Error-Correction in Electronic Architectural Drawings

Zili Zhang
Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China
zhzili@gmail.com

Xuan Wang
Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China
wangxuan@cs.hitsz.edu.cn

[1,2]Waqas Anwar
[1]Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China
[2]Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan
waqas@ciit.net.pk

***Abstract*—Recently, three-dimension reconstruction has attracted much attention. The reconstruction from electronic architectural drawing is an important as well as urgent task according to the current situation. Errors in drawings will result in inaccurate three-dimension model, part of the methods based on scanned drawings which have been used before will be incomplete and have some drawbacks, therefore this paper proposes new strategies to deal with different categories of errors in the new system based on electronic architectural drawings.**

Keywords—scanned drawings, electronic architectural drawings, DXF files, error-correction

I. INTRODUCTION

In the field of Document Analysis and Recognition, the reconstruction of three-dimension model from two-dimension architectural drawings had aroused the interests of many researchers, some of whom have gained remarkable achievements, however, the final result disappointed them. Just as Tombre [1] said, "the architectural drawings are between the art and the science". Therefore, the conversion to satisfactory outcome still has a long way to go.

According to our analysis, different kinds of errors in the architectural drawings are the root causes of inaccurate three-dimensional models. "Existing CAD systems, however, do not provide the functions such as the checking of design errors in engineering drawings", Tsujio S. et al. [2] said. So decades ago, researchers had been studying this problem. In 1992, Tsujio S. et al. [2] proposed a method to check dimensions errors that is the most serious error in architectural drawings or engineering drawings. In 1997, Yu et al. [3] classified the errors into three types during symbol classification phase: interconnection errors, labeling errors, residual segmentation errors, which were measured by the numbers of gaps, labels and some new symbol components and connection lines, and solved by an interactive correction module. In scanned engineering drawings, the symbols have certain common regularity, so researchers can use three orthographic projections to study symbols. Poliakoff et al. [4] detected and solved the missing lines during reconstruction phase using three-dimensional model. In 2002, "superstrict" methods in line drawings have been proposed, once there were a few mistakes which cannot meet the tolerance degree, the vertices and edges would be removed, Ros et al. [5] proposed a method, which moved the vertexes to get an approximately complete drawing, to overcome this deficiency. Nowadays people pay special attention to sketch, a lot of researchers and organization probe into this field and gain plenty of achievements. Because the primitive elements are lines, the author [6] of the paper obtained inspiration to study the reconstruction of three-dimensional model. Gribov [7] thought that the vector drawings would bring about parity errors; it meant that in drawings chances for lines which had thickness of even pixels to error are bigger than lines which had thickness of odd pixels, so they present a method to overcome these errors. Liu et al [8] also summarized the errors. Huang et al [9] used the Graph theory to recognize CAD data, and got satisfactory results, but the errors which they discussed were not complete. According to the geometrical, topological and semantic consistency, the author [10] unearthed some inconsistencies; which were just a part of the total errors.

In a word, the literatures listed by us have two main weaknesses. Firstly, their input data were scanned architectural drawings, which must be adopted segmentation and vectorization and so on, during these processes errors were inevitable. Currently the scanned architectural drawings appear less and less while electronic architectural drawing increase day by day; secondly, all of them merely summarized part of errors. Motivated by the above-mentioned reasons, we take the electronic architectural drawings as input data and analyze the whole errors to meet the current industry's need.

The rest of paper is organized as follows. Section II analyzes the characters of errors in electronic architectural drawings. Section III discusses all errors and relevant strategies in details. Section IV shows a three-dimensional model of our system. Finally, section V provides a summary of this work.

