A case-based design with 3D mesh models of architecture

Hao Hua

Key Laboratory of Urban and Architectural Heritage Conservation (Southeast University) Ministry of Education, China
School of Architecture, Southeast University, 2 Sipailou, Nanjing 210096, China

ABSTRACT
This paper presents a case-based design (CBD) focusing on the segmentation of building models and the synthesis of new compositions. Our TRAMMA (Topology Recognition and Aggregation of Mesh Models of Architecture) program is developed as an application of the method. The segmentation includes the recognition of floors, stairs, and walls out of the mesh models. The aggregation process recombines the building segments according to the user-defined topology. The topology representation of multi-story buildings and the spatial synthesis of 3D building models are integrated in the program.

1. Case-based reasoning and architecture
Case-based reasoning (CBR) has been characterized as a formal methodology for reasoning from experience [1]. The theory of CBR sheds light on the cycle of retrieving cases, evaluating/adapting the cases in new situations, and storing/retaining the cases [2,3]. Its significance increases when reasoning from first principles is not adequate in the domain [4]. Thus, this methodology should benefit the field of architectural design as well. However, the subject of CBR in architecture is usually limited to the floor plans rather than the 3D models of buildings. With few exceptions, they only deal with the floor plan layouts with orthogonal walls. These limitations are attributed to the difficulties in indexing various information in buildings. Case indexing is regarded as a key issue in CBR, since it encodes our knowledge of the cases and substantially shapes the way we reuse the cases. People have to decide how to represent and organize the knowledge of the cases (as solutions) and the knowledge of the corresponding problems. As Maher and de Silva Garza [5] state, “The most important issue when defining a case representation is what information in the design facilitates its reuse”.

It is not easy to establish a general and versatile representation of architecture. There are several reasons. First, the design problems have been regarded as ill-defined problems or “wicked problems” [6]. More precisely, both the design criteria and the solution space are difficult to be formally defined. Second, there is no general model of architecture that has been widely accepted. Building Information Modeling (BIM) [7,8] aims to establish a standard modeling platform through integrating various models in the Architecture, engineering and construction (AEC) industry. Nevertheless, various difficulties arise when the distinct designers replace their domain models with the standard BIM model. Thus, some researchers (e.g. [9]) chose to develop additional software to adopt the BIM standard. Third, architectural design seldom leads to the “query-retrieval” procedure—a list of specifications leads to solutions [3]. Rather, the process of query, retrieval, evaluation, and adaptation are diffused in the design process.

2. Topology representation and synthesis
Case-based design (CBD) has been treated as an application of CBR in design [5]. Heylighen and Neuckermans [10] and Richter...
Table 1

<table>
<thead>
<tr>
<th>Representation</th>
<th>Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology of floor plan (2D)</td>
<td>Topology of multi-story building (3D)</td>
</tr>
<tr>
<td>Floor plan (2D)</td>
<td>Multi-story building (3D)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program</th>
<th>Representation</th>
<th>Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archie-II [12]</td>
<td>No a</td>
<td>No</td>
</tr>
<tr>
<td>SEED-Layout [14]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IDiom [13]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CADRE [15]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Rosenman [20]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>de Silva Garza [21]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>a.S.Catch [17]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Grabska [18]</td>
<td>Yes</td>
<td>Yes c</td>
</tr>
<tr>
<td>Langenhan [9]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Merrell d [19]</td>
<td>Yes</td>
<td>Yes d</td>
</tr>
</tbody>
</table>

a The topology is described as text rather than in a formal language.

b The program adapts the cases according to geometric or topological concerns.

c The program can reason about the topology in floor plans.

d The approach learns topology from cases, however, the structured data of the cases are manually encoded.

et al. [11] reviewed a set of CBD programs for architecture, casting light on the structure and organization of knowledge as well as on the reasoning process (retrieval and manipulation). This section compares a series of CBD programs from two aspects: topology representation and synthesis. Topology representation is essential for formulating the relationship between the rooms in buildings, and subsequently spatial synthesis could use the topology information to adapt and manipulate the cases.

