Available online at www.sciencedirect.com

ScienceDirect

Engineering Applications of Artificial Intelligence 21 (2008) 1073–1079

Engineering Applications of ARTIFICIAL INTELLIGENCE

www.elsevier.com/locate/engappai

Graph theory-based approach for automatic recognition of CAD data

H.C. Huang[a], S.M. Lo[a,*], G.S. Zhi[b], R.K.K. Yuen

[a]*Department of Building and Construction, City University of Hong Kong, Tat Chee Avenue, Kowloon Tong, Hong Kong*
[b]*Xi'an Jiaotong University, Xi'an, PR China*

Received 24 February 2006; accepted 2 December 2007
Available online 6 March 2008

Abstract

CAD architectural plans basically contain original geometrical information of graphical primitives. However, in many applications, such as building's 3D reconstruction, auto-detecting errors of design, original CAD data are very hard to be directly utilized. It is always a time-consuming and exhaustive task to extract useful information like the coordinates of a line from CAD files. To make this task performed in more efficient way, a way to develop an automatic method to extract spatial information from architectural plans produced in the form of computer-drawn CAD drawings is proposed in this article. The aim of the proposed method is to provide automatic transformation of architectural drawings into the spatial and topological information of the enclosure in a building. To auto-understand the 'meaning' of the graphic elements in the drawings such as walls, doors and rooms, the approach employs algorithms in graph theory, which can identify every functional component in the enclosure and establish their connectivity relationships. The method has been implemented by using object-oriented C++ language and is found to be able to produce satisfying results.

Keywords: Graph theory; Information extraction; Computer-aided design

1. Introduction

Nowadays, computer technology has been widely applied in various engineering application fields. In mechanical/manufacturing and architectural fields, design of products or buildings depends more and more on computer-aided systems. In the past, a CAD system was relatively simply established and mainly acted as an interactive graphics platform. Recently, more intelligent tools have been added into the system or separately developed, providing the capability of automatically finishing some works such as building's 3D reconstruction, auto-detecting errors of design, plan evaluation, plan checking, etc. However, prior to using these tools, we usually need to carry out pre-process works, which require extensive preparation of input data. It is always a time consuming and exhaustive task, especially for complex setting. For example, the detection of errors from a complex building plan requires the extraction of extensive information about the configuration of every room and door.

Generally, a CAD drawing contains original geometrical information of graphical primitives. However, the original CAD data is very hard to be directly utilized. In traditional pre-process modules, input data relating to the CAD drawings are required to be manually extracted and stored in a file. In fact, the information represented by the data is naturally embraced in the CAD drawings. To make the pre-process task performed more efficiently and accurately, it is necessary to develop a method to automatically extract meaningful information from the original drawings.

The study on automatic capturing CAD drawing is closely related to the specific engineering application. The development of generic recognition of CAD plans originated from the mechanical/manufacturing field (Prabhu and Pande, 1999; Meeran and Taib, 1999). Related information about the attributes of the products is captured to provide the input for manufacturing/production process. In the recent decades, some works have been

*Corresponding author. Tel.: +852 27887683; fax: +852 27887612.
E-mail address: bcsmli@cityu.edu.hk (S.M. Lo).

0952-1976/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2007.12.001

done on the intelligent understanding (IU) of architectural plans with the aim of assisting building designers in architectural, structural or cost analysis.

Ah-Soon and Tombre (1997) and Ah-Soon (1997) and have performed architectural plan analysis on the basis of symbol detection, geometrical and spatial analysis. In their approach, graphic symbols are represented by a set of geometrical features constraints. These constraints are propagated through the network hierarchy. Similar to Ah-soon's constraint network, Messmer's (1996) network for exact and inexact graph matching also has shown a good performance in recognition of the architectural symbols. Lewis and Sequin (1998) and Bukowski and Sequin (1997) have presented a method called building model generator (BMG) to build 3D building models from 2D architectural plans form a smoke-spread prediction model, CFAST. The BMG uses room label as seed point to find interior space contours, and can provide topological correction and semantic enrichment architectural model for CFAST. The BMG have largely reduced the modeling time; however, it is a semi-automatic processor that requires users to point out the location of each vertex. Other models like building EXODUS (Owen et al., 1996), SIMULEX (Thompson and Marchant, 1995) and SGEM (Zhi et al., 2003) for evaluating the evacuation system of a building also have been tried to incorporate automatic function to capture CAD information from architectural plans. These models are not fully automatic and require pre-treatment of the CAD files.

