Vis Comput (2016) 32:151–166
DOI 10.1007/s00371-015-1061-7

ORIGINAL ARTICLE

Procedure-based component and architecture modeling from a single image

Fei Hou · Hong Qin · Yue Qi

Published online: 13 February 2015

Abstract This paper advocates a new component-aware framework to reconstruct 3D architecture from a single image. Different from existing work, our motivation is to obtain a complete set of semantically correct 3D architectural components, which enables part reusability towards rapid model reproduction and facilitates model variation. The core of our system is a novel algorithm to adaptively segment repeated curved stripes (e.g., roof tiles, building floors) into individual elements, based on which 3D dimensions as well as architectural components are derived from a single image. Specially for Chinese architectures, we further devise an interactive method to identify outer columns based on user-specified inner columns. Finally, 3D components are generated and shape rules are derived, from which the buildings and their variants are constructed. Our new component-aware framework minimizes the use of data resource (i.e., one single image) and emphasizes component utility during rapid 3D architecture reproduction by advocating a component-aware approach.

Keywords Architectural modeling · Image-based modeling · Procedural modeling

Electronic supplementary material The online version of this article (doi:10.1007/s00371-015-1061-7) contains supplementary material, which is available to authorized users.

F. Hou (✉) · Y. Qi
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
e-mail: houfei@buaa.edu.cn

Y. Qi
e-mail: qy@vrlab.buaa.edu.cn

F. Hou
School of Computer Engineering, Nanyang Technological University, Singapore, Singapore

H. Qin
Department of Computer Science, New York University (SUNY Stony Brook), New York, USA
e-mail: qin@cs.sunysb.edu

1 Introduction and overview

Rapid creation of 3D architectural models with functionalities is of great significance for interactive 3D graphics, heritage preservation, urban simulation at city-scale, game/film production, etc. Although architectural models can be reconstructed from various data sources, such as 3D scanner data [1], airborne data [2] and multiple street-view images [3], such data sources heavily rely on either tremendous manpower or specialized equipment of high cost. Moreover, the conventional image-based modeling methods [4] lack of well-defined components and functional contents, which may prevent the model variation and further reuse. Recently, much work [5,6] is devoted to analyzing planar building façade structures. However, very few work concentrates on structural analysis of Chinese architectures or curved building façades. We observe that the curved alternately repeated patterns, such as roof concave and convex tiles (Fig. 1) or building floors (Fig. 3a), are pervasive in both ancient and a few modern architectures, which afford important clues for 3D dimension recovering and component generation/reuse.

In this paper, we devise a novel unified framework to build a complete set of semantically correct 3D architectural components as well as 3D models for Chinese architectures and curved façades of modern architectures. Our generated models are composed of components and can be controlled by procedural rules with a suite of changes. Our motivation is

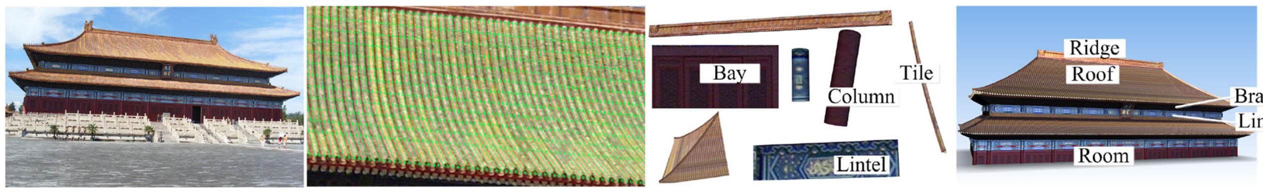


Fig. 1 Taking as input a single Imperial Ancestral Temple image, our system reconstructs the 3D architectural components and generates the Imperial Ancestral Temple model procedurally