Archie-II [12] is one of the earliest CBD applications in architecture. It represents each case by floor plan and verbal descriptions. Most CBD approaches caught the geometric information of floor plans; however, they usually assumed that the floor plan is made of rectangles [13,14] or right-angle polygons [15]. Alternatively, the recent work by Dillenburger [16] employed a pixel-based representation of buildings and its environment. Besides, a number of works [13–15] succeeded in representing the topological configurations of floor plans. For instance, Langenhan [9] and Weber [17] extracted the topological configurations from BIM models as graphs, benefiting from the semantic information embedded in the model components. Moreover, Grabska et al. [18] extended the topology of 2D floor plans to the topology of multi-story buildings. Merrell [19] employs Bayesian networks to learn topological/geometric information from cases (multi-story buildings) and synthesizes 3D buildings according to the learned topology. However, the original topological/geometric information (the training data for the program) is manually encoded from cases. It is important to emphasize that most CBD approaches deal with floor plans instead of 3D models (Table 1). There have been diverse strategies for manipulating the case for new designs. CADRE [15] supports both the dimensional adaptation and the topological adaptation. IDiom [13] and SEED-Layout [14] can make new compositions by re-combining the parts of previous floor plans. Intelligent synthesis was a big challenge to the early CBD programs. More sophisticated synthesis usually involves advanced search methods such as evolutionary algorithms, e.g., Rosenman [20] and de Silva Garza [21]. The evolutionary process in both programs can manipulate the cases and generate new designs satisfying user-defined objectives. Nevertheless, the synthesis of these programs is conventionally limited to floor plans (Table 1).

The CBD approaches listed above indicate a need for representing and synthesizing multi-story buildings. Thus our TRAMMA program commences with the 3D models of building. The major challenges of the work include: (1) recognition: extracting the topological information from the 3D mesh models, in other words, breaking the model into meaningful parts such as floors, walls, and stairs. (2) synthesis: combining the parts of cases into multi-story buildings that satisfy specified topology. In opposite to the programs in Table 1, our research emphasizes the analysis and manipulation of 3D models of multi-story buildings rather than that of floor plans.

3. Processing 3D models of buildings

3.1. Overview

The input data of our TRAMMA program (written in Java) is a repository of 3D mesh models of buildings (Fig. 1). In the stage of recognition, the program extracts the essential structure—floors out of the mesh models. After that the walls and the vertical
The element (floor–connections–floor) is extracted from the sample model (a, b, c, d). Combining the elements as a new skeleton of building according to the user-defined topology (e). The walls are attached to the floors that they originally belong to (f).

Recognition of floors. Each element is painted with a unique color in order to indicate the identities of the extracted elements. Step (1) select horizontal triangles from the mesh; Step (2) merge connected triangles into larger meshes as set $E$; Step (3) select large meshes from set $E$; step (4) merge close meshes into larger meshes as set $G$; step (5) select large meshes from set $G$, the resulting meshes $H(h)$ constitute the floors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Every element extracted during the recognition stage consists of three parts: floor–vertical connections–floor, namely two floors connected with stairs or slopes (Fig. 2(d)). Combining such elements vertically or horizontally results in a network of floors and connections, i.e., a skeleton of the building (Fig. 2(e)). The TRAMMA program can produce a desired network according to the user-defined topology. In the final stage, the walls are attached to the floors that they originally belong to (Fig. 2(f)).

Although the CBD method we propose has not been tested in extensive building libraries, we worked with a limited repository of building models created in SketchUp (3D Warehouse [22]) for demonstration purposes. The repository includes 26 models of modern architecture, a few samples are shown in Fig. 1. A typical building in the repository consists of 2–4 floors; the rooftops are flat or irregular. In order to process the models in Java, they are transformed from the original SketchUp models to triangular mesh representations. In our application the triangulated mesh serves as the input for recognition and aggregation described below.