This article further describes the study on the use of graph theory algorithms to analyze and capture the spatial information of the building layout from architectural plans. In a CAD file, the vertices and lines connecting them are first identified; and then a graph is established, in which the nodes denote vertices and arcs denote lines. On the basis of the graph theory, the topological relationship of the nodes is analyzed according to some rules. As the result, the spatial information of the enclosures (rooms, passages, stairs, obstacles, etc.) inside the building can be obtained for specific applications. The proposed method has been implemented in C++ and integrated into a building evacuation simulation model as the pre-process module.

2. Problem statement and difficulties

The goal of the study is to extract spatial information inside a building from architectural plans. In a CAD drawing, a floor plan can be decomposed into many enclosure components with openings or without opening, such as compartments, corridors, halls, stairwells, walls, columns, etc. The capturing process needs to identify all the enclosures, their types and connectivity relationships. Each enclosure has its boundary represented by a series of vertices. So the process should start with recognizing all the key vertices including endpoints and intersection points. After determining the boundaries of every enclosure, their relationships should be further recognized. On such basis, we attempt to conclude the type and function of each enclosure. The task of information reorganization can be defined as: "Given a floor plan, identify the boundary of each enclosure and decide its type and relations with other enclosures." For example, Fig. 1 presents a possible floor plan of a building consisting of four rooms, one passage and one column. After identifying all the vertices, enclosures A–F can be recognized. The doors, common vertices can be used to decide connectivity relationships of these enclosures; and then the type of each enclosure may be presumable. Table 1 lists the output of capturing. Details of this process are further described in Section 3 and 4.

To capture information from architectural plans, there are several difficulties that need to be considered (Zhi et al., 2003):

1. drafting errors in CAD files may cause misunderstanding;
2. useless information such as dimensions, text notes, and other technical information/specifications must be removed;
3. it is hard to locate the openings of a enclosure usually represented as the doors(single-leaf, double-leaf);

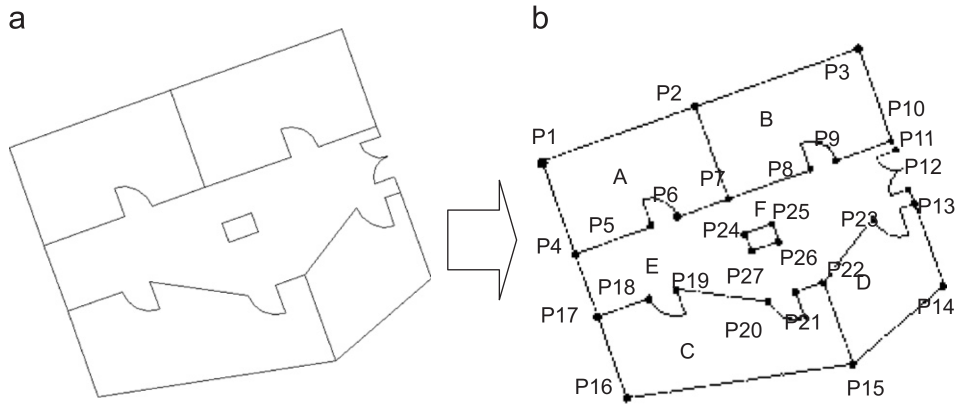

Fig. 1. Task of recognizing a floor plan (a) given floor plan and (b) result after processing.

Table 1
Output of capturing in Fig. 1

Function	Enclosures	Boundary	Opening(s)
A	P1, P2, P7, P6, P5, P4, P1,……	Connected to E via opening (P5, P6)	Room
B	P2, P3, P10, P9, P8, P7	Connected to E via opening (P8, P9)	Room
C	P17, P18, P19, P20, P21, P22, P15, P16	Connected to E via opening (P18, P19) and (P27, P21)	Room
D	P22, P23, P13, P14, P15	Connected to E via opening (P13, P23)	Room
E	P4, P5, P6, P7, P8, P9, P11, P12, P13, P23, P22, P21, P27, P19, P18, P17	Connected to outside via one opening (P11, P12)	Corridor
F	P24, P25, P26, P27	No opening	Obstacle

