

Received 12 September 2024, accepted 30 September 2024, date of publication 3 October 2024, date of current version 16 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3472610

METHODS

Hash-Based Method for Generating Building Information Models From 2D CAD Drawings

KYUNGJIN KANG¹, ILKYEUN RA², (Member, IEEE), AND SANGO PARK¹, (Member, IEEE)

¹School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea

²Department of Computer Science and Engineering, University of Colorado Denver, Denver, CO 80204, USA

Corresponding author: Sangoh Park (sopark@cau.ac.kr)

This work was supported in part by the Chung-Ang University Research Grant in 2023 and in part by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (HRD Program for Industrial Innovation) under Grant P0020632.

ABSTRACT A building information model (BIM) encompasses the essential information required for a building's entire lifecycle, based on a 3D model. However, because building design begins with spatial planning, the initial design is based on a 2D plane. Furthermore, introducing BIM to general buildings is not effective in terms of modeling because of the design cost and the inherent 2D structure of older buildings. Consequently, research conducted over the past few years aimed to automatically generate 3D models from 2D drawings; however, the performance is limited in terms of data structure and methodology. Therefore, this study proposes a method that generates 3D information necessary for generating BIM models from 2D drawings using 2D hash-based data structures and planar graph analysis to resolve the above problem. Time and spatial complexities for planar graph generation from drawings consisting of n line segments among the proposed methods are $O(kn)$ and $O(n)$, respectively, where n is the number of the lines in drawing, k satisfies $E(p) \leq k \leq n$, and $E(p)$ denotes the average number of intersections in each hash cell. By applying it to 2D drawing data used in actual architectures, a 3D model could be developed with approximately 2.9% effort compared to the entire 2D drawing being modeled by a worker. This enables designers unfamiliar with BIM environments to conveniently obtain IFC files containing BIM models and then utilize BIM software to harness 3D information.

INDEX TERMS Building information model, CAD drawing analysis, geometrical algorithm, polygon detection.

I. INTRODUCTION

The building information model (BIM) refers to the modeling and management of information generated in the entire lifecycle of a building based on 3D, including planning, designing, constructing, inspecting, completing, facility operations, remodeling, to eventual demolition [1]. The design plan begins with a movement line plan or plane, and the schematic design is formally calculated by reflecting the arrangement of various facilities considering the movement line based on the 2D plan [2]. Despite buildings are constructed in 3D, drawings are typically rendered in 2D, leading to various errors during the construction phase. Introducing BIM at the design phase allows for early detection of these

issues by representing drawings in 3D [3]. Furthermore, 3D designs based on BIM offer numerous benefits, such as efficiently responding to emergencies like fires by integrating building spatial information with Geographic Information Systems (GIS) [4], and enhancing residents' well-being and productivity by incorporating systems like the Internet of Things (IoT) [5].

Research on analyzing 2D drawings is necessary for the following reasons. First, although the advent of BIM technology traces back to the early 2000s, recent data indicates that over 25% of architectural engineering firms continue to bid on only projects grounded in 2D CAD drawings [6]. Modeling accounts for roughly 5% of the construction period [7]. Buildings with minimal influence on 3D design elements, such as general apartments, houses, and shopping malls, are not notably impacted by this cost.

The associate editor coordinating the review of this manuscript and approving it for publication was Claudio Zunino.

In other words, even if BIM technology becomes more common, certain projects can be conducted based on 2D CAD. Second, the life expectancy of buildings built with reinforced concrete methods, which are mainly constructed for buildings with three or more floors, is approximately 40 years and is known to be usable for more than 70 years, depending on maintenance [8]. This means that even if BIM technology becomes common within the next 10 years, it will be difficult to apply BIM technology to buildings built in the past 40-70 years. Alternatively, research is required to analyze 2D drawings to apply BIM technology to old buildings. Obtaining 3D information for BIM from 2D drawings includes removing elements unrelated to buildings and classifying layers by members, such as columns, beams, floors, and walls. However, the method of modifying the drawing still requires manual work, and methods using member size can be omitted if the size of the shape on the list of members and the size of the shape on the floor plan are different.

Research endeavors aimed at extracting 3D BIM data from 2D drawings proposed in recent years such as to modify some drawing shapes, to recognize patterns from the shape of polygons, and to match the symbol of members to the rectangular described in the drawings. However, the method of modifying the drawing still takes manual work time, and some members are expected to be omitted if the size of the member's schedule and the size of the figure on the floor plan are different in methods using the member size.

This study proposes a method for quickly calculating the shortest distance between coordinates through a 2D-based hash data structure and extracting the symbol of structural members and nearby polygons on the drawing to solve the problem of manual and scheduling mismatches. Time and space complexities of the proposed graph generation are $O(kn)$ and $O(n)$, respectively, where n is the number of lines in drawing, $E(p) \leq k \leq n$, $E(p)$ is the average number of intersection points included in each cell. In addition, applying the proposed method to drawings used in actual construction allowed the 3D model to be generated with an effort of 2.9% effort compared to the entire previous drawing being modeled by the operator.