3.2. Recognition

The goal of recognition is to group the triangles of the mesh into meaningful categories such as walls and floors. The definition of the categories depends on the desired design operations rather than on a general expertise. In other words, the segmentation provides necessary information for the synthesis (aggregation). The segmentation includes three tasks: recognizing floors, stairs/slopes, and walls.

The recognition of floors is a multi-level clustering process. The goal is to construct $H(h)$, each element $h$ represents one floor. In order to find these floors out of the mesh, the algorithm (1) puts flat triangles into set $D$; (2) groups the triangles in $D$ into set $E(e)$; (3) selects the elements of $E$ as set $F(f)$; (4) groups $f$ into higher-level clusters $G(g)$; (5) selects the elements of $G$ as set $H$. The details are as follows (also illustrated in Fig. 3):

1. Preprocessing the triangles. Grouping all the triangles into four sets according to their normal vectors: $A(t : n(t) < 0.1) B(t : 0.1 \leq n(t) < 0.95) C(t : 0.95 \leq n(t) < 0.995) D(t : n(t) \geq 0.995)$ (horizontal pieces).

$t$: triangle made of three Euclidean vectors (vertices).

$n(t)$: the absolute value of the $z$ component of the normal vector (normalized) of triangle $t$.

2. Making set $E = \{e\}$, $\cap_{e \in E} e = \phi$, $\cup_{e \in E} e = \cup_{t \in D} t$. If share $(t_i, t_j)$ is true they must fall into the same set $e$. 

Fig. 2. The element (floor–connections–floor) is extracted from the sample model (a, b, c, d). Combining the elements as a new skeleton of building according to the user-defined topology (e). The walls are attached to the floors that they originally belong to (f).

Fig. 3. Recognition of floors. Each element is painted with a unique color in order to indicate the identities of the extracted elements. Step (1) select horizontal triangles from the mesh; Step (2) merge connected triangles into larger meshes as set $E$; Step (3) select large meshes from set $E$; step (4) merge close meshes into larger meshes as set $G$; step (5) select large meshes from set $G$, the resulting meshes $H(h)$ constitute the floors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
Table 2
Performance of the algorithm for recognizing floors.

<table>
<thead>
<tr>
<th>Model</th>
<th>Model 2</th>
<th>Model 3</th>
<th>Model 4</th>
<th>Model 5</th>
<th>Model 6</th>
<th>Model 7</th>
<th>Model 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall (%)</td>
<td>97.68</td>
<td>98.15</td>
<td>94.34</td>
<td>95.31</td>
<td>99.34</td>
<td>98.66</td>
<td>96.65</td>
</tr>
<tr>
<td>Precision (%)</td>
<td>100.00</td>
<td>98.98</td>
<td>96.59</td>
<td>99.89</td>
<td>99.07</td>
<td>98.74</td>
<td>100.00</td>
</tr>
<tr>
<td>Number of triangles of the model</td>
<td>11,321</td>
<td>10,017</td>
<td>27,030</td>
<td>13,916</td>
<td>13,842</td>
<td>5780</td>
<td>19,710</td>
</tr>
<tr>
<td>Time (ms)</td>
<td>321</td>
<td>183</td>
<td>632</td>
<td>393</td>
<td>368</td>
<td>207</td>
<td>424</td>
</tr>
</tbody>
</table>

\(^{a}\) Recall = \(\text{area}(P \cap Q)/\text{area}(P)\); \(P\): the set of triangles of the actual floors. \(Q\): the set of triangles of the recognized floors.

\(^{b}\) Precision = \(\text{area}(P \cap Q)/\text{area}(Q)\).

