Highly Automatic Approach to Architectural Floorplan Image Understanding & Model Generation

Siu-hang Or, Kin-Hong Wong, Ying-kin Yu, Michael Ming-yuan Chang*

Department of Computer Science & Engineering Department of Information Engineering*
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Email: shor,khwong,ykyu@cse.cuhk.edu.hk mchang@ie.cuhk.edu.hk

Abstract

Authoring of a 3D model using computers typically involves tedious manual efforts. In this paper, we propose a highly automated approach to generate the 3D model of a building from its 2D floorplan image. The proposed system treats the floorplan as a vector image in that the lines in the plan are converted into vector outlines. Building wall identification is thus reduced to a matching problem of the outlines in the plan. A number of advantages are resulted from using this approach. For example, special constructs such as doors can be recognized through the geometric characteristics of the line signatures. The concept of interior space can also be explored through examining the line loop within the plan. Our system allows the generation of building interior as well as the exterior model. The presented algorithm is useful for data generation in architectural walkthrough of city scenes as well as 3D games development.

1 Introduction

An architectural floorplan has the feature of abstracting much spacial information in a 2D postulation. However, authoring of the corresponding 3D model using computer would involve tedious manual efforts. Much work therefore focus on speeding up the construction process by using the floorplan document as input, and try to generate the corresponding 3D model. The primary challenge here is the problem of symbol recognition [3], in which an automatic way is sought to identify the various special constructs inside the floorplan such as doors and windows. A number of researchers [4, 2, 7, 6] had investigated the problem and impressive results have been established. However to our understanding, most existing algorithms focused solely on the symbol recognition task [2, 6] and did not provide a complete solution for the reconstruction problem. In our opinion, the reconstructed 3D model should be easily edited by an ordinary user so that additional modeling efforts can be made.

In this paper, we present a complete system to tackle the problem of generating 3D model description of a building solely from its 2D floorplan. Our contribution primarily lies in the 3D model authoring process, which is important in architectural walkthrough as well as computer game development. Using our approach, an ordinary user can produce a 3D model of a building in relatively short time. Our proposed approach has three important features which contribute to the state-of-art techniques in producing the 3D models of a building: 1) The reconstruction result retains the original volumetric attributes inside the floorplan, such as walls, windows and doors; 2) the model generated matches with the original floorplan with a precision up to pixel level; 3) Finally the reconstruction process is very fast that it can save lots of manual efforts to produce a model of the same complexity, and that the results are best suited for further artistic editing such as texturing and adding fine details. The presented framework will be useful for content generation in architectural walkthrough on old buildings as well as 3D games development.

2 Previous work

Symbol recognition is a well developed discipline inside the field of pattern recognition [3]. Symbols on a paper drawing in applications such as circuit diagrams, geographic map and engineering drawings are recognized and provide to the user for further usages. Among these applications, architectural drawing is a relatively unexplored field [2]. [6] proposed a string based method to recognize both the symbols and texture inside a floorplan by constructing a Region Adjaceny Graph (RAG) from the raster image. Graph matching technique is applied to extract the symbols represented in the plan. [2] proposed a neural network based solution to identify the symbols, essentially the doors and windows of the target floorplan. A 3D elevation is use to illustrated the result, which is quite impressive. In [4], a complete system for analyzing architectural drawing is presented.

All the above approaches focused primarily on the problem of symbol recognition inside the floorplan, which is essentially a hard problem and is recognized as requiring problem dependent solution. However, from the computer graphics point of view, the problem is only solved partially. The remaining problem, which is also of importance, is the format of reconstructed 3D model. As most of the modern graphics application is polygon based, a user would expect the reconstructed model to be also of polygon ingredient. Polygons, as a planar structure, needed to be grouped together as a mesh for further manipulation. This intermediate representation is crucial to all the subsequent modeling work onto the final product.

[7] proposed an algorithm to interactively reconstruct the exterior of a building from an aerial images. A fitting algorithm is proposed to identify a building within an aerial images. The problem being addressed is basically different from what we are going to tackle, namely to reconstruct the interior of a building, i.e. the constructs inside an architectural floorplan.