The remainder of this paper is organized as follows. Section II examines studies that obtain the 3D information necessary for BIM from 2D CAD drawings and explains the method proposed in Section III. Section IV tests the performance of the proposed method when applied to real-world drawings and addresses the problem of accuracy through real-world modeling cases. Finally, Section V presents conclusions and future research tasks.

II. RELATED WORKS

This section outlines existing research on topics such as automating modifications in 2D CAD drawings, extracting and classifying text patterns, and matching squares and symbols. A general construction member list consists of a simple table form, as introduced in [9], [10], and [11],

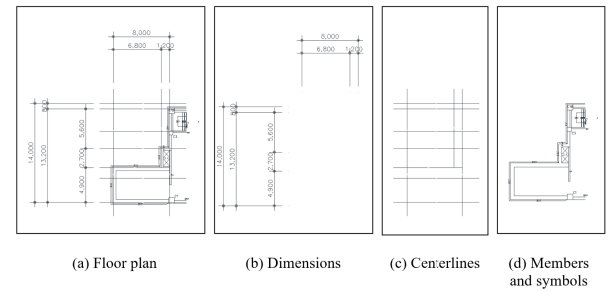


FIGURE 1. Composition of floor plan.

The floor plan (a) consists of shapes corresponding to the dimension line (b), center line (c), and member and symbols (d), as shown in Fig. 1. The methods introduced in this section analyze the shapes of the construction member in Fig. 1 and then generate data necessary for 3D model information using the thickness of girders, beams, and slabs, and height of the columns and walls on the member lists.

A. A SEMI-AUTOMATIC BIM MODEL GENERATION METHOD

Because the most important factor in entering a BIM model from a 2D drawing is the new input of 3D information based on the 2D drawing, a semi-automatic BIM model generation method was proposed in [9] focused on quickly processing this input task. In the proposed semiautomatic 3D model generation method, a portion of the original drawing (a) on the left is modified, as illustrated in Fig. 2, and a 3D model is then generated using the analysis algorithm implemented in the design software.

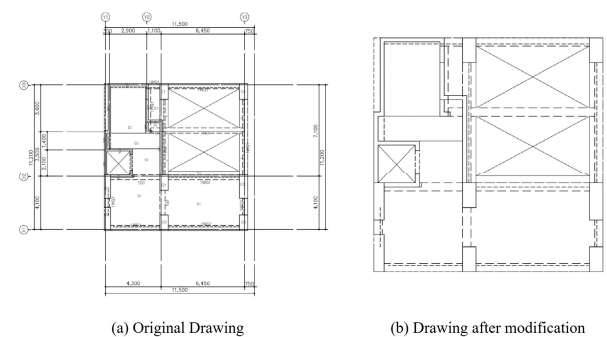


FIGURE 2. Drawing modification.

Drawing modification includes three subworks: clean drawing, layering, and redrawing. In the drawing arrangement stage, the worker removes all elements that are unrelated to the building, such as hatches and dimensions, while in the stratification stage, the worker separates the shapes of the structural elements, such as the pillar walls that make up the building layers. Finally, in the rewrite operation, the operator combines line segments for members that are difficult for the analysis algorithms to implement internally in the design software to automatically extract or add polygons, if necessary. In this study, the basic 3D

model required for BIM is created through 2D geometric information processing, such as drawing load, various shapes, dimensions, and layer-by-layer shape analysis, with existing algorithms implemented in the software.

By working with specific building drawings through the proposed method, it required approximately an hour to modify the drawings and approximately 15 minutes to generate the BIM model through the implemented algorithm; 12 matching errors out of 1756 building components occurred, and it required an additional hour to correct them. It requires approximately two days for a similar model to be processed manually using the BIM software. The method proposed shows significantly improved results compared to manual processing in existing BIM software. However, it requires approximately 15 minutes to create an automated BIM model, indicating that the existing method has limitations.

B. A PATTERN EXTRACTION AND TEXT CLASSIFICATION METHOD

To address issues such as the conversion time from 2D design drawings to 3D taking up approximately 5% of the construction period, a framework automating the creation of 3D objects by leveraging the combination of lines and text in drawings was developed in [10]. Within this automated framework, 3D objects are primarily generated through two procedures: text inference and pattern recognition based on line segment intersections.

To address problems such as the time required to convert 2D design drawings into 3D drawings, which is approximately 5% of the construction period, a framework was developed in [10] to automate 3D object creation using a combination of lines and text in drawings. In the developed automation framework, 3D objects are generated primarily by inferring text and recognizing patterns based on line-intersection information. The text inference procedure classifies the types of members in the membership list and the text indicating their absence and then deduces the types of members from the text described in the floor plan through Bayes filter training. The pattern recognition procedure classifies drawing recognition, line segments, and text using existing file processing and geometric methods and then extracts the intersections of each line segment. Finally, a 3D object is created using the height information of the cross-section by recognizing a pattern of nodes close to the text among the intersection points extracted from the floor plan to infer the planar position of the member. For example, in Fig. 3, the shape of C1 is inferred using the pattern for the left upper point P_{lt} , the middle point P_{cl} , and the left lower point P_{rb} of the floor plan (a), and the height information is obtained using the middle point P_{ct} and middle point P_{cb} of the cross-section (b).