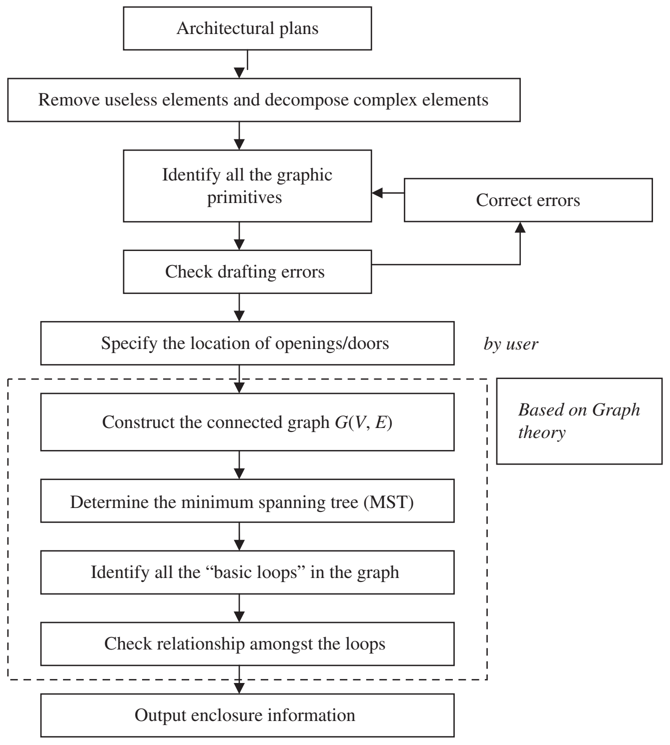
Architectural plans
Remove useless elements and decompose complex elements
Identify all the graphic primitives
Correct errors
Check drafting errors
Specify the location of openings/doors
by user
Construct the connected graph G(V, E)
Based on Graph theory
Determine the minimum spanning tree (MST)
Identify all the "basic loops" in the graph
Check relationship amongst the loops
Output enclosure information

Fig. 2. The spatial information extraction method based on graph theory.

4. it is hard to understand the exact type of a building space and
5. how to recognize the correct logical relationship amongst functional enclosures and obstacles?

3. The graph-based algorithm

To automatically extract architectural information from 2D plan drawings, a graph-based approach has been made (Zhi et al., 2003). This method can recognize information of all the enclosures. Fig. 2 describes the steps incorporated in the method. To simplify the problem, the following assumptions have been made:

1. the architectural plan is a single-line or double-line diagrams;
2. the whole plan is a combination of enclosure components;
3. a enclosure is a combination of loops and
4. except the obstacles or walls, each enclosure has at least one opening connected with other enclosures.

On the basis of above assumptions, an architectural plan can be firstly transformed in to a graph, and adopt graph theory to determine the solutions. Details are described as follows.

3.1. Identify all the graphic primitives

Before we perform the algorithm, pre-process operations are necessary to remove useless elements and correct the geometrical errors. The initial plan information is a set of symbols, lines (curves) and text. Some of these graphic elements have no relation with spatial information such as decorative symbols, annotated information, dimensions, specification notes, etc. These useless elements need to be deleted.

Because the architectural plans are drawn by designer in a computer, invisible drafting errors are impossible to be absolutely avoided. However, even a minor error may cause the wrong automatic reorganization of the plans (see Fig. 3). Fig. 3 shows the usual drafting errors including gaps or extensions between two vertices that should be the same. To correct these errors, we meld the vertices if the distance between them is less than a certain tolerance ξ.

In order to simplify the algorithm, we need to convert all the elements in the plan into basic graphic primitives such as lines with only two endpoints. Moreover, all the primitives may have common endpoints but must not intersect each other. Thus, complex elements such as "blocks" in CAD are decomposed into basic elements and the intersecting lines are broken down into segments at intersect points (see Fig. 4). This conversion is automatically performed by computer.

After above pre-process operations, all the graphic primitives including basic line segments and end vertices can be identified.

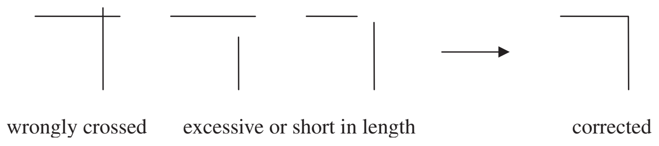

Fig. 3. Drafting error correction.