II. CHARACTERISTICS OF ERRORS IN ELECTRONIC ARCHITECTURAL DRAWINGS

As it is known to all, the research based on scanned-architectural drawings need a lot of paper drawings, therefore many procedures, such as raster-to-vector, thinned, vectorization, should be adopted. In these processes errors may be made because of our equipments, operations, or algorithms. Nevertheless, if we adopt electronic architectural drawings as the input data, the errors caused by equipments and operations will be eliminated or reduced. Our team develops a system about Three-dimension reconstruction

978-0-7695-4927-9/12 $26.00 © 2012 IEEE
DOI 10.1109/FIT.2012.32

based on electronic architectural drawings, the type of input data is DXF (Drawing Exchange Format) format of AutoCAD's software. Therefore, what we can get from the files is just the coordinates of lines and symbols. This might raise new problems which cannot be seen in scanned architectural drawings. One of the advantages is that the errors which were committed by architect not by our processes, i.e. segmentation, vectorization can be easily detected in electronic architectural drawings.

The errors can be classified into three types: firstly, the drawings miss certain symbols such as lines, namely missing type; secondly, the drawings have redundant symbols, viz redundant type; thirdly, the drawings are incomplete, i.e. incomplete type, e.g. two near lines should be joined together, but there is a gap or over intersection. Some methods detected the errors during the reconstruction stage, but our system detects the error during the parsing stage. During the parsing stage, we use the data extracted from the DXF file, and algorithms to judge the relations of lines. So before the reconstruction of three-dimension models, the errors have already been descried and rectified, the last task is to compute the relations of the components and convert the model to 3ds files to display the three-dimension model.

III. ERROR-CORRECTION IN EVERY CASE

The errors which we will discuss are categorized to three types, known as missing type, redundant type, and incomplete type, respectively.

First of all, we briefly introduce our algorithm about how to tackle these coordinate data. Every time when we get a new line **A**, we will create a new linked list **L**, using the start point of the new line **A** as the first node of **L**. The end point of this new line **A** will be the start point of another line **B**, so we add the end point of line **A** i.e. the start point of line **B** to **L** as the second node, ... Keep searching if the first node of the **L** is the end node of the same **L** (forming a loop), if so, we will get a closed section. Sometimes, this closed section may be a big rectangle which stands for walls in the drawings.

A. Missinig types

We get the coordinate data from the DXF files, and then analyze the input coordinate data. If we cannot get a closed section, which means we may get a partial rectangle due to lose some lines. Using the algorithm above, we can parse that the components miss some lines and we should add this missing line in order to get an intact three-dimension model later.

B. Redundant types

At times, when we execute some operations in CAD software, single lines were copied and plastered unconsciously. This kind of error cannot be seen by unaided eyes in electronic drawings and can be ignored in scanned drawings, but in electronic drawings the DXF files record every operation and every data, namely, all information are contained in DXF files.

C. Incomplete types

Under most circumstances, this kind of errors is very common. The figures below list the kinds of errors and demonstrate some methods to tackle these errors.

- Two vertical lines

This happens when two lines intersect vertically. The architects who operate incorrectly or forget to operate the chamfer command with CAD software can bring about these errors. When we check the linked list of points, the algorithm may encounter these errors and deal with them according to the various situations below.

1. To Fig. 1.a, firstly compare the x-coordinates of the P1 and P2, we must extend x1 to x2, so get x2; compare the y-coordinates of the P1 and P2, we must extend y2 to y1, so get y1; finally extend the P1 and P2 to new Point(x2, y1).

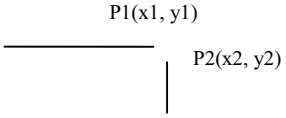

Fig. 1.a Error type of two vertical lines

Algorithm 1-1:

```
if P1.x1 < = P2.x2
        new_Point.x = P2.x2;
if P1.y1 >= P2.y2
        new_Point.y = P1.y1;
extend P1 to new_Point;
entend P2 to new_Point;
```

2. To Fig. 1.b, due to the same y-coordinate of the P1 and P2; compare the x-coordinates of the P1 and P2, we must extend x1 to x2, so get x2; finally extend the P1 to new Point(x2, y1).