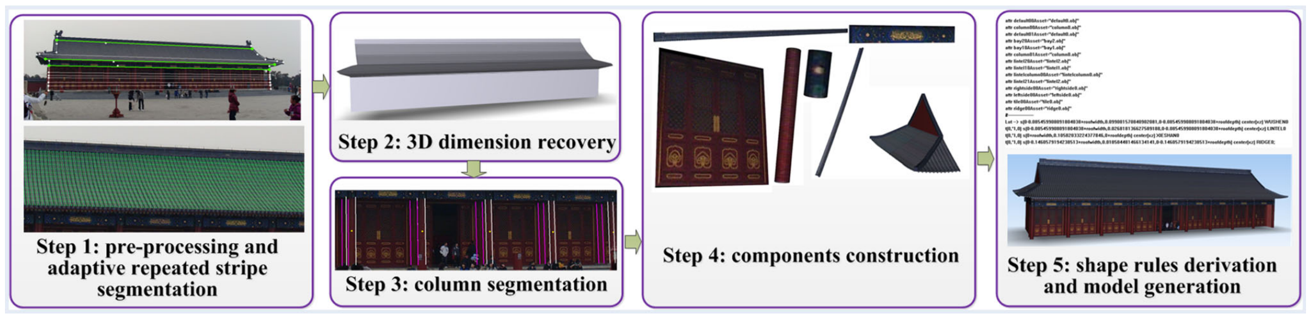

Step 1: pre-processing and adaptive repeated stripe segmentation
Step 2: 3D dimension recovery
Step 3: column segmentation
Step 4: components construction
Step 5: shape rules derivation and model generation

Fig. 2 The pipeline of our system. Step 1: After pre-processing, repeated stripes are adaptively segmented. Step 2: The 3D dimensions are recovered. Step 3: For Chinese architectures, the columns are segmented. Step 4: The 3D components are generated. Step 5: The shape rules are derived and the 3D models are generated

to advance the structural analysis of Chinese architectures and curved façades with fewer interactions. In particular, we detail a novel repetition analysis algorithm to segment curved stripes into individual ones (e.g., roof tiles). After calibrating the camera based on these stripes' directions, for curved modern building façades, we present an algorithm to extract the curved surfaces based on their grid patterns. Specially for Chinese architectures, we present a method to identify the outer columns with the help of user-specified inner columns locations to segment and complete the occluded bays as well as the lintels. Based on our originally developed segmentation and 3D reconstruction, we afford the construction of 3D components and derive the shape rules, from which the 3D architectural models along with their possible variants are generated.

Contributions: Our primary contribution is that we design an alternately repeated pattern analysis algorithm to segment curved stripes adaptively (e.g., roof tiles and building floors), based on which the 3D dimensions as well as architectural components can be derived from a single image. Our secondary contribution is that we develop a complete component-aware architecture modeling system from a single image.

System pipeline: Figure 2 illustrates the pipeline of our system. First, after pre-processing (Sect. 3), we detail the curved alternately repeated stripe segmentation algorithm (Sect. 4). Second, we recover the 3D dimensions of the buildings based on the curved stripe directions, where the curved façade of modern architecture is refined by a non-linear optimization process (Sect. 5). Third, for Chinese architectures, we specially identify their columns, which determine the room structures (Sect. 6). Fourth, we generate the 3D architectural components and grammar rules, from which the 3D models and their variants are generated (Sect. 7).

2 Related work and our motivation

Image-based modeling and procedural modeling (to which this paper is of relevance) have been active for many years. Musialski [7] presented a comprehensive overview of urban reconstruction. We briefly review the most related work in the following categories, which naturally lead to our rationales.