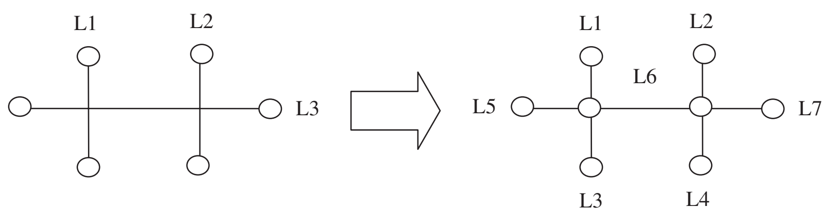

Fig. 4. Decompose intersecting lines into basic elements.

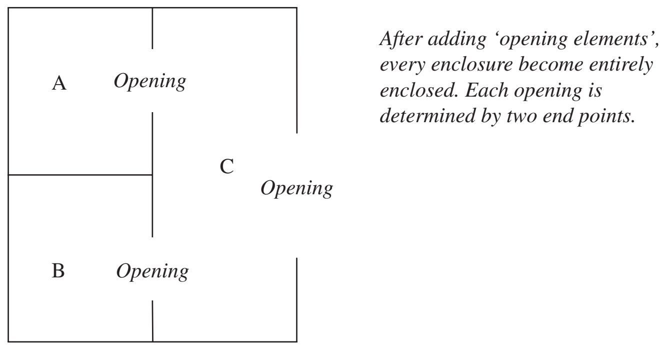
A
Opening
C
Opening
B
Opening
After adding 'opening elements', every enclosure become entirely enclosed. Each opening is determined by two end points.

Fig. 5. Specify all the openings.

3.2. Specify the openings

In a building, the interior space is divided into interconnected subspaces named enclosures in this paper. Each functional enclosure must be connected to other enclosures with at least one opening. The opening may be doorways or passages. When employing graph theory, identifying all the enclosures require that they are closed by the lines and openings. Hence, prior to conversion of an architectural plan into a graph, we should specify all the openings by adding virtual 'opening' entities to the plan (see Fig. 5).

3.3. Convert an architectural plan into a graph

After performing above operations, we can construct a connected graph $G = (V, E)$. V denotes the node set. A node represents a key point in the plan, with the attributes including coordinates (x, y) indgree, outdgree and adjacent edges. E denotes the edge set. An edge generally means a wall or an opening in the plan. An edge's attributes include the adjacent two nodes, the type of the edge. The number of nodes in a graph G is its *order*, and the number of edges is its size. For example, the floor plan in Fig. 1 can be considered as two connected graphs showed in Fig. 6.

In computer, the graph can be described as several matrix representations. The vertex-edge incidence matrix is adopted in this article. An incidence matrix $A_\alpha = [a_{ij}]$ of graph G is defined as

$$\begin{cases} a_{ij} = 1 & \text{when edge } e_j \text{ is incidence on node } v_i, \\ a_{ij} = 0 & \text{otherwise.} \end{cases}$$

As the incidence matrix is a complete description of the graph, it implies all the information about the enclosures and their topological relationships. From Fig. 6, it is easy to understand that if we can find all the basic loops in the graph, we may obtain the spatial information in the building.

3.4. Determine a minimum spanning tree (MST)

In order to identify the basic loops in the established graph, we determine the spanning tree first. A spanning tree of a graph is just a subgraph that contains all the vertices and is a *tree*. Now suppose the edges of the graph have weights or lengths. The weight of a wall edge is equal to the actual distance between the two vertices. The weight of an opening edge is equal to an extremely large value (e.g. 10 times the largest wall edge weight). The weight of a tree is just the sum of weights of its edges. The minimum spanning tree (MST) is the tree that has the minimum weight. There are many algorithms for finding the MST. In this article, we adopt Kruskal's algorithm (Kruskal, 1956). This algorithm is known as a *greedy algorithm*, because it chooses at each step, the minimum edge to add to MST. Kruskal's algorithm is described as follows:

```
Step 1: Give each wall edge a weight equal to its length
Step 2: Find the largest length in wall edges denoted by $L$
Step 3: Give each opening edge a weight equal to $10 \times L$
Step 4: Sort the edges of $G$ in increasing order by weight
Step 5: Keep a subgraph $S$ of $G$, initially empty
Step 6: For each edge $e$ in sorted order
            if the endpoints of $e$ are disconnected in $S$
            add $e$ to $S$
        return $S$
```

Thus, the result S is the MST. The algorithm can also work well on nontrivial connected weighted graphs such as architectural plans.