3 Our Approach

Fig. 1 shows part of a typical floorplan of a building. A readily observation is that a floorplan consists of a number of lines of varying width postulating the top view of a building. The lines designate walls of different thickness. To represent openings on the walls, e.g. doors, windows etc., special signatures such as a pair of thin lines is used to denote that an opening should be placed there. Depending on the nature of the building, usually a set of rules or signatures is used to represent the deviation from the case of simple wall.

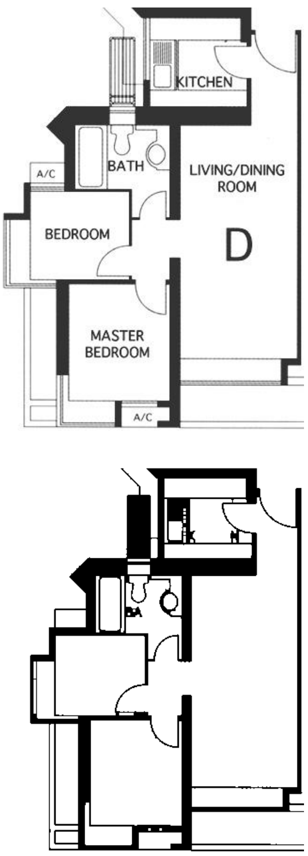

Figure 1: Input to our proposed system. left: original floorplan image, right: preprocessed result.

To automatically convert a floorplan into the corresponding 3D description, we propose to parse the floor plan image into a number of connected segments and analyze their relationship to generate the 3D models accordingly. The proposed system is shown in fig. 2 and we will describe in details each step in below.

3.1 Preprocessing of input floorplan image

Our assumption about the floorplan to be reconstructed is that they are in hard copy format, raster input is thus needed to digitize them into bitmap images for further processing. The scanned image is properly thresholded into a binary image so that

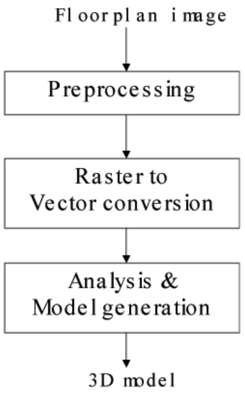

Figure 2: Block diagram of the proposed system

only the lines representing the construct are left. Textual patterns as well as other irrelevant symbols also need be removed. Statistical techniques are employed to extract the textual/symbol blocks inside the image and segmenting them out. In addition, sampling noise during scanning of the floorplan document should also be removed as they will affect the correct interpretation during later algorithm application. Bottom of fig.1 shows the result after the preprocessing stage. As our proposed system focused on the generation of the 3D model of the building, some symbols which are not intended to be reconstructed, for example the cupboard and sink in the kitchen, are therefore have to be removed from the image through human effort. And this is the only manual intervention needed for our system. An example of the result of this stage can be seen in fig. 3.

3.2 Raster conversion

The output from the previous stage is fed into a raster to vector conversion and the outlines are extracted. Typical implementation will apply thinning to the input image to extract the skeleton of the plan. However in our system only double lines i.e the outline of the floorplan are extracted. In addition, straight lines and arcs are also classified as arcs usually corresponds to special constructs such as doors in a floorplan. We also record the change in angle of each arc during this process since they are useful in later recognition stage. The result obtained after vectorization can be seen in fig. 3.

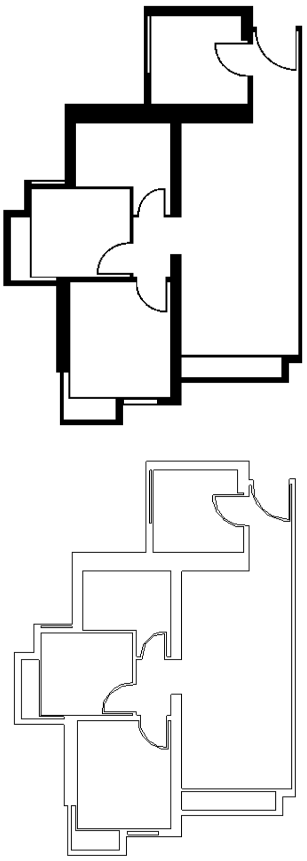

Figure 3: Preprocessed floorplan image for input to vector conversion and vectorized result. Top: final preprocessed image for vectorization, bottom: vectorized result.