Procedural architecture modeling: Procedural modeling is a powerful method to generate 3D models automatically. Wonka et al. [8] introduced a powerful split grammar for architectural procedural modeling. Müller et al. [9], while extending the split grammar, also introduced the CGA shape grammar capable of generating massive urban models. To simplify the rule editing, Lipp et al. [10] introduced a visual editing system with direct local control for procedural architectures. Talton et al. [11] tried to control the L-system production based on maximum a posteriori estimation. Hou et al. [12] constructed Chinese architectures procedurally from façade drawing. Kelly et al. [13] presented an interactive framework to extrude buildings from footprints. In addition to the rule-based procedural modeling, Merrell et al. [14] presented an example-based model synthesis method. Bokeloh et al. [15] synthesized new models based on symmetry analysis of the example models. However, these example-based model synthesis methods are designed to generate various models with certain style, which tend to be difficult to precisely control their results due to random variables used for various model generation. Instead, we present an inverse procedural modeling approach to extract rules from a single image automatically.

Image-based architecture modeling: Image-based architectural modeling has been extensively studied. Goesele et al. [16] and Agarwal et al. [17] studied multi-view stereo using Internet images. Pollefeys et al. [18] presented a real-time system for 3D urban reconstruction from video. However, the general image-based modeling approaches are becoming less attractive because of their instability in handling noise. To ameliorate, through adding certain structural information, many alternative approaches have been proposed to deal with more specific problems. Sinha et al. [19] presented an interactive system to build piecewise planar buildings preserving sharp edges and corners. Xiao et al. [3,20] built street-side buildings from street-view image sequence focusing on segmenting the images into architectural elements. Zhao et al. [21] proposed an approach to parse street-view registered images into architectural units for large-scale city modeling. Vanegas et al. [22] integrated image-based modeling with procedural modeling to model modern Manhattan-pattern buildings. Ceylan et al. [23] presented an optimization framework to extract the repeated elements of urban façade as well as recovering the 3D geometry simultaneously. These work mainly concentrates on modern building constructions from multiple views and some of them are incapable of constructing component-aware models for further applications.

Single-view architectural modeling: Modeling from a single image is an ill-posed problem which needs additional clues to recover 3D information. Much work has focused on recovering 3D shapes based on constraints such as symmetry [24], parallelism and orthogonality [25] or parallelepiped [26]. Based on symmetry, Jiang et al. [27] presented a framework to build symmetric models, such as Chinese architectures, from a single image. They focused on camera calibration from a frustum but users have to mark all the architectural structures explicitly via interaction. Wu et al. [28] presented an approach for dense reconstruction from a single view of repeated scene structure. However, these approaches lack of component information which could potentially contribute to model variation. Instead, we propose to generate architectural components, and then construct variable 3D models procedurally. In the aspect of sketch-based modeling, Chen et al. [29] designed a system to convert freehand sketch to realistically looking 2.5D architectural models. Prasad et al. [30] reconstructed curved surface from a single view. Zhang et al. [31] unwrapped generalized cylindrical surfaces based on low-rank decomposition, but their method is inadequate to segment the repeated textures into individual stripes.

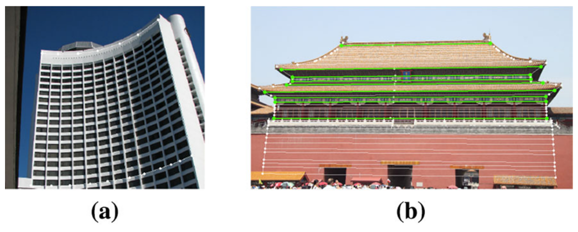


Fig. 3 Interactive façade segmentation. **a** A hotel model. The user first draws the façade contour. **b** For ancient Chinese buildings, the user specifies necessary lines (*colored in green*) to segment the façade horizontally, where the *white lines* are candidates with larger gradients

We not only unwrap the distorted textures but also segment them into individual repeated components (e.g., roof tiles and building floors). Furthermore, they formulate the surface by a polynomial model, but our method isn't restricted by this condition.

Repetition analysis: Liu et al. [32] and Mitra et al. [33] presented comprehensive reviews of symmetry analysis in image and geometry, respectively. In the aspect of architectural modeling, Müller et al. [34] and Musialski et al. [5] detected symmetric or partially symmetric elements in façade image to reconstruct façade procedurally. Wu et al. [35] proposed a framework to analyze large repeated structures in urban scenes. Shen et al. [36] presented an adaptive urban façade partition algorithm. However, to split irregular façades, this algorithm does not explore regular repetitions sufficiently in the first splitting, which would make it unstable. Our strategy is rather different. We explore irregular repetitions based on regular repetitions.