\(^{c}\) Processor: 2.5 GHz Intel Core i5, Memory: 4 GB 1067 MHz DDR3, Java 1.6, no multithread implementation.

c: a set of triangles, actually a continuous face made of triangles.
share\((t_i, t_j)\): test if two triangles share two vertices.
3. Constructing set \(F = \{f : f \in E, \text{area}(f) > a_1\}\).
   \(f\): a set of triangles, actually a continuous face made of triangles.
   \(a_1\): a threshold, constant.
   \text{area}(f): the sum of the areas of the triangles in set \(f\).
4. Calculating set \(G = \{g\}\). \(\bigcap_{g \in G} g = \emptyset \), \(\bigcup_{g \in G} g = \bigcup_{f \in F} f\). If close\((f_i, f_j)\) is true, all the triangles in \(f_i\) and in \(f_j\) must fall into the same set \(g\).
   \text{close}(f_i, f_j) = \exists p, t_q : \text{closeZ}(t_p, t_q) \land \text{closeXY}(t_p, t_q),
   t_p \in f_i, t_q \in f_j.
\(g\): a set of triangles.
   \text{closeZ}(t_p, t_q): true if the minimum difference between the \(z\) components of the vertices of the two triangles are below a threshold; otherwise false.
   \text{closeXY}(t_p, t_q): true if the minimum distance between the vertices of the two triangles are below a threshold, or the two triangles overlap in the XY plane; otherwise false.
   5. Making set \(H = \{h : h \in G, \text{area}(h) > a_2\}\).
   \(h\): a set of triangles, actually a floor.
   \(a_2\): a threshold, constant.

The set \(H\) is the set of floors (Fig. 4, top). The above algorithm is designed for handling ill constructed meshes and modeling variations (e.g., a floor can be modeled either as a single surface or as two surfaces). The performance of the algorithm for recognizing floors is reported in Table 2.

The stairs/slopes are the vertical components connecting a pair of floors (Fig. 4 middle). Therefore, the stairs/slopes are searched between any pair of two floors in each model. Similarly to the recognition of the floor, the recognition of stairs/slopes is also a multi-level clustering process. The detailed procedure for floor \(h_i\) and floor \(h_j\) is as follows:

1. Constructing set \(K = \{t : t \notin \bigcup_{h \in H} h, t \in C \cup D, \min z(t) > h_i, \max z(t) < h_j\}\).
   \(\min z(t)\): the minimum \(z\) coordinate of the triangle \(t\).
   \(\max z(t)\): the maximum \(z\) coordinate of the triangle \(t\).
   \(h_i\): the upper floor.
   \(h_j\): the lower floor.
2. Calculating set \(M = \{m\}\), \(\bigcap_{m \in M} m = \emptyset\), \(\bigcup_{m \in M} m = \bigcup_{t \in K} t\).
   If close\((t_i, t_j)\) is true, they must fall into the same set \(m\).
   \text{close}(t_i, t_j) = \text{share}(t_i, t_j) \lor \text{closeZ}(t_i, t_j) \lor \text{closeXY}(t_i, t_j)

3. Making set \(N = \{n : n \in M, \text{area}(n) > a_3\}\) \(n\): a set of triangles.
   \(a_3\): a threshold, constant.
4. Making set \(S = \{s : s \in N, \exists t : c(t, h_i) \land c(t, h_j), t \in s\}\).
   \text{c}(t, h_i) = \exists \emptyset : \text{closeZ}(t, t_j) \land \text{closeXY}(t, t_i),
   t_i \in h_i
   \text{c}(t, h_j) = \text{closeXY}(t_j, t_i),
   t_i \in h_i
   \text{s}: a set of triangles.

As a result, there could be a set of stairs/slopes (set \(S\)) connecting a pair of floors, i.e., it is possible to have more than one walkable path between two floors. Finally, a pair of floors and the corresponding set of stairs/slopes make a topological element \(p = \{h_i, h_j, S\}\) (see Fig. 2(a)-(d)). These elements extracted from the input models institute the pool \(P = \{p\}\). Besides, the walls (Fig. 4 bottom) are recognized after the recognition of floors and stairs/slopes. Each floor saves a reference to its corresponding walls (if there is any).