3.3 Recognition & Generation

The core of our algorithm lies in converting the polylines from the previous stage into 3D descriptions. Owing to the nature of the raster conversion, there is a built-in relationship for the direction of the polyline and the structures inside the floorplan: the building wall must be on the right of the propagating direction of a polyline and open space should be on its left.

Moreover two polylines with opposite propagating directions and with some distance apart should

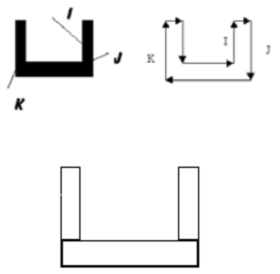

Figure 4: Example in polylines matching. Input image is on the left and converted polylines are shown on the right. The matching result is shown at bottom.

correspond to a wall, as can be seen from fig. 4. In fig. 4, the line *I* and *J* should be matched together and form a solid wall, which we called a *block*. By noting the coordinates of the four points, we can generate a *block* and thus representing the corresponding wall. However this simple criterion would break down in real cases.

In fig. 4, consider line *J* and *K*. There is part of line *K* that matched with *J* which takes account of the horizontal wall. A simple comparison would reject line *K* in favor of line *I*. In that case, the corner portion cannot be recovered.

To solve the problem of multiple matching and the problem above, we propose a concept of *span list*. For each line in the floorplan, a list of span length is maintained. The span list contains spans of all other possible candidates to this line. A span is defined as the length of the vertical projection made by another line. The idea is that during span list construction, those segments that have overlapping i.e. interesecting spans, should resolve for the correct span and finally lines with multiple matching can generate the correct *blocks* based on the spans calculated. It is possible that a line can have have more than one spans/projections e.g. in fig. 4, line *J* has projections (spans) with both line *I* and *K*. The algorithm for matching two lines is thus shown as below.

To extract the segments that match each other, the following criteria must be fullfilled

- The two lines have opposite directions,
- The two lines have overlapping(s) on one of the *x*- or *z*-axis at least,
- The two lines have the matching line on its right, those not satisfying this criterion might be the windows or doors;
- In addition, do the following for each pair of lines, *I* and *J*, to be matched :

```
1. Determine the span of
   of line I on J. For the
   calculated span, check
   whether there are any
   intersection with existing
   spans in line J.
2. For each span that have
   intersections with existing
   span on J do:
   if distance of line I from
   line J> intersecting span
   distance
      // intersecting span is
      closer
      // this span is being
      rejected
      remove this span from
      our line span
   else
      // our line is closer
      remove the span length
      of our line from the
      intersecting span
      register this span and
      the new span distance
```

Note that in the above, step (2) is applied to both lines *I* and *J* i.e. span of *I* with *J* and *J* with *I*.

After execution of the above algorithm, we obtain a number of square *blocks* which represent walls of different thickness of the floorplan.The matching results for the previous U-shape example are also shown in fig. 4.

3.4 Symbol Recognition

To recognize other special constructs inside a floorplan, we consider the geometric characteristics of the symbols. In a floorplan, the frequently encountered symbols are doors and windows. The recognition of such symbols is not a simple task as it involves analyzing complicated structures including arcs. In our presented framework, we made a simplifying assumption that all arcs involved only corresponds to special symbols i.e walls with curvature

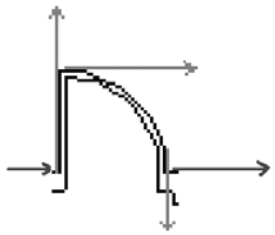

Figure 5: Construct of a door after vectorizing. The segments forming a door undergo several changes in angle and these are the matching charactertistics for the proposed system to identify the symbol.

are not allowed, we use the following procedure to extract an arc. Firstly the angles of all the segments extracted from the vector map are recorded. A comparison of the angle differences between any segment and its previous and next one is made, and the segment is said to be part of an arc if the differences are significant e.g. large than 0.1 radian. The start and end of an arc is then recorded and the in between segments will not take part in further processing so as to prevent it from creating any block. The angle change from a start to the end segment of an arc is also recorded for that arc.