The performance of the Bayesian filter and the error between the model formation of the generated 3D object and the position on the 2D drawing were tested; the performance of the bay filter was 97% accurate, and the position of all

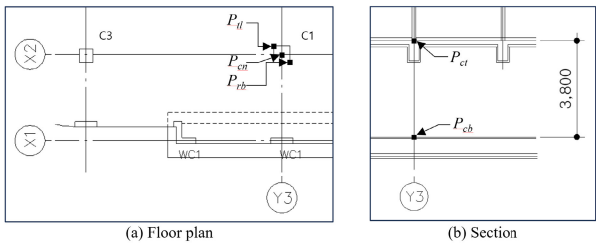


FIGURE 3. Drawing modification.

TABLE 1. Structural member abbreviation.

Member	Abbreviation
Columns	C
Girders	G
Beams	$(([0-9])(\sim -))?([0-9])?([A-Z])?$
Slabs	S
Walls	W

members showed a consistent error along the X-axis. This error resulted in certain errors in reference point processing. The proposed method is primarily divided into inference methods for Bayes filter-based text classification and the recognition of patterns of appearance, which are considered a de facto standard. Sign marks in the Anglo-American region, including Korea, do not deviate significantly from the regular expression in Table 1.

It is challenging to consider this approach effective, particularly when dealing with drawings from countries using unique notations. These can be more intricate than adjusting regular expressions and might require retraining Bayes filters developed in other regions. In addition, if a member-shaped pattern is recognized based on a predetermined size, it is difficult to ensure the accuracy of the results if the size of the pillar, thickness of the beam, and wall differ from the specified size and shape.

C. A SYMBOL TO RECTANGULAR MATCHING METHOD

Numerous countries still maintain building permits and public work environments in 2D form, making productivity enhancement challenging. Additionally, there's a lack of regulations, such as the development of BIM standards and defining corporate roles [7]. Accordingly, a system that automatically generates 3D information from 2D drawings and then outputs it in a BIM standard file format was implemented in [11]. The implemented system only processes the division of drawings such that the system user stores them in one file per drawing, and the system classifies and analyzes the files and outputs files in an IFC format compatible with the BIM software. The drawing is largely divided into a list in the form of a member and a floor plan in the form of shapes and text. The rows and columns of the table are analyzed based on whether the entry direction of the member information on the table is horizontal or vertical. In plan analysis, shapes are then classified based on squares, and a set of (shape, symbol) pairs is generated based on the sign and the polygon closest to the sign. When these two tasks are

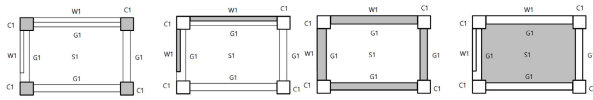


FIGURE 4. Rectangular matching method.

completed, (layer, symbol) pairs and (shape, symbol) pairs are matched in the order of columns (C1), walls (W1), and beams (G1), and after the matching is completed, the space surrounded by columns, walls, and beams naturally becomes a slab (S1) like Fig. 4. Finally, the 3D data construction outputs BIM software-compatible files in the IFC format using the IfcOpenShell library.

The performance evaluation was conducted for two buildings with a total floor area of less than $10,000m^2$. The evaluation of each building required less than 1 min, from the entry of the file classified by the system user to the printing of the IFC. The accuracy of the expected and calculated quantities was compared, and the results are listed in Table 2 below. The reason for the omission of the quantity was that either the code not in the list was written on the floor plan, or the code notation was omitted from the floor plan.

TABLE 2. Accuracy for member detection.

Test Drawing	Columns	Girder & Beams	Slabs	Walls
House	97.54%	99.21%	100%	99.47%
Parking facility	100%	98.73%	100%	100%

The implemented system expected to address the problems of high performance and size mismatch through a simple polygon search algorithm and simple matching of signs obtained from the member lists. However, it will be difficult to process if part of the building is diagonal. In addition, when the shapes and dimensions of the members listed in the list differ from those written on the floor plan, the treatment method is not presented.

III. PROPOSED HASH-BASED METHOD

The proposed method has a hash data structure that uses two-dimensional coordinates as keys, a method for generating a graph from each line of the drawing to obtain a polygon for the sign of the drawing member, and a method for extracting a recent polygon. Hash-based data structures divide planes into cells based on coordinates and store and manage geometry-related data to improve the search performance for elements of shapes or generated graphs closest to specific locations. The nearest polygon extraction method converts the shape of the drawing into a graph, extracts the polygon closest to or containing a specific location, and then applies the thickness of the member or the height of the plane to generate 3D modeling information.