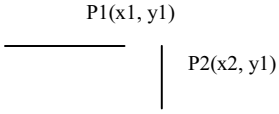

Fig. 1.b. Error type of two vertical lines

Algorithm 1-2:

```
if P1.x1 <= P2.x2
        new_Point.x = P2.x2;
if P1.y1 == P2.y1
        new_Point.y = P1.y1;
extend P1 to new_Point;
```

3. To Fig. 1.c, firstly compare the x-coordinates of the P1 and P2, we must extend x1 to x2, so get x2; compare the y-coordinates of the P1 and P2, we must shorten y2 to y1, so get y1; finally extend the P1 to new Point(x2, y1) and shorten the P2 to new Point(x2, y1).

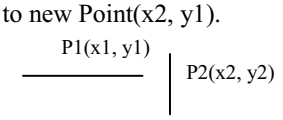

Fig. 1.c Error type of two vertical lines

Algorithm 1-3:

```
if P1.x1 <= P2.x2
        new_Point.x = P2.x2
```

```
if P1.y1 <= P2.y2
        new_Point.y = P1.y1
extend P1 to new_Point;
shorten P2 to new_Point;
```

4. To Fig. 1.d, due to the same x-coordinate of P1 and P2; compare the y-coordinates of the P1 and P2, we must extend y2 to y1, so get y1; finally extend the P2 to new Point(x1, y1).

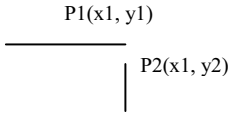

Fig. 1.d Error type of two vertical lines

Algorithm 1-4:

```
if P1.x1 == P2.x1
        new_Point.x = P1.x1;
if P1.y1 >= P2.y2
        new_Point.y = P1.y1;
extend P2 to new_Point;
```

5. To Fig. 1.e, firstly compare the x-coordinates of the P1 and P2, we must shorten x1 to x2, so get x2; compare the y-coordinates of the P1 and P2, we must extend y2 to y1, so get y1; finally extend the P2 to new Point(x2, y1) and shorten P1 to new Point(x2, y1).

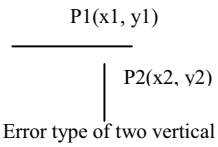

Fig.1.e Error type of two vertical lines

Algorithm 1-5:

```
if P1.x1 >= P2.x2
        new_Point.x = P2.x2;
if P1.y1 >= P2.y2
        new_Point.y = P1.y1;
extend P2 to new_Point;
shorten P1 to new_Point;
```

6. To Fig. 1.f, due to the same y-coordinate of the P1 and P2; compare the x-coordinates of the P1 and P2, we must shorten x1 to x2, so get x2; finally shorten the P1 to new Point(x2, y1).

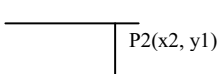

Fig.1.f Error type of two vertical lines

Algorithm 1-6:

```
if P1.x1>= P2.x2
        new_Point.x = P2.x2;
if P1.y1== P2.y1
        new_Point.y = P1.y1;
shorten P1 to new_Point
```

7. To Fig. 1.g, due to the same x-coordinate of P1 and P2; compare the y-coordinates of the P1 and P2, we must shorten y2 to y1, so get y1; finally shorten the P2 to the new Point(x1, y1).

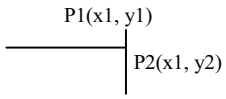

Fig. 1.g Error type of two vertical lines

Algorithm 1-7:

```
if P1.x1==P2.x1
        new_Point.x = P1.x1;
if P1.y1 <= P2.y2
        new_Point.y = P1.y1;
shorten P2 to new_Point;
```

8. To Fig. 1.h, firstly compare the x-coordinates of the P1 and P2, we must shorten x1 to x2, so get x2; compare the y-coordinates of the P1 and P2, we must shorten y2 to y1, so get y1; finally shorten P1 and P2 to the new Point(x2, y1).