3.4.1 Door

The construction of a door symbol is shown in fig. 5. The features of a door include:

1. It consists of an arc being connected to a line. The arc should have an angle change of 90 or 180 degree with respect to the line (as indicated by the change in arrow direction in fig. 5).
2. The line has a length roughly equal to the distance between the starting point of the line and the ending point of the arc.
3. Both the arc and the line are connected with two lines with the same orientation.

The above features uniquely identify a door inside a floorplan and we can use it as a recognition function which will be performed after the matching of *blocks* has been performed. In fact, the occurence of an arc is such an salient characters of the symbol inside the floorplan that can easily help the recognition.

3.4.2 Window

For the detection of window, noting that the window symbol usually result in one or more thin boxes with the same orientation being placed very close together within a small space. An opening i.e. window, can thus be identified at that area.

3.5 Room Identifcation

An advantage of using the vectored double-line representation in our approach is that the notion of enclosed space can be readily identified. It can be observed from the vectorized plan that when tracing along a particular polyline, a closed area will be resulted. This enclosed area corresponds to either windows, bay windows, or rooms. By converting into actual scale in human world, a room can be identified easily from the vectored description. Recognition of rooms will provide many useful information for the floorplan analysis. In fig. 6 we illustrate a simple usage in automatically generating a light source and placed it at the centroid of the room polyline.

3.6 Building Outlook Generation

Another usage of the enclosed space identification is to detect the polyline enclosing the exterior of the whole floorplan. The reconstruction of the exterior of a building is not as simple as one might expect. The main problem here is to identify those blocks which do not belong to exterior and skip their generation. Moreover the exterior look of windows or bay windows would also call for different generating process from their interior correspondents.

To identify the outerhull of the floorplan, the simple notion of longest polyline is not valid as the plan may have a structure which produces a complex interior polyline with a total length greater than that of the true outerhull. To locate the correct outerhull, we based our application on the observation that the outerhull of a floorplan should enclose the maximum area within the image. For a polyline description, the area being enclosed is given by

$$\frac{1}{2}\sum_{i=1}^{N}(x_i y_{i+1} - x_{i+1} y_i), \qquad (1)$$

where *N* is the total number of points in the polyline, (x_i, y_i) is the coordinates of the point *i* where *i=1,2,...N* within the sequence of points forming the polyline.

To locate the polyline corresponding to the outerhull is thus reduced to finding the polyline enclosing the largest area withint the image. This line will then be used to match with the neighboring polylines to further identified the structure it corresponds to, say a wall or window. The process is essentially the same as that described in section 3.3.

4 Implementation and Discussion

We implemented the proposed approach and tested the generated 3D files on *Genesis3D* [1], which is a 3D game engine available freely. The engine allows interactive walkthrough of the generated model so that one can immediately inspect the model generated. In principle, given that the polygonal description of most 3D file formats are basically the same, it is relatively easy for one to modify the program to generate files in other 3D formats.

For the preprocessing step, we used the QGAR tools [8] to implement the the steps described. In particular, we threshold the input image by the binarization technique described by Trier [9]. Textual symbols are identified by grouping the image data into blocks and computing the distribution of dimensions of the bounding box associated. By making assumptions about the typical textual information within the plan, those blocks belong to textual data can be segmented out, leaving the structural image intact. Further image closing operations are performed to remove the salt-and-pepper noise in the scanned image. The above steps being implemented as separate modules, are invoked together in a script file which further simplifies the preprocessing procedures.

During the raster conversion stage, we need to convert the binary image into a vectored format, i.e. the polylines. We use *Ras2Vec* (Raster to vector conversion) program which is written by Libenzi [5]. The polylines generated are then stored in a text file for the application of our algorithm.