A. 2D-HASH BASED DATA STRUCTURE

The proposed 2D hash uses geometric coordinates as keys. To conceptually explain the data structure using a set of points, when points are given in a two-dimensional coordinate

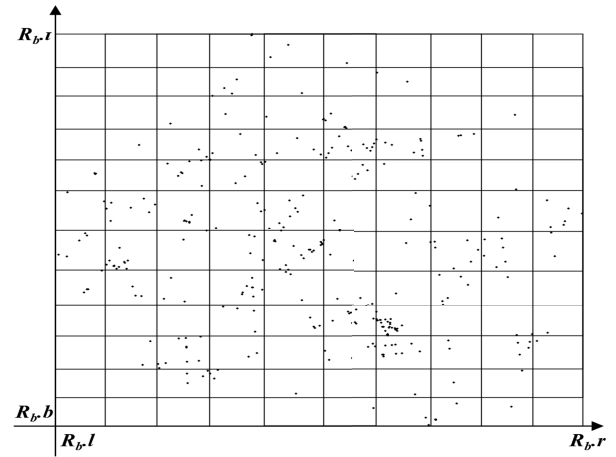


FIGURE 5. Cell space for point data.

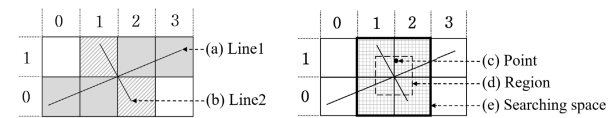


FIGURE 6. Example for rasterization and search space.

space, as shown in Fig. 5, the coordinate space is divided into arbitrary cells and maintained. Alternatively, the hash table used in this study consists of a two-dimensional array. Furthermore, by normalizing the space in which the entire drawing shapes is stored with respect to the origin, an edge square R_b , consisting of four pairs (l, r, b, t) is obtained, where four pairs mean left ($R_b.l$), right ($R_b.r$), bottom ($R_b.b$), and top ($R_b.t$) of the square, respectively, and after normalization, the values of l and b become zero.

In addition, if the horizontal and vertical coordinates of R_b can be divided into the same number, and the number is B , the size of the horizontal and vertical coordinates of R_b may be different. Therefore, B is defined as a pair (w, h), where w and h refer to the width and height of R_b , respectively, and can be calculated by dividing the width and height of R_b by C . That is, the method in which the coordinates are stored is one in which points mapped to the same pixel are stored in one hash cell when the points on the vector are lettered. For lines, faces, circles, and arcs, the horizontal and vertical indices of the cells to be stored can be obtained by processing them using a rasterization algorithm. However, as shown in Fig. 6, the stored cells may not overlap. Therefore, they should be stored in all cells sharing vertices, and all adjacent cells should be examined because there is no guarantee that the shape closest to the point exists in the same cell.

In Fig. 6, when (a) segment 1 is rasterized, it corresponds to cells (0, 0), (0, 1), (1, 2), and (1, 3), and (b) segment 2 corresponds to cells (1, 1) and (0, 2); therefore, additional storage is required. In addition, if only the cell to which point (c) belongs is searched, search for adjacent shapes based on location, inaccurate results can be obtained; therefore, we must examine the three adjacent cells based on cell division into four.

B. GENERATING GRAPH FROM SHAPES

The graph we aim to create is planar graph, as introduced in [12], and a method for generating a graph from a line segment set can be created by obtaining the intersection point of a line segment with a shapes that constitutes a drawing from the line segment set, described in [13]. Adjacent polygons can be extracted using the location of this graph and sign, as described in detail in Section III-C. As illustrated in Fig. 7, line segment decomposition is the operation of dividing each line into four line segments based on the intersection point at which the two lines intersect, which is essential for generating a line segment graph.

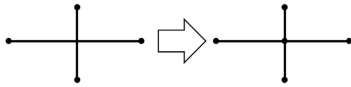


FIGURE 7. Example for rasterization and search space.

This work is decomposed through the line separation algorithm below for line segment l , line segment sets L , and hash H belonging to all cells to which l belongs.

Algorithm 1 Line Separation

```

1: procedure LineSeparation( $l, L, H$ )
2:    $P \leftarrow \emptyset$ 
3:   while  $0 < |L|$  do
4:      $l_0 \leftarrow$  Pop first element of  $L$ 
5:      $p \leftarrow$  Detect intersection between  $l$  and  $l_0$ 
6:     if  $p \neq \emptyset$  then
7:       Delete  $l_0$  from  $H$ 
8:       Insert (left of  $l_0, p$ ) into  $H$ 
9:       Insert ( $p$ , right of  $l_0$ ) into  $H$ 
10:      Insert  $p$  into  $P$ 
11:    end if
12:  end while
13:  Delete  $l$  from  $H$ 
14:  for Each  $p_i \in P$ , where  $i \leftarrow 1$  to  $|P| - 1$  do
15:    Insert ( $p_{i-1}, p_i$ ) into  $H$ 
16:  end for
17: end procedure

```

The algorithm decomposes another line crossing line l (Steps 1-10), decomposes l (Steps 11-20), and adjusts hash H using the decomposition results (Steps 5-7, Steps 19-20). In the algorithm, repetition increases with the size of L and the number of intersections p above line segment l , where L is the number of lines within distance l and B is the width of the 2D hash calculated in paragraph 3.1.1. Because the number of intersections p is consistently less than or equal to the size of L , it is determined by the resolution R of hash.