3 User interactions

Some interactions are needed to assist the model reconstruction. Given an image, the user draws the façade contour (Fig. 3). The vertical (modern building) or horizontal (Chinese architectures) vanishing point is computed as the intersection of the two vertical (horizontal) segments of the contour. For curved façade, just the contour is needed, more interactions are needed for Chinese architecture as detailed below.

As shown in Fig. 3b, the user should also provide an initial vertical segmentation (denoted as green lines). The system first computes some horizontal candidate lines (see Fig. 3b) passing through the vanishing point with larger gradients and then the user should select the necessary lines from candidates or draw additional lines to segment the façade into semantic partitions (Fig. 3b) and identify their types, such as ridge, roof, bracket set lintel, room, platform, etc.

Furthermore, the user should also specify the locations of columns as well as the width of one of the columns for column segmentation and reconstruction (see Sect. 6 for details). Please also refer to the affiliated video for interaction details.

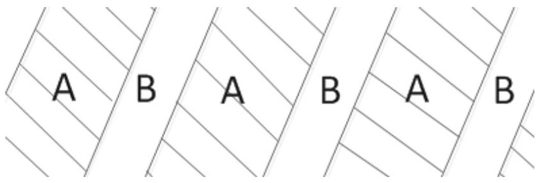


Fig. 4 An alternately repeated pattern $\{A, B, A, B, A, B, \ldots\}$. The elements A and B appear interleaved

4 Adaptive segmentation of alternately repeated stripes

Repetitions are pervasive in both ancient and modern buildings, hence playing a critical role in architectural modeling process. The common characteristic of roof tiles (Fig. 10) of Chinese architectures and floors (Fig. 11c) of modern buildings is that they essentially present curved stripe shapes (in this paper, the curved stripe refers to individual roof tile and building floor) and alternately repeated patterns, i.e., a repeated sequence of $\{A, B, A, B, A, B, \ldots\}$ consisting of alternately repeated elements (see Fig. 4). The A's and B's may either represent concave tiles and convex tiles, respectively, or window floors and wall floors, respectively, both of which are alternatively repeated. The roof tiles are alternately repeated regularly, i.e., the repeated stripes cover the whole surface, while some building floors, such as Fig. 29a, are alternatively repeated irregularly due to the existence of the purple dashed regions, i.e., they cover part of the surface. We now present an automatic algorithm to segment the regular or irregular curved stripes into individual ones.

The algorithm consists of two steps: direction evaluation (Sect. 4.1) and stripe segmentation (Sect. 4.2). In the first step we evaluate the curved stripe direction field based on a robust RANSAC procedure. The second step is the core of our method, where the curved stripes are segmented based on their repetitions.

4.1 Stripe direction field

We first evaluate the stripe directions which are a prerequisite for stripe segmentation. On the projected 2D image plane, the curved stripes can be regarded as a vector field by associating the stripe tangent direction with every pixel. We present a method to evaluate this direction field which is also used later for stripe segmentation and camera calibration.

Evaluating direction at a point: Although the stripes are curved, they are nearly straight locally, as shown in Fig. 5a. We regard the stripes as several alternately repeated straight lines locally, whose local directions are given by the direction of these straight lines. We observe that the tiles are locally bilateral symmetric with respect to the axis orthogonal to its direction. We find the direction of this axis by iteratively enumerating from 0° to 180° at the pixel to minimize the