The conversion of the polylines into block descriptions is performed by a *Python* program which carries out the algorithm described above. The *Python* program also directly generates the data file suitable for use by *Genesis3D*. For the 3D model generation, typical parameters are used to construct the model. For example, the height of each building block is set to be 2.5 units, with each unit corresponds to one meter in human world. Each window is assigned a value of 1 unit tall at 1 unit height above the floor. A user can thus perform a walkthrough on the data file, which is very convenient for further refinement. A screenshot of the generated 3D model of fig. 1 is shown in fig. 8.

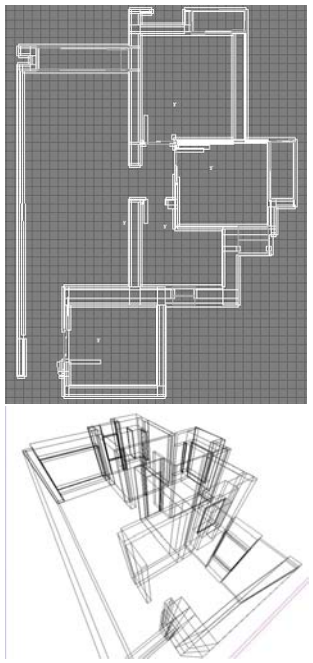
Figure 6: Floorplan in fig. 1 being converted into 3D model. Top: Converted model at top view with lights being inserted at centroid of each room, the lights are represented as a symbol 'x' in the layout. bottom: 3D wireframe view which illustrate the various blocks forming the model

4.1 Discussion

In this paper, we present an efficient approach to quickly generate 3D data from a floorplan image. The generated model can then be used in virtual reality or computer game application. Usually for these applications, beside lowering the cost of model generation, one would also want to modify as well as fine tune the results so that more interesting effects can be provided for final applications.

In our experiments, the proposed approach generates 3D models which have high precision with respect to original floorplan. Moreover relatively little knowledge is required for a user to generate a 3D model from the floorplan. This opens up more avenues for this approach. One interesting application is an integrated environment where the user just has to draw lines and place special symbols denoting corresponding structures on an image and the program will build a model for an interactive tour. The application should be well suited as a creativity tool for children, etc. Another advantage is that the time used to build a 3D model is significantly reduced, for example, the 3D model in fig. 10, which presents a complicated floorplan, is generated in less than 4 hours, including the image processing time. In addition, our system can generate both the interior and exterior model at the same time, which is an added advantage. Finally, by specifying the number of levels of the resulting building, the user can conveniently obtain the 3D model of a multi-storey building in just a command only, this can be seen from the result in fig. 10.

The drawbacks of the proposed approach is that sometimes a few mismatches will be generated during the recognition and generation stage. Fig. 7 shows the mismatch results of data set in fig. 10. The mismatches are highlighted using the ellipses circling around them. These blocks when viewed in the 3D engine would correspond to extra or transparent(due to reverse vertex orders) blocks. However we noted that the number of mismatches is very few comparing with the correct blocks generated.

In addition, the recognition rate of door is not perfect. For example, there is all together 5 doors in the floorplan in fig. 1. Our current system is able to detect four of them with one door missing. The reason seem to be that the matching result for an arc will have a direct impact on later stage of door extraction as door symbol recognition involves a number of constraints to be satisfied. Too loose the matching rule will lead to many false matching. On the other hand a tight bound on the criteria will have the result of missing symbols. Tuning of the matching parameters is a major step for the proposed system to be of industrial usage.

Finally the speed up of model building is spectacular. Consider fig. 10 and 9, which shows complicated floorplans together with the reconstructed models. Both models are of approximately eight times more complicated than the previous example(c.f. 8 flats together in same plan). The automatic algorithm can successfully reconstruct the entire model as seen with only a few mismatched block which can be easily removed with the aid of level editor.