If R is 1, all line segments belong to one cell; therefore, it becomes equal to the number of line segments n , and as R increases, it becomes closer to the number p belonging to the cell. That is, the number of comparisons k required to obtain all intersections for a line segment l satisfies $p \leq k \leq n$, such that when the number of lines is n , k is $E(p) \leq k \leq n$, and

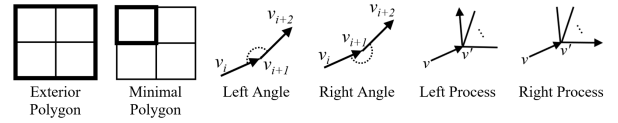


FIGURE 8. Visualization of the 6 definition.

$E(p)$ is the average number of intersections included in each cell. Method for generating a line segment graph G is used to select any line l from the hash and then enter three pairs (p_1, l, p_2) in the form of (point, edge, and vertex) into graph G using both endpoints p_1 and p_2 of l . Graph G can be created by verifying the existence of each vertex among the three pairs (point, edge, and vertex) entered into the graph data structure; if it is already entered, add the edge and adjacent vertices to the two vertices, respectively. If the data structure of graph G is also maintained as a 2D hash, only vertices are added; thus, it is the same as the time complexity when the vertex is input. However, a separate space complexity of $O(n)$ is required. That is, the time and space complexities of the proposed graph generation method are $O(kn)$ and $O(n)$, respectively.

C. POLYGON DETECTION FROM THE GRAPH

In this study, for a polygon that can be generated in a plane graph, a polygon with zero vertices and edges outside is defined as an outer polygon, and a polygon with zero vertices and edges inside is defined as the minimum polygon. Furthermore, by visiting adjacent vertices belonging to any polygon in the order of v_i, v_{i+1} , and v_{i+2} , the angle formed by both edges can be measured around v_{i+1} , where the angle measured from the left is defined as the left angle, and the opposite side is defined as the right angle. In the sequence of v to v' , the procedure of choosing the smallest neighboring vertex, excluding v' , is termed as selecting the leftmost adjacent vertex. Conversely, the process of picking the smallest vertex adjacent to v' is termed as the right progression. The following two coronaries can be intuitively induced from the left and right progression. For left progression, if v is reached through progression, the direction is bound to be counterclockwise; the right progression is also bound to be clockwise.

From the left process and the right process, below two coronaries can be intuitively derived: if a visit order reaches arbitrary vertex v through a left process, the direction must be counterclockwise. Similarly, the right process can only imply in the clockwise direction.

Corollary 1: Visit order for a minimal polygon P is counterclockwise if and only if P is generated by the left process for an edge that is not in the outer polygon.

Corollary 2: The visit order for a minimal polygon P is clockwise if and only if P is generated by the right process for an edge that is not in the outer polygon.

Corollary 3: If on any edge e that does not belong to an outer polygon, if a polygon is created through left-hand progression and right-hand progression, it is the smallest polygon that shares e .

Theorem 1: If a polygon is created through left and right progression for any edge e belonging to an outer polygon, then an outer polygon and a minimum polygon can be obtained, respectively.

Proof Assuming clockwise rotation for an outer polygon in an arbitrary graph, the right angle at all vertices of the outer polygon is minimal; assuming counterclockwise rotation, the left angle at all vertices of the outer polygon is minimal. In addition, assuming a clockwise rotation for all minimum polygons in Corollaries 1 and 2, the left angle at all vertices is the minimum, and assuming a counterclockwise rotation, the right angle at all vertices is the minimum. Alternatively, the right progression produces an outer polygon, the left progression produces a minimum polygon, the right progression produces a minimum polygon, and the left progression produces an outer polygon. Thus, regardless of the direction of rotation, the left and right progressions symmetrically produced the minimum polygon and outer polygon containing the corresponding edge, respectively.

From Corollaries 3 and 1, when a left and right progression is performed on any edge e closest to a point if the point exists outside the outer polygon, an outer polygon and a minimum polygon can be obtained, and the point is located outside the two. In addition, if a point exists inside an outer polygon, at least two outer polygons and at least one polygon can be obtained depending on the position; the point is one of the two polygons in the former and both polygons in the latter. Apart from this consideration, the sign of each member and the location of the polygon are generally described outside the polygon. For floor plates, they are characterized as being described inside the polygon.

Therefore, it is feasible to create two polygons based on the nearest edge for external codes and then extract polygons corresponding to members by selecting polygons that do not belong to the location of the code and those that belong to the protection position. Meanwhile, for one edge, that is, the time complexity for polygon extraction for members increases linearly with the number of symbols.