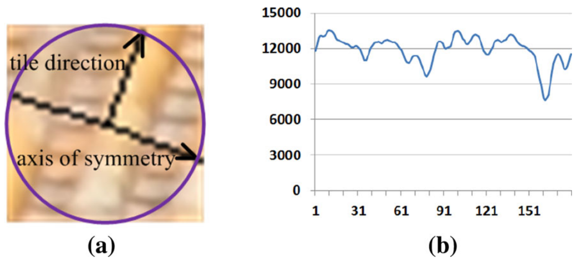


Fig. 5 **a** An enlarged circle neighborhood with radius of 20 pixels, and its symmetric axis and the tile direction at the center. **b** The sum of intensity differences with respect to different angles of symmetric axes

intensity difference between corresponding symmetric pixels. Fig. 5b shows the sum of intensity differences for the patch in Fig. 5a with respect to all possible angles, and the optimal axis direction is shown in Fig. 5a.

Evaluating directions on the surface: As shown in Fig. 6, in the image, we evenly sample several (10 in the example) horizontal lines (*sample line*) passing through the horizontal vanishing point (e.g., the white lines in Fig. 6). In 3D space, the tiles are parallel, so their tangent directions are parallel on each sample line. Therefore in 2D image space, the tile directions on each sample line should be intersecting at a vanishing point. Meanwhile, the vanishing points on different sample lines should be collinear. And for each sample line, after computing the tile directions at several (50 in the example) sample points, which are sampled evenly on each sample line, we vote for their corresponding vanishing points using RANSAC followed by computing the vanishing line using RANSAC, where the incorrect vanishing points are filtered out. The result is shown in Fig. 6. For the building in Fig. 11, we first split the façade automatically by the orange lines in Fig. 11c using the method detailed in Sect. 4.2, and then, as shown in Fig. 11a, sample the lines vertically between adjacent orange lines to avoid vague symmetry when evaluating the floor directions. The two lost columns (orange columns) are filtered out due to the fact of conflicting to the vanishing line in the RANSAC procedure.

4.2 Adaptive stripe segmentation

Being aware of the stripe directions, we devise an heuristic algorithm to adaptively segment the regular (irregular) repeated stripes into individual stripes to facilitate 3D component generation. It may be noted that whether it is regular or not is specified by the user. A naive method is to extract all the repetitive sequences to select the most dominated ones. Instead we present an intelligent algorithm to extract the dominated repeated sequences efficiently while bypassing unnecessary searchings. The critical idea behind this algorithm is that we evaluate the repetition patterns hierarchically in different scales and then combine them to vote for the best

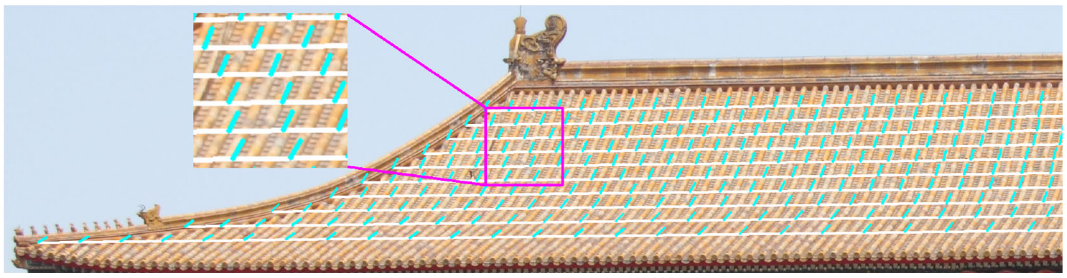


Fig. 6 The tile direction field. The lines (*colored in cyan*) indicate the tile directions at the sample points on the sample horizontal lines (*colored in white*)

ones to split the stripes. We extract candidate splitter subsequences (Sect. 4.2.1) and then combine them on every sample line (Sect. 4.2.2). Finally, we combine them adaptively on the entire surface to split the whole stripes (Sect. 4.2.3).