3.5. 'Basic loop' extraction

Formally, we define a loop as: a loop is a path that ends at the vertex it begins. Every region in the architectural plans can be represented by a loop in the corresponding graph (see Fig. 6). However, NOT all the loops are spatial regions. So the next step is to find these special loops. In order to correspond to a spatial region, the concept of *basic*

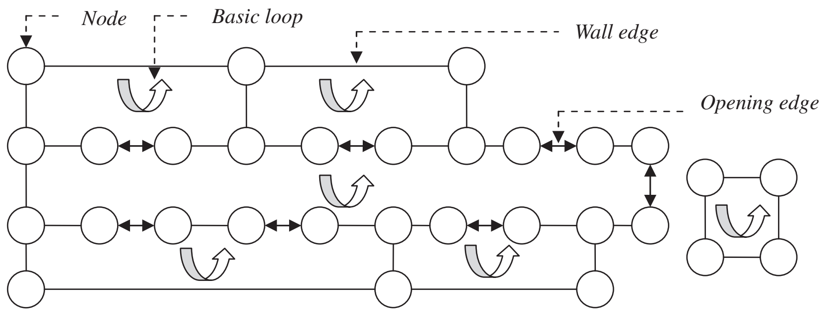

Fig. 6. Construct graphs for plan in Fig. 2.

loop and *basic loop set* (*BLS*) are introduced. Firstly, the signed area of a loop is calculated by

$$\text{Area} = 0.5 \times \sum_{i=0}^{n-1} (y_i + y_j)(x_i - x_j), \quad j = \begin{cases} i+1 & \text{if} \quad i<n, \\ 0 & \text{if} \quad i=n, \end{cases} \tag{1}$$

where $(x_i, y_i),\ i = 0, 1, \ldots, n-1$ is the vertex sequence in a loop. The sign of area shows the direction of loop:

if area $= 0$, then not a loop;
if area > 0, then the loop has an anti-clockwise direction and
if area < 0, then the loop has a clockwise direction.

A loop is a BL only when it satisfies the following conditions:

- must has a non-negative area;
- does not include any loops. For example, if loop L_1 includes L_2, and L_2 includes L_3, then there are three basic loops: L_3, $L_2 - L_3$ and $L_1 - L_2$ and
- does not overlap with any other loops.

It is easy to conclude that a BL corresponds to a spatial region. The BLS is unique for a given graph and only two kinds of relationships exist between any two loops: *detached* or *adjoining* (*neighboring*) (Pais and Ointo-Ferreira, 1998).

In order to determine the BLS, the MST obtained above is employed. A given graph has its incidence matrix $A_{p\times q}$, where p is the size of node set and q is the size of edge set. When a MST of the graph is determined, we can denote $A_{p\times q}$ in the following form:

$$A_{p\times q} = [A^{\mathrm{C}}_{p\times(q-p+1)} A^{\mathrm{MST}}_{p\times p-1}], \tag{2}$$

where A^{C} is the complement subgraph of the MST consisting of $q-p+1$ columns; A^{MST} represents the MST consisting of $p-1$ columns.

We can suppose that the incidence matrix of BLS, which is a subgraph of A, is denoted as

$$B^{\mathrm{BLS}} = \left[EB_{(q-p+1)\times(p-1)}\right], \tag{3}$$

where E is an identity matrix with the order $(q-p+1)$; B is the remaining $(q-p+1)\times(p-1)$ sub matrix corresponding to the MST. Accordingly, B can be determined by the following equation:

$$B = [A^{\mathrm{C}}]^{\mathrm{T}}[[A^{\mathrm{BST}}]^{\mathrm{T}}]^{-1}.$$

Consequently, each row in B^{BLS} implies a basic loop. The BLS is a linear dependent base of the loop space. Every loop in the space can be expressed as a linear combination of the BLS. BLS is unique for a given graph.