IV. EXPERIMENT AND CASE STUDY

This section examines the performance changes according to the increase in line segments when applied to actual drawings for intersection search and polygon extraction methods for graph generation and examines the necessary supplements when applied at this time through actual drawing examples.

A. LINE CROSS PERFORMANCE

The intersection algorithm identifies the Bentley-Ottmann algorithm [14], which is primarily used for intersection searches, and the performance difference according to the increase in the line fractions for the same set. The set of line segments for comparison was created at random locations, increasing the number of line segments of length $l/20$ from 10 to 1,000 in a space of size $l \times l$. At this time, the number of cells C of the resolution hash was set to 256, and at 1,000, the input sample demonstrated approximately

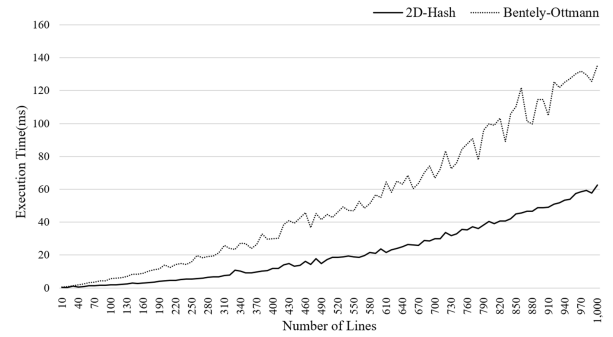


FIGURE 9. Execution time for detecting cross points to set of lines.

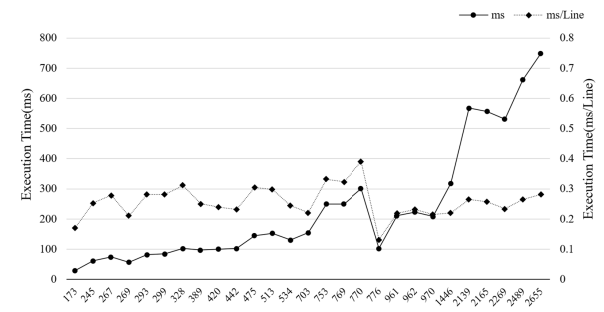


FIGURE 10. Execution time for detecting polygons to number of lines.

50% processing time compared with the existing algorithm, as illustrated in Fig. 9.

B. POLYGON DETECTION

The experimental details of the polygon extraction method are as follows: we obtained a set of line segments from 35-floor plans in four existing buildings, performed the polygon generation method 30 times for 35 sets of line segments, and measured the average performance time. The resolution R was determined based on the condition that the greater of $B.width$ and $B.height$ for the cell's upper and lower sizes, denoted as B , should not surpass 7m. This is because, due to the nature of the architectural drawings, intersections occur at the points where members are in contact and the effective column spacing is approximately 7m, such that the point at which the members are in contact (e.g., columns, beams, and beams) is also likely to be approximately 7m. That is, it may be ensured that a minimum of two intersection points are included in one cell. By applying the proposed polygon extraction method to 30 building drawings with 173 to 2655 lines, the number and execution time of lines included in the drawings were as shown in Figure. 10, and the processing time increased linearly, as mentioned in Section III-C.

C. CASE STUDY

The results in the previous section examined and analyzed the processing time, accuracy, and location of members when extracting pillars, beams, floor plates, and walls of buildings using polygons for a warehouse facility design with five floors above the ground and a total floor area

of approximately $5,849m^2$. For an accuracy comparison, the exact quantity of each member prepared on the floor plan was determined manually. The implemented test code pre-processed each drawing such that only one drawing was stored in a single file and then proceeded in three stages: file load and all polygon generation, member extraction and remaining polygon post-processing, and component modeling and confirmation polygon output. The first stage implemented based on .dwg file specification in [15], second stage implemented as presented in this paper, and last stage implemented by using IFCopenshell library. The system used for component extraction and polygonal verification was conducted on a desktop with a core i-7 12700 CPU, 64 GB of main memory, and Microsoft Windows 11 64-bit operating system. For the calculated quantity of members, there was no difference in the calculated quantity except for the wall related to post-processing, because if there is a sign and there is an unmatched polygon, it must be calculated. The member position was directly verified by saving the line segment of the input drawing and extracted member in the drawing file and visually verifying the difference between the plan view and extracted member drawing, as shown in Table 3.

TABLE 3. Accuracy for member position.

