automatic reconstruction of 3D models from 2D drawings: a state-of-the-art review, Eng (2024) 784–800, https://doi.org/10.3390/eng5020042.

[46] P. Schönfelder, F. Stebel, N. Andreou, M. König, Deep learning-based text detection and recognition on architectural floor plans, Autom. Constr. (2024), https://doi.org/10.1016/j.autcon.2023.105156, 105156.

[47] H. Jang, K. Yu, J.H. Yang, Indoor reconstruction from floorplan images with a deep learning approach, ISPRS Int. J. Geo Inf. (2020) 65, https://doi.org/10.3390/ijgi9020065.

[48] J. Seo, H. Park, S. Choo, Inference of drawing elements and space usage on architectural drawings using semantic segmentation, Appl. Sci. (2020) 1–14, https://doi.org/10.3390/app10207347.

[49] P. Ghannad, Y.C. Lee, Automated modular housing design using a module configuration algorithm and a coupled generative adversarial network (CoGAN), Autom. Constr. (2022) 104234, https://doi.org/10.1016/j.autcon.2022.104234.

[50] L. Chen, Q. Lu, X. Zhao, A semi-automatic image-based object recognition system for constructing as-is IFC BIM objects based on fuzzy-MAUT, Int. J. Constr. Manag. (2022) 51–65, https://doi.org/10.1080/15623599.2019.1615754.

[51] J. Rho, H.S. Lee, M. Park, Automated BIM generation using drawing recognition and line-text extraction, J. Asian Archit. Build. Eng. (2021) 747–759, https://doi.org/10.1080/13467581.2020.1806071.

[52] P. Ghannad, Y.C. Lee, J. Dimyadi, W. Solihin, Automated BIM data validation integrating open-standard schema with visual programming language, Adv. Eng. Inform. (2019) 14–28, https://doi.org/10.1016/j.aei.2019.01.006.

[53] K. Wang, M. Savva, A.X. Chang, D. Ritchie, Deep convolutional priors for indoor scene synthesis, ACM Trans. Graph. (2018) 1–14, https://doi.org/10.1145/3197517.3201362.

[54] W. Wu, L. Fan, L. Liu, P. Wonka, MIQP-based layout design for building interiors, Comput. Graph. Forum. (2018) 511–521, https://doi.org/10.1111/cgf.13380.

[55] N. Nauata, S. Hosseini, K.-H. Chang, H. Chu, C.-Y. Cheng, Y. Furukawa, House-GAN++: generative adversarial layout refinement network towards intelligent computational agent for professional architects, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 13632–13641. https://ennauata.github.io/houseganpp/page.html.

[56] A.B. Yenew, B.G. Assefa, E.G. Belay, HouseGanDi: a hybrid approach to strike a balance of sampling time and diversity in floorplan generation, IEEE Access (2024) 125235–125252, https://doi.org/10.1109/ACCESS.2024.3451406.

[57] 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. (2024) 105264, https://doi.org/10.1016/j.autcon.2023.105264.

[58] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects, IEEE Trans. Neural Netw. Learn. Syst. (2022) 6999–7019, https://doi.org/10.1109/TNNLS.2021.3084827.
[59] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM (2017) 84–90, https://doi.org/10.1145/3065386.
[60] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf (accessed June 10, 2025).
[61] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015, pp. 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.
[62] X. Liang, N. Yabuki, T. Fukuda, Generating a dataset of 3D BIM models of condominium units using deep learning, in: 10th International Conference on Innovative Production and Construction (IPC2024), Curtin University and East China Jiaotong University, Perth, Australia, 2024, pp. 71–80, in: https://icipc2024.com/wp-content/uploads/2024/07/ipc2024-conference-proceedings-2.pdf (accessed June 9, 2025).
[63] Revit API Docs. https://www.revitapidocs.com/, 2025 (accessed June 9, 2025).
[64] Revit Libraires. https://manage.autodesk.com/products/RVT?version=2026&platform=WIN64&language=EINT, 2025 (accessed June 9, 2025).
[65] F. Milletari, N. Navab, S.-A. Ahmadi, V-net: fully convolutional neural networks for volumetric medical image segmentation, in: 2016 Fourth International Conference on 3D Vision (3DV), IEEE, 2016, pp. 565–571, https://doi.org/10.1109/3DV.2016.79.
[66] A. Dai, A.X. Chang, M. Savva, M. Halber, T. Funkhouser, M. Nießner, ScanNet: Richly-annotated 3D reconstructions of indoor scenes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5828–5839. https://openaccess.thecvf.com/content_cvpr_2017/papers/Dai_ScanNet_Richly-Annotated_3D_CVPR_2017_paper.pdf (accessed June 10, 2025).
[67] Q. Xu, X. Sun, C.-Y. Wu, P. Wang, U. Neumann, Grid-GCN for fast and scalable point cloud learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 5661–5670. https://openaccess.thecvf.com/content_CVPR_2020/papers/Xu_Grid-GCN_for_Fast_and_Scalable_Point_Cloud_Learning_CVPR_2020_paper.pdf (accessed June 10, 2025).
[68] S. Song, F. Yu, A. Zeng, A.X. Chang, M. Savva, T. Funkhouser, Semantic scene completion from a single depth image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1746–1754. https://openaccess.thecvf.com/content_cvpr_2017/papers/Song_Semantic_Scene_Completion_CVPR_2017_paper.pdf (accessed August 11, 2025).