3.6. Loops refinement

In general, most basic loops are related to spatial enclosures if they have openings that are represented by *opening* edges. However, there may be two special cases that require special consideration:

1. a loop surrounds another loop and
2. a loop has no opening edge.

4. Information reasoning

By implementing the above graph-based algorithm, all the enclosures inside a building can be identified; obtain their geometrical information and relationships in the architectural plan. In a certain application scenario, the functions and attributes of these enclosures should be further determined. The automatic understanding of an engineering drawing is a complex inference process. The traditional expert system uses too many 'if-then' rules to draw a conclusion, which makes the system hard to maintain. In this study, we adopt an object-oriented reasoning approach.

Generally, an expert system consists of a knowledge base, database, inference engine, explanation system, knowledge acquisition and man-machine interface. An object-oriented inference engine is composed of two bases: knowledge class base and target object class base. The knowledge class base defines all knowledge entities such as rules, functions and each knowledge entity has its own inference engine. The root class of the knowledge base describes the common properties. The target object base is used to implement the inference process for the target

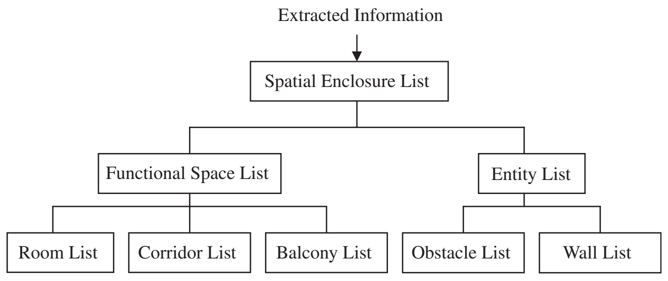

Fig. 7. A sample hierarchy of object-oriented understanding.

object. For example, we may have an object hierarchy showed in Fig. 7. The root denotes the data extracted from a CAD drawing, which is waiting for understanding. Then, inference goes ahead from the root to its children nodes step by step. When the leaf nodes are reached, the objects implied by these leaf nodes are the results of understanding. For example, to identify the spatial enclosures extracted in the above section, we attempt to understand their types such as rooms, obstacles, corridors, stairs, etc. The start point is the root 'spatial enclosure'. Then we attempt to go to the consequent objects using the inference rules of current level. Finally, a leaf node is reached; confirming the type of enclosure is a room or corridor. We adopt deep-first search algorithm to control the inference process. After all the enclosures are identified, the object lists in the leaf nodes are the result of understanding. The inference rules depend on the specific engineering application.

5. A sample application

The enclosure extraction and reasoning approach have been implemented by using C++. We have integrated the above method as the pre-process module into an egress simulation model for buildings, called the spatial-grid evacuation model (SGEM) (Lo and Fang, 2000; Lo et al., 2004) The evacuation model is developed to simulate the evacuation process in large complex building. In order to simulate the movement of occupants, geometry information of the building should be proved as boundary conditions. And the setting in the building is required to be resolved into a network of nodes (e.g. rooms) and arcs (e.g. doorways). In the past, the spatial information is input by writing coordinates of all the key points in a data file. Since there are enormous key points in a building, the modeling process is a long-time work. By implementing the proposed method, the information of the spatial enclosures and their connection relationships are automatically extracted from original architectural plans that are contained in CAD files. It can greatly shorten the time of modeling. Fig. 8 shows the results of capturing spatial information for a multi-auditorium theater. All the enclosures and related geometrical information are successfully extracted. Fig. 9 shows the simulation output of the SGEM.

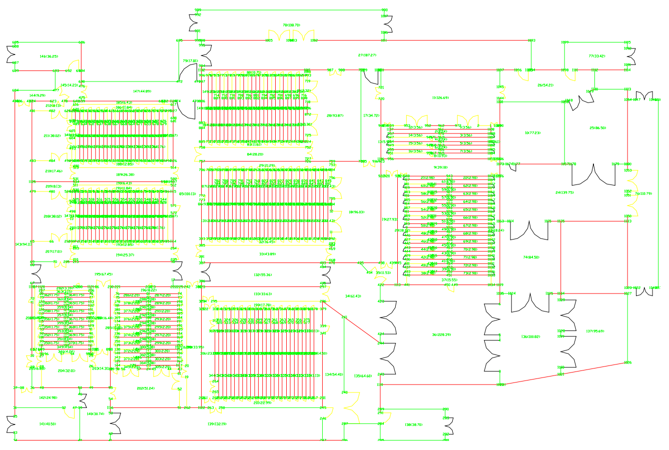

Fig. 8. Results of spatial information extraction.