Floor Plan	Columns		Girders		Walls		Slabs	
	Detect	Miss	Detect	Miss	Detect	Miss	Detect	Miss
1 st Floor	21	3	-	-	51	0	-	-
2 nd Floor	21	3	96	2	53	0	68	4
3 rd Floor	21	1	96	1	64	0	68	5
4 th Floor	21	1	96	1	64	0	68	5
5 th Floor	21	2	96	1	62	0	68	6
Roof	-	-	96	1	62	0	67	1
Total	105	10	480	6	356	0	339	21
Error	9.53%		1.25%		0%		6.20%	

After reviewing the drawings, it was determined that the error was largely caused by ambiguity (a) and design error (b), as shown in Fig. 11. Ambiguity occurs when one edge overlaps two different members, or when the edge closest to the position of the sign is another, and design errors occur when the drawing is inaccurate. This can be addressed by adjusting the position of the code or partially modifying the line segment of the drawing. Thus, assuming that the cost of modifying the drawing for each member is the same, the same 3D result can be obtained with approximately 2.9% effort compared to manually modeling the entire 2D drawing. In addition, if the same sign is written several times on the list or the floor plan layers are marked overlap (e.g., 1-3S1, 2-7S1), there may be a problem in processing the member; however, in that case, it is not feasible to process it simply by modifying the drawing and must be modified through consultation between the structural and architectural.

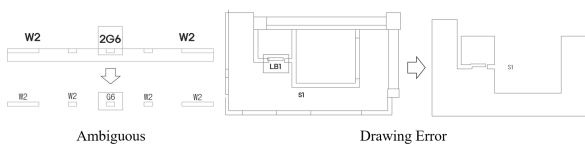


FIGURE 11. Graphical case of error source.

TABLE 4. General case of error source.

Source	Cases
design error	A symbol occurs on plan without writing on the member lists. Missing symbols on the floor plan. The size of the list is different from the size of the floor plan. The floor plan is not drawn correctly.
Ambiguous	The sign is far away from the polygon of the member in floor plan. The same sign is written multiple times in the list.
Visual limitations	Endpoints of line segments are off. Unnecessary line segments are included.

TABLE 5. Reconstructing BIM from 2D structural drawings for existing buildings.

Category	TP	FP	FN	Precision	Recall	F1 score
Column	963	57	83	88.45%	82.34%	85.29%
Beam	349	37	27	90.41%	92.82%	91.60%
Beam _v	402	29	38	93.27%	91.36%	92.31%
Beam _s	31	4	3	89.19%	91.67%	90.41%

The detail reason why the accuracy of member extraction does not reach 100% despite processing drawings based on straightforward algorithms and rules is partly due to the member extraction rules, as discussed in Section 4.4.1, but also due to design errors. In this section, the main reasons for inaccuracies in drawing processing can be categorised into design errors, processing ambiguities, and limitations of the proposed method, as shown in Table 4.

The errors mentioned above are challenging for designers and constructors to identify solely from the drawings and are typically discovered during construction. These issues can be addressed by first creating a BIM model and then utilizing additional information, such as the zone ontology introduced in [16], to validate the spatial organization.

In addition to the method proposed in this paper, which generates 3D data from 2D drawing files, research is also being conducted on generating 3D data by analyzing raster images of drawings using artificial intelligence techniques [17]. A recent research [18] presents a method for generating BIM data in IFC format by detecting columns and beams (horizontal, vertical, diagonal) through character analysis using existing Optical Character Recognition techniques based on CNNs. This approach involves recognizing the type of member and mapping the detected characters to bounding boxes within the image. Although a direct comparison is not feasible due to differences in input data (vector data vs. raster images), the performance of this methodology is summarized in the table 5. Statistical abbreviations used in the evaluation include True Positive (TP), False Positive (FP), and False Negative (FN).

V. CONCLUSION AND FUTURE WORKS

In this study, we propose a method for segmenting vector-based drawing data using a 2D hash data structure,

generating a line segment graph, and extracting polygons from the generated graph. The time and space complexity for the proposed graph generation are $O(kn)$ and $O(n)$, respectively, and $E(p) \leq k \leq n$, where $E(p)$ is the average number of intersection points included in each cell. Furthermore, we assume that the cost of modifying drawings for each member is the same as applying the proposed method to general 2D CAD drawing data created and utilized in the construction industry. The same 3D result can be obtained with approximately 2.9% effort compared to manually modeling the entire 2D drawing. This allows designers who have difficulty handling BIM software to obtain files in IFC format in which the BIM model is stored conveniently, and then utilize the BIM software to utilize the contents. However, by case analysis, the accuracy of component extraction was found to vary depending on the quality level of the drawing required by the method proposed in this study, and there are clear design errors, such as missing signs or incorrect size or thickness of members. In particular, figures unrelated to space, such as doors, furniture, facilities, and interior decorations, should be deleted before analysis; however, they are written on a floor separate from the relevant floor and are difficult to separate. Alternatively, research is required to analyze drawings at the human level, which are difficult to address using straight algorithms, and artificial intelligence-based research, such as machine learning, is required.