3.3. Aggregation

The aggregation procedure combines the extracted elements \(p\) into new compositions, as illustrated in Fig. 2. The desired topology is manually defined by the user; the TRAMMA program stacks the elements according to the defined topology automatically. The elements could be combined vertically (Figs. 2 and 7) or horizontally (Fig. 9). The topology of multi-story buildings is represented by graphs. Graphs have been widely employed to represent
the topology in 2D floor plans, both in analysis [23] and synthesis [19,24]. In this program each floor is modeled as a node, the vertical connections (stairs or slopes) between a pair of floors as a link/edge (Fig. 5). In the graph, a topological element \( p \) is represented by two nodes with one link. Two jointed nodes (e.g. the node \( B, C \) in Fig. 5 left) denote that the two floors connect horizontally. To some degree the user-defined topology is arbitrary; three topologies are tested in the application: LINEAR, CIRCLE, and BRANCH (Fig. 5).

After the topology is specified, the program randomly retrieves elements from the library (set \( P \)) and combines them to satisfy the topology. For example, if we are going to implement the LINEAR topology, the procedures are:

1. retrieve one element from the library as component \( AB \) in LINEAR topology
2. FOR each connection in LINEAR topology //1st: \( AB \) to CD; 2nd: CD to EF
   WHILE (error function \( f(v) \) is TRUE) //1st-level trial
      retrieve a new element \( p_i \) from the library
   WHILE (error function \( f(v) \) is TRUE) //2st-level trial
      move and rotate element \( p_i \) by vector \( v \)
   END
END

The error function:

\[
f(v) = \text{overlap}(v, l(p_i), h(p_i)) \land \text{as}(v, s(p_j), l(p_i)) \lor \text{as}(v, s(p_j), h(p_i))
\]

\( p_i \): the upper element, \( p_j \): the lower element (illustrated in Fig. 6).

\( l(p) \): the lower floor of element \( p \)

\( h(p) \): the upper floor of element \( p \)

\( s(p) \): the set of stairs/slopes of element \( p \)

\( v \): the 2D transformation vector of three components: angle (rotation), \( x \) and \( y \) (translation).

Actually the error function (4) contains two sub-error functions, \( \text{overlap}(v, h_i, h_j) \) tests if floor \( h_j \) overlaps floor \( h_i \) (translated by \( v \)), see Fig. 6(a); \( \text{as}(v, s, h) \) tests whether stairs \( s \) keeps away from the obstacle from floor \( h \) (translated by \( v \)), see Fig. 6(b).

\[
\text{overlap}(v, h_i, h_j) = \exists t_p, t_q: \text{overlap}(t_p, t_q), t_p \in h_i, t_q \in h_j
\]

\[
\text{as}(v, s, h) = \exists t_p, t_q: \text{intersect}(t_p, t_q), t_p \in s, t_q \in h
\]

\( t_p \): the 2D transformation vector of \( t_p \) (translation), \( t_q \): the 2D transformation vector of \( t_q \) (translation).

The major task of aggregation is error elimination. The errors occur in the vertical/horizontal connection between the floors (Fig. 6(a) illustrates a valid connection between floors), and in the collisions between stairs/slopes and floors (Fig. 6(b) illustrates an invalid connection with collision). The error functions for different topologies (Fig. 5) are slightly different. The pseudocode of the aggregation process indicates that the process is a two-level “trial and error” algorithm. Therefore, the performance of the algorithm can be partially evaluated by how many trials it must take to eliminate the errors (Table 3).
Table 3  
Performance of aggregation algorithm.