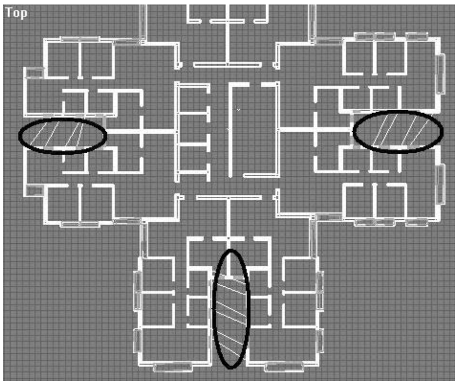

Figure 7: The effect of matching errors in block generation process will produce extra blocks, which are highlighted by the ellipses.

5 Future directions

As modern buildings are mostly of multi-storey type, our proposed system will produce more interesting results if the system can generate the 3D model of a building with many floors. In fact, our system can immediately produce a 3D mesh with more than one floor with little changes e.g. adding a ceiling to each floor and repeat the generation with increasing parameters. Another important addition that we have to work on is the extraction of stairways in a floorplan. In general, a floorplan of a building usually include the designation of stairway which lead to both up or down stairs. The problem then is to locate and extract the symbol and its orientation e.g. how to lead up and connecting the two floors seamlessly.

Another possible future direction is to further automate the rasterizing step. In our proposed system, the preprocessing step is important in that it involves human interactions to remove noises as well as smooth out the outlines for subsequent vectorizing step. More intelligent drawing adjustment routines should be developed to help in further reducing the manual work.

We conclude that using image processing and symbol recognition techniques, we are successful in substantially reducing the time needed to construct a 3D model from a floorplan. As a simple benchmark of the performance of our algorithm, we noted that the floorplan presented in fig. 1 originally need a time of approximately one to two working days to build the resulting model using plain estimation and hand editing of data file, in contrast with the time reported above. One may argue that using sophisticated modeling package, one can significantly reduce the modeling time. But as the modeling package nowaday are written with generic purposes in mind, it would need extensive configuration work in order to build the same model. An added requirement in this case is an experienced user of the particular package is needed, which will also add to the cost of construction. Our algorithm takes advantages of the orthogonal feature of architectural floorplan, and the redundant constructs in man-made building, and that our system can generate both the interior as well as outlook of the building. The resulting algorithm can thus save more times in comparison to more general modeling approaches.

6 Conclusion

An efficient algorithm to construct a 3D model from a floorplan image is presented. The method adopts a recognition approach to extract various constructs inside a floorplan image by analyzing the polylines generated from the raster image. Each polyline is used to generate a set of polygons which represent that particular block. The proposed system can generate both the interior model as well as the building's outlook. Both merits and drawbacks of the algorithm are discussed and a number of results are presented.

7 Acknowledgement

This work was supported by the Research Grant Council of Hong Kong Special Administrative Region. (Project No. CUHK4204/04E) The author would also like to thank the anonymous reviewer for helpful comments on the paper.

References

[1] Genesis 3D. Genesis 3d: An open source 3d game development engine. *http://www.genesis3d.com*.

[2] Christian Ah-Soon. A constraint network for symbol detection in architectural drawings. *Lecture Notes in Computer Science: Graphics Recognition–Algorithms and Systems*, 1389:80–90, 1997.

[3] Atul K. Chhabra. Graphic symbol recognition: An overview. *Lecture Notes in Computer Science: Graphics Recognition–Algorithms and Systems*, 1389:68–79, 1997.

[4] Ph. Dosch, K. Tombre, C. Ah-Soon, and G. Masini. A complete system for analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, 3(2):102–116, December 2000.

[5] Davide Libenzi. Raster to vector conversion program. *http://www.xmailserver.org/davide.html*.

[6] Josep Llados, Gemma Suchez, and Enric Mart. A string based method to recognize symbols and structural textures in architectural plans. *Lecture Notes in Computer Science: Graphics Recognition–Algorithms and Systems*, 1389:91–103, 1997.

[7] F. Rottensteiner. Semi-automatic building reconstruction integrated in strict bundle block adjustment. *Proceedings of the XIXth ISPRS Congress at Amsterdam*, XXXIII-B3:461–468, 2001.

[8] QGAR Software. Qgar software. *http://www.qgar.org*.