REFERENCES

- [1] S. Azhar, M. Khalfan, and T. Maqsood, "Building information modeling (BIM): Now and beyond," *Australasia J. Construct. Econ. Building*, vol. 12, no. 4, pp. 15–28, 2012.
- [2] P. Landau. *Architectural Design Process: A Quick Guide*. ProjectManager.com, Inc. Accessed: Jul. 20, 2022. [Online]. Available: <https://www.projectmanager.com/blog/architectural-design-process>
- [3] A. Kazaz, T. Acikara, S. Ulubeyli, and H. Koyun, "Detection of architectural drawings errors in 3 dimension," *Procedia Eng.*, vol. 196, pp. 1018–1025, Jun. 2017.
- [4] H. Xia, Z. Liu, M. Efremochkina, X. Liu, and C. Lin, "Study on city digital twin technologies for sustainable smart city design: A review and bibliometric analysis of geographic information system and building information modeling integration," *Sustain. Cities Soc.*, vol. 84, Sep. 2022, Art. no. 104009.
- [5] M. Valinejadshoubi, O. Moselhi, A. Bagchi, and A. Salem, "Development of an IoT and BIM-based automated alert system for thermal comfort monitoring in buildings," *Sustain. Cities Soc.*, vol. 66, Mar. 2021, Art. no. 102602.
- [6] JBKnowledge. (2021). *Construction Technology Report*. [Online]. Available: <https://contechreport.com/wp-content/uploads/2021/12/2021-JBKnowledge-Construction-Technology-Report.pdf>
- [7] C. Botton, S. Kubicki, and G. Halin. *4D/BIM Simulation for Preconstruction and Construction Scheduling. Multiple Levels of Development Within a Single Case Study*. Diamond Congress Ltd. Accessed: Jul. 10, 2020. [Online]. Available: <https://hal.science/hal-02890081>
- [8] M. Liang, K. Wang, and C. Liang, "Service life prediction of reinforced concrete structures," *Cement Concrete Res.*, vol. 29, no. 9, pp. 1411–1418, 1999.
- [9] P. Janssen, K. W. Chen, and A. Mohanty, "Automated generation of BIM models," in *Proc. eCAADe*, vol. 2, 2016, pp. 583–590.
- [10] J. Rho, H.-S. Lee, and M. Park, "Automated BIM generation using drawing recognition and line-text extraction," *J. Asian Archit. Building Eng.*, vol. 20, no. 6, pp. 747–759, Nov. 2021.
- [11] Y. Byun and B.-S. Sohn, "ABGS: A system for the automatic generation of building information models from two-dimensional CAD drawings," *Sustainability*, vol. 12, no. 17, p. 6713, Aug. 2020.
- [12] M. M. Syslo, "An efficient cycle vector space algorithm for listing all cycles of a planar graph," *SIAM J. Comput.*, vol. 10, no. 4, pp. 797–808, Nov. 1981.
- [13] A. Ferreira, M. Fonseca, and J. Jorge, "Polygon detection from a set of lines," in *Proc. Encontro Portugues de Computacao Grafica*, 2003, pp. 159–162.
- [14] J. L. Bentley and T. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Trans. Comput.*, vol. C-28, no. 29, pp. 643–647, Sep. 1978.
- [15] *Open Design Specification For. DWG Files, Version 5.4.1*, Open Des. Alliance, Scottsdale, AZ, USA, 2018.
- [16] R. Ferrari, H. Dibowski, and S. Baldi, "A message passing algorithm for automatic synthesis of probabilistic fault detectors from building automation ontologies," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4184–4190, Jul. 2017.
- [17] P. N. Pizarro, N. Hirschfeld, I. Sipiran, and J. M. Saavedra, "Automatic floor plan analysis and recognition," *Autom. Construct.*, vol. 140, Aug. 2022, Art. no. 104348.
- [18] Y. Zhao, X. Deng, and H. Lai, "Reconstructing BIM from 2D structural drawings for existing buildings," *Autom. Construct.*, vol. 128, Aug. 2021, Art. no. 103750.



KYUNGJIN KANG received the B.S., M.S., and Ph.D. degrees from the School of Computer Science and Engineering, Chung-Ang University, in 2008, 2010, and 2023, respectively. He has been an Adjunct Professor with the Research Center, Chung-Ang University, since 2022. He founded and served as the CEO of Cyphicsoft, Inc., from 2010 to 2021. His research interests include computational geometry, building information modeling, and embedded systems.



ILKYEUN RA (Member, IEEE) received the B.S. and M.S. degrees in computer science from Sogang University, the M.S. degree in computer science from the University of Colorado Boulder, and the Ph.D. degree in computer and information science from Syracuse University, in 2001. He was a Research Staff Member with the LG Information and Communications (currently LG Telecom) Research Center. He joined the Department of Computer Science and Engineering, University of Colorado Denver, in 2001. His main research interests include computer networks, and developing adaptive distributed system software and high-speed communication system software to support high-performance distributed computing applications.



SANGOH PARK (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Computer Science and Engineering, Chung-Ang University, in 2005, 2007, and 2010, respectively. He was a Senior Researcher with the Global Science Experimental Data Hub Center, Korea Institute of Science and Technology Information, from 2012 to 2017, and a Research Professor with the School of Computer Science and Engineering. He has been a Professor with the School of Computer Science and Engineering, Chung-Ang University, since 2017. His research interests include high-performance computing, distributed systems, embedded systems, and cyber-physical systems.

...