<table>
<thead>
<tr>
<th>Topology</th>
<th>Average number of trials until success (1st level trial&lt;sup&gt;a&lt;/sup&gt;)</th>
<th>Average number of trials until success (2nd level trial&lt;sup&gt;b&lt;/sup&gt;)</th>
<th>Time to completion&lt;sup&gt;c&lt;/sup&gt; (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st connection</td>
<td>2nd connection</td>
<td>1st connection</td>
</tr>
<tr>
<td>LINEAR</td>
<td>1.30</td>
<td>1.30</td>
<td>39.05</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>1.25</td>
<td>2.15</td>
<td>31.50</td>
</tr>
<tr>
<td>BRANCH</td>
<td>1.05</td>
<td>4.95</td>
<td>9.50</td>
</tr>
</tbody>
</table>

<sup>a</sup> Retrieve new elements from the library of extracted elements.
<sup>b</sup> Adjust the position/orientation/scale of the retrieved element.
<sup>c</sup> Processor: 2.5 GHz Intel Core i5, Memory: 4 GB 1067 MHz DDR3, Java 1.6, no multithread implementation.

4. Results and future work

The three topologies in Section 3.3 can be quickly achieved by the TRAMMA program (usually after dozens of iterations). The information acquired in the recognition phase enables the aggregation process to make valid compositions. It implements the general task of CBD: “creating a new design solution by combining and/or adapting previous design solution(s)” [25]. Figs. 7–9 show two results for each topology (Fig. 5) respectively. The results imply that a small collection of samples (26 cases) can lead to a wide range of forms. Some results surprise us because they present the 3D geometric features of different cases within a novel composition. For architects, some generated buildings appear as manually designed. Thus, the program could help the architects to explore new designs with a few cases.

The complexity of the recognition algorithm is between $O(n)$ and $O(n^2)$. For instance, the complexity of steps 1, 3, 5 of floor recognition is $O(n)$; that of steps 2, 4 is $O(n^2)$. The synthesis algorithm has a linear complexity. The program could be inefficient with large repository or with complex models. Besides, there are some drawbacks in the generated designs from a point of view of architect. For instance, some generated buildings lack structural feasibility. Moreover, some interior spaces could be chaotic because the program focuses on the topology between floors instead of the topology within a single floor. This problem could be solved by manual manipulation or by other synthesis programs for single floor.

The future work can be developed in several directions. First, the category of semantic structures such as floors, walls, and stairs could be extended so that the method we propose could be more widely applicable to architectural design. Second, the recognition–aggregation process could be integrated with the optimization algorithm to consider more criteria. Third, it is interesting to integrate this program into urban design. An experimental scenario has been tested: first, creating a virtual construction site (380 m × 380 m), with several roads dividing the site into irregular parcels; and second, employing a 2D packing algorithm in order to place the generated buildings (according to their convex hulls) in the parcels compactly (Fig. 10). It is possible to add more constraints to the packing algorithm to have more sophisticated controls on the planning in the future work.

5. Conclusions

Briefly, it is rewarding to regard CBD not only as the formalization of previous knowledge but also as an opportunity to
create new designs and to stimulate new reflections. In TRAMMA program, some synthesized designs have surprised us by unexpected combination of previous cases, e.g., by an unprecedented mixture of different kinds of rooftops. The results suggest that CBD could be a middle ground between traditional use of CAD software and automatic spatial synthesis [19,24,26,27]. Commercial modeling software provides a convenient platform for modeling and storing cases; spatial synthesis emphasizes the early stages of design, especially the automatic production of spatial compositions via formal operations. Our TRAMMA program functions independently, however, it is possible to integrate the program with CAD tools in the future. Such kind of CBD programs can directly take the models in CAD software as samples and then employ a generative process to construct new compositions. In this scenario, CBD can lead to new designs without the extensive manual interventions in traditional modeling, but still benefit from the increasing number of models based on common modeling software.

Acknowledgments

Prof. Dr. Ludger Hovestadt at the chair of Computer-Aided Architectural Design ETH Zürich has given valuable advices on this work. The SketchUp models are downloaded from Trimble 3D Warehouse (2013).

References