The splitters are a sequence of points to split a line into alternately repeated elements. On each sample line, we sample an ordered sequence of points $S = \{p_1, p_2, \ldots, p_n\}$ as splitter candidates (i.e., the final splitters are a subset of S), with spacing of one pixel. Neither the number of final splitters nor the spacings between splitters are known in advance. We only require that the splitters should be alternately repeated. Given a subsequence $S' \subset S$, $S' = \{p'_1, p'_2, \ldots, p'_n\}$, the set of continuous alternately repeated subsequences of S' is denoted by $\mathcal{R}(S')$. A subsequence $l \in \mathcal{R}(S')$, if the following conditions hold:

(a) (continuous) $l = \{p'_k, p'_{k+1}, \ldots, p'_{k+m}\}$,
(b) (two alternately repeated)

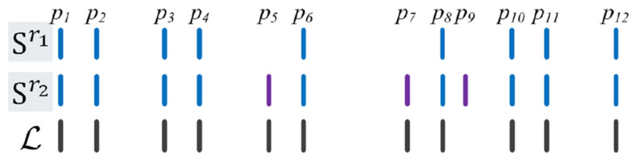
$$\frac{\min(\|p'_i - p'_{i+1}\|, \|p'_{i+2} - p'_{i+3}\|)}{\max(\|p'_i - p'_{i+1}\|, \|p'_{i+2} - p'_{i+3}\|)} \geq \tau_1,$$
$$k \leq i \leq k + m - 3.$$

In the irregularly repeated case, the textures between intervals (p'_i, p'_{i+1}) and (p'_{i+2}, p'_{i+3}) should also satisfy

$$\frac{\sum_{x,y}(I(x,y) - I'(x,y))^2}{\sqrt{\sum_{x,y} I(x,y)^2 \cdot \sum_{x,y} I'(x,y)^2}} < \tau_2$$

(c) (longest) Neither the subsequence $\{p'_{k-1}, p'_k, p'_{k+1}, \ldots, p'_{k+m}\}$ nor $\{p'_k, p'_{k+1}, \ldots, p'_{k+m+1}\}$ satisfies condition (b), where m is a positive integer, and I and I' indicate image intensity for (p'_i, p'_{i+1}) and (p'_{i+2}, p'_{i+3}), respectively.

In our experiments, we set $\tau_1 = 0.75$ and $\tau_2 = 0.045$ or 0.05. In the example of Fig. 7, $\mathcal{R}(S^{r_1}) = \{\{p_1 \ldots, p_4\}, \{p_3, p_4, p_6\}, \{p_4, p_6, p_8\}, \{p_6, p_8, p_{10}\}, \{p_8, p_{10}, p_{11}, p_{12}\}\}$, $\mathcal{R}(S^{r_2}) = \{\{p_1, \ldots, p_6\}, \{p_5, \ldots, p_8\}, \{p_7, p_8, p_9\}, \{p_8, p_9, p_{10}\}, \{p_9, p_{10}, p_{11}\}, \{p_{10}, p_{11}, p_{12}\}\}$,



p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12}
S^{r_1}
S^{r_2}
$\mathcal{L}$

Fig. 7 The top two subsequences represent the level r_1 and level r_2 subsequences of a sequence. The third sequence is their combined sequence. S^{r_1} is a subset of S^{r_2} and the *purple lines* are the newly added lines in S^{r_2}

and both satisfy the property of two alternately repeated. It may be noted that the subsequences may overlap, e.g., $\{p_1, \ldots, p_4\}$ and $\{p_3, p_4, p_6\}$ appear in both sets.

4.2.1 $\mathcal{R}(S)$ extraction on a line

On each sample line, we first extract a set of continuous alternately repeated subsequences $\mathcal{R}(S)$ as candidates for further combination. Given candidate splitters S, the expected splitters may skip several elements of S (that are not yet known at this very moment). We have observed that a point with larger derivative with respect to the direction orthogonal to its tangent direction is more likely to be a splitter. For every candidate p_i, we denote its potential to be a splitter by $\mathcal{P}(p_i)$ and evaluate it as follows: let the sum of differences between symmetric pixels with respect to the stripe direction at p_i be $d(p_i)$. $\mathcal{P}(p_i) = r$, *iff.* in the neighborhood $\{p_{i-r}, \ldots, p_i, \ldots, p_{i+r}\}$, $d(p_i) > d(p_k)$, $k \neq i, i - r \leq k \leq i + r$, and $d(p_i) < d(p_{i-r-1})$ or $d(p_i) < d(p_{i+r+1})$. That is, $d(p_i)$ is the largest in its r neighborhood. As shown in Fig. 8, the potentials of two extrema are marked, whose potentials are the half length of the marked segments. The potential of non-extremum is 0.