[9] S. Due Trier and T. Taxt. Improvement of "integrated function algorithm" for binarization of document images. *Pattern Recognition Letters*, 16:277–283, March 1995.

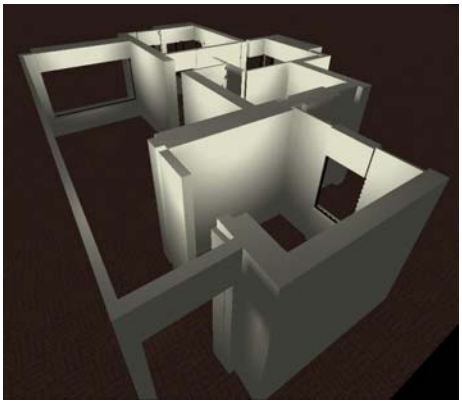

Figure 8: View of fig. 6 with additional work such as texture and lighting.

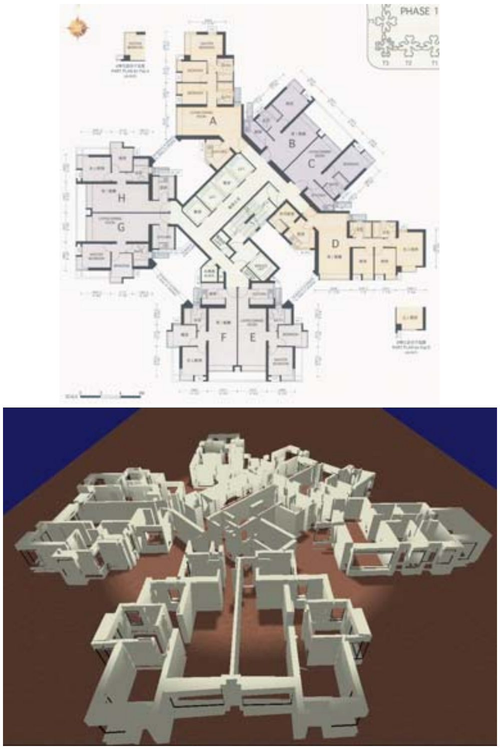

Figure 9: More example. Top: Original floor plan image; bottom: rendered view of output of our algorithm with a light source placed at the top

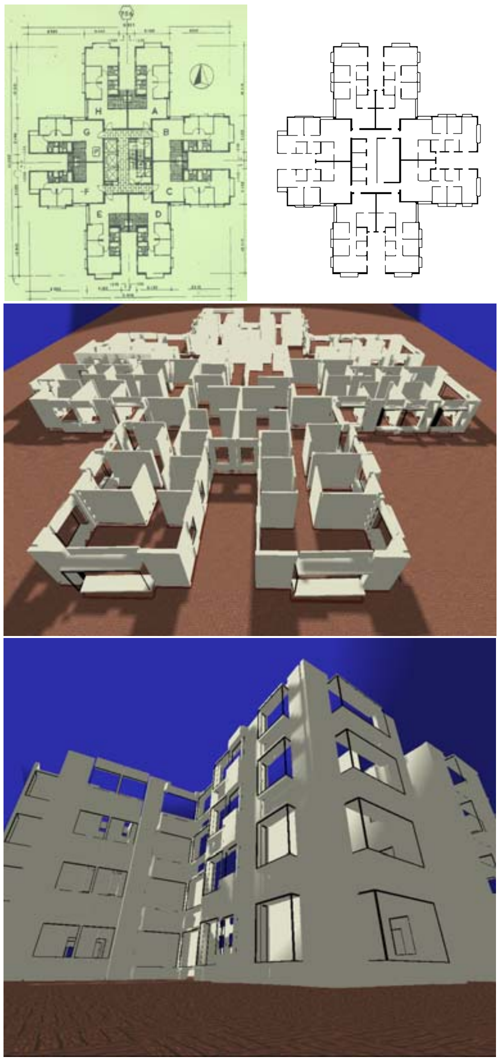

Figure 10: More challenging example. Top left: input floorplan, top right: preprocessed plan; middle: reconstructed interior model, bottom : exterior model obtained by specifying 4-storey building output.