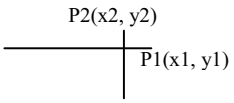

Fig. 1.h Error type of two vertical lines

Algorithm 1-8:

```
if P1.x1 >= P2.x2
        new_Point.x = P2.x2;
if P1.y1 <= P2.y2
        new_Point.y = P1.y1;
shorten P1 to new_Point;
shorten P2 to new_Point;
```

- Two parallel lines

We will exemplify with horizontal parallel lines. The method of dealing with vertical parallel lines is the same as that of horizontal parallel lines.

When architects want to extend a single line, misusing operation can cause four types of errors in the electronic files. This kind of error is very difficult to judge. To solve the first two type errors Fig.2.a and Fig.2.b, we should first get the width of the wall entity in electronic files; this is the criterion whether these two lines are a part wall entity or broken lines with a gap. As to the last two errors Fig.2.c and Fig.2.d, this is a correct line in scanned drawing, but is not a correct line in electronic drawings since the electronic files record all information. By naked eyes this is a complete and single line, actually they are two section lines superposing in the DXF files.

1. To Fig. 2.a and Fig. 2.b, when the absolute value **d** of the y-coordinate of the two parallel lines is smaller than the 0.5 width of the wall entity in electronic drawings according to the specific circumstance, we should merge the two parallel lines to a single one. When the **d** is bigger than 1.5 widths, the two parallel lines should be treated as redundant lines, both of them should be deleted. When the **d** is between 0.5 width and 1.5 widths, the two lines are a wall entity.

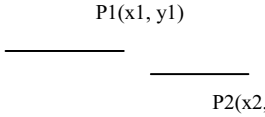

Fig.2.a two parallel lines

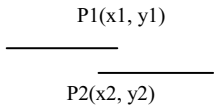

Fig. 2.b two parallel lines

Algorithm 2-1:

d = absolute value (P1.y1 – P2.y2);
if d>0 && d < 0.5width of wall
merge the two parallel lines;
if d> 1.5 width of wall
delete the two parallel lines;
if d > 0.5 width of wall && d < 1.5 width of wall
the two lines are a wall entity;

2. To Fig. 2.c, these two lines are parallel and overlapping; one of the line is just a part of the other line, delete the part line and use the complete line to get rid of errors in the reconstruction stage.

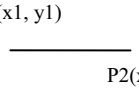

Fig. 2.c two parallel lines

Algorithm 2-2:

if P1.y1 == P2.y2
&& the line in which P1stands is a closed section
&& the line in which P2stands is single line
delete the line in which P2 stands;

3. To Fig. 2.d, one of these two lines in some section is copied. When we analyze the two lines, the linked list has two lines with the same start point and same end point, one of them is a single line which does not link any other lines, so we should delete the isolated line.

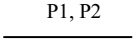

Fig. 2.d two parallel lines

Algorithm 2-3:

if P1.x1 == P2.x2 && P1.y1 == P2.y2
&& the line in which P1stands is a closed section
&& the line in which P2stands is single line
delete the line in which P2 stands;

- Multi-lines

In the CAD electronic drawings, multi-lines are always used to demonstrate the wall entity, as Fig.3. Therefore, this kind of error is the variant of the Fig. 1. According to the eight strategies above, these errors can be corrected.

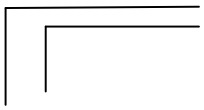

Fig. 3 multiline

- Doors and stairs

In the CAD electronic drawings, the doors entity and stairs entity may have some deficiencies, not all architecture components are satisfactory with standard symbols.

To solve the Fig. 4's errors, when we analyze the CAD electronic drawings, if we find that a rectangle and a quarter arcs nearly touch together but there is still a little gap between them, we assert that this is a door. In reconstruction phase we will use a model which is prepared in advance to present this door. Of course, there are many kinds of door entity, but no matter what kind of door it is, we will first summarize the feature of the door, and then we treat them on the basis of this idea.

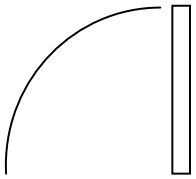

Fig. 4 one kind of doors entity

The stair entities as Fig.5 are the characteristic of many parallel lines. Therefore, when we find many parallel lines, even if there are some gaps, we can assert that this symbol is a stair entity.