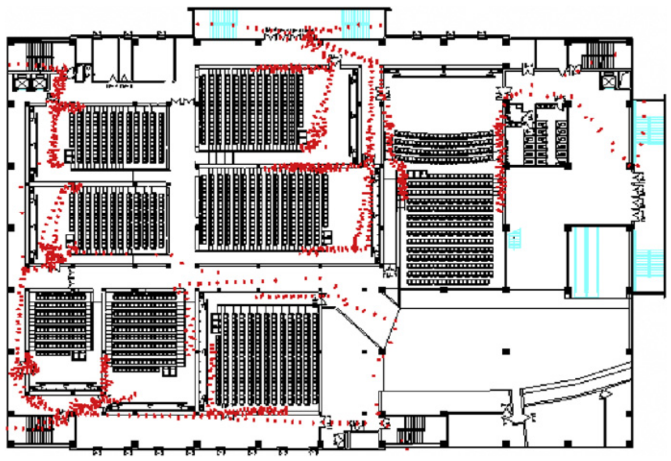

Fig. 9. The sample application of evacuation in a multi-auditorium theater.

6. Concluding remarks

This paper has presented an automatic method to extract architectural spatial information from CAD plan drawings. The method employs graph theory to analyze geometrical primitives and their connectivity relationships. For understanding the attributes of extracted architectural regions/components, we design an object-oriented inference engine to identify the type of each recognized spatial objects. This approach has been implemented in C++. Through practical applications, we found that by using this technique, the costs of spatial information extraction can be minimized. Further development and refinement of the study is undergoing.

Acknowledgment

The work described in this paper was fully supported by the grant from the Research Grant Council of the Hong Kong Administrative Unit, China [Project No. CityU1172/01E (A/C #9040615)].

References

Ah-soon, C., Tombre, K., 1997. Paper Presented at the Proceedings of the Fourth International Conference on Document Analysis and Recognition, Ulm, Germany.

Ah-soon, C., 1997. A constraint network for symbol detection in architectural drawings. In: Tombre, K., Chhabra, A.K. (Eds.), Graphics Recognition. Springer, Berlin.

Bukowski, R., Sequin, C., 1997. The FireWalk system: fire modelling in interactive virtual environments. In: Paper Presented at the Second International Conference on Fire Research and Engineering (ICFRE2), Gaithersburg, MD.

Kruskal, B.J., 1956. On the shortest subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7 (1), 48–50.

Lewis, R., Sequin, C., 1998. Generation of 3D building models from 2D architectural plans. Computer-aided Design 30, 765–769.

Lo, S.M., Fang, Z., 2000. A spatial-grid evacuation model for building. Journal of Fire Science 18, 376–394.

Lo, S.M., Fang, Z., Lin, P., Zhi, G.S., 2004. An evacuation model: the SGEM package. Fire Safety Journal 39, 169–190.

Meeran, S., Taib, J.M., 1999. A generic approach to recognizing isolated nexted and interacting features from 2D drawings. Computer-aided Design, 891–910.

Messmer, B.T., 1996. Automatic learning and recognition of graphical symbols in engineering drawings. In: Kasturi, R., Tombre, K. (Eds.), Graphics Recognition—Methods and Applications. Springer, Berlin, pp. 123–134.

Owen, M., Galea, E.R., Lawrence, P.J., 1996. The EXODUS evacuation model applied to building evacuation scenarios. Fire Engineers Journal 56, 26–30.

Pais, J., Ointo-Ferreira, C., 1998. Search strategies for reasoning about spatial ontologies. In: Presented at the Tenth IEEE International Conference on Artificial Intelligence.

Prabhu, B.S., Pande, S.S., 1999. Intelligent interpretation of CAD drawings. Computer & Graphics 23, 25–44.

Thompson, P., Marchant, E., 1995. A computer model for the evacuation of large building population. Fire Safety Journal 24, 131–148.

Zhi, G.S., Lo, S.M., Fang, Z., 2003. A graph-based algorithm for extracting units and loops from architectural floor plans for a building evacuation model. Computer-Aided Design 35 (1), 1–14.