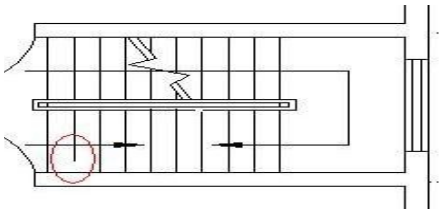

Fig. 5 one kind of stairs

Algorithm 3:

if there is a quarter circular arc
&& there is a rectangle
&& the distance of arc and rectangle is smaller than the threshold
it is a door;

Algorithm 4:

if there are many parallel lines
&& distance of two adjacent lines is smaller than the threshold
it is a stairs;

IV. EXPERIMENT

In this part, we show the outcome of our system — three-dimensional model. Fig.6 is one of the results.

This is a 10-storey hotel. Our system firstly reads the electronic drawing, analyzes every component, and then rectifies the errors using the methods elaborated in this paper, at last reconstructs a complete three-dimensional model.

This system has used all the previous algorithms, if any algorithm is incorrect, the three-dimensional model cannot be reconstructed.

Fig. 6 10-storey hotel.

Some commercial companies have their own three-dimensional reconstruction systems, but it is impossible to make comparison between their systems and our system due to the confidentiality.

V. CONCLUSIONS

Although errors are inevitable in architectural and engineer drawings, we can tackle them with sophisticated methods. Our team has developed a system based on electronic architectural drawings. We analyze all the kinds of errors and give the relevant strategies to solve them, which is different from the previous methods.

ACKNOWLEDGMENT

To develop this interesting and challenging project, every member in our team has paid great efforts, so herewith our sincere gratitude to the laboratory members of Jia Cao, Zhongmei Zhou, and Zhiwen Liu.

REFERENCES

[1] Tombre K., "Analysis of Engineering Drawings: State of the Art and Challenges," The 2nd Ini'l Workshop on Graphics Recognition, Nancy, France, 1997.pp:257-264.

[2] Tsujio S., Ono T. and Lee S.S., "Computer-Aided Drawing Check for CAD Systems—A Method for the Checking of Dimensions in Multiview Mechanical Drawings," IEEE Int. Conf. Systems Engineering Kobe, Japan (Sep 1992) pp:234-237

[3] Yu YH., Samal A., Sharad C.Seth., "A system for Recognizing a Large Class of Engineering Drawings," IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 19(8)(1997) pp:868-890.

[4] Poliakoff J.F., Thomas P.D., "Error Correction in Scanned Engineering Drawings Using three-dimensional Knowledge-Based Reconstruction.Graphics Recongintion Algorithms and System," Springer Berlin/Heidelberg, 1389/1989, 1998,pp:280-290.

[5] Ros,L. Thomas,F., "Overcoming Superstrictness in Line Drawing Interpretation," IEEE Transactions on Pattern Analysis and Machine Intelligence. 24(4)(2002) pp:456-466.

[6] Jatupoj, P., "Sketchboard: The Simple three-dimensional Modeling From Architectural Sketch Recognition." Proceedings of the 10th International Conference on Computer Aided Architectural Design Research in Asia.NewDelhi(India) 28-30 April, 2005, 1, pp:1-8.

[7] Gribov, A. Bodansky, E.,"Vectorization and Parity Errors," LNCS 3926, 2006,pp:1-10.

[8] Liu,W.Y., Zhang,W., Luo,Y., "An Interactive example-driven approach to graphics recognition in engineering drawings", International Journal Document Analysis and Recognition (2007) 9,pp:13-29.

[9] Huang,H.C., Lo,S.M., Zhi,G.S, Yuen,KKR, "Graph theory-based approach for automatic recognition of CAD data", Engineering Application of Artifical Intelligence, 2008, 21,pp:1073-1079.

[10] Horna,S.,Meneveaux,D.,Damiand,G.,Bertrand,Y.,"Consistency constraints and three-dimensional building reconstruction", Computer-Aided Design 41(2009),pp:13-27.