We analyze the repetitive pattern points $p_i \in S$ hierarchically based on their potentials to reveal all possible potential repeated patterns contained in it. The sample points in S are classified into different levels of subsequences $S^r = \{p_i\}$ where $p_i \in S^r$ *iff.* $\mathcal{P}(p_i) \geq r$. For example, in Fig. 7, the sequence is classified into two levels S^{r_1} and S^{r_2} ($r_1 > r_2$), thus $S^{r_1} \subset S^{r_2}$.

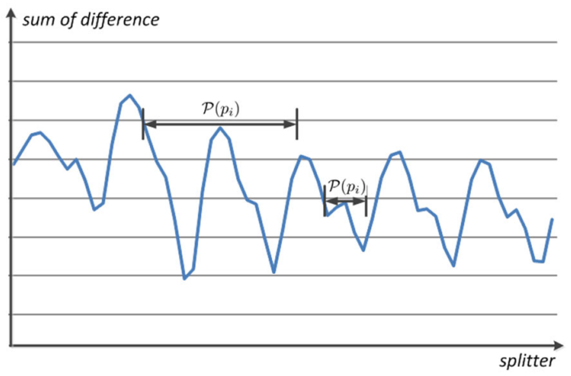


Fig. 8 The potential of splitters. The potentials of two extrema are marked, whose potentials are defined as the half length of the marked segments. The potential of non-extremum is 0

Definition 1 Suppose l_1 and l_2 are two sequences. If the union sequence of l_1 and l_2 is also two alternately repeated, l_1 and l_2 are called mergeable. In Fig. 7, the subsequences $\{p_1, \ldots, p_4\}$ and $\{p_1, \ldots, p_6\}$ can be merged to form $\{p_1, \ldots, p_6\}$.

Given S, we extract $\mathcal{R}(S)$ repeatedly by varying r from maximum (set to 16 for Chinese architecture and 20 for curved façade) to 1. In each loop, first, we try to extend the subsequences in $\mathcal{R}(S)$ by S^r. For every element $l_i \in \mathcal{R}(S)$, we expand l_i by S^r. That is to say, we add the splitters into S^r which are not confined by the first and last splitters of l_i to form l_i', so that the repeated pattern of l_i is reserved. If a subsequence $s \in \mathcal{R}(l_i')$ satisfies $l_i \subset s$ (meaning that l_i is a continuous subsequence of s), then l_i is substituted by s. For example, in Fig. 7, suppose $l_i = \{p_8, p_{10} p_{11}, p_{12}\}$ and it is expanded by S^{r_2} to form $l_i' = \{p_1 \ldots, p_8, p_{10}, p_{11}, p_{12}\}$. We find that $l_i \subset \{p_7, p_8, p_{10}, p_{11}, p_{12}\} \in \mathcal{R}(l_i')$ so l_i is substituted by $\{p_7, p_8, p_{10}, p_{11}, p_{12}\}$. Second, we compute $\mathcal{R}(S^r)$, and for every subsequence $s \in \mathcal{R}(S^r)$, if no $l_i \in \mathcal{R}(S)$ s.t. $l_i \supseteq s$, then s is inserted into $\mathcal{R}(S)$. For example, in Fig. 7, $\{p_1, \ldots, p_6\} \in \mathcal{R}(S^{r_2})$ is inserted into $\mathcal{R}(S)$, since it is not contained in $\mathcal{R}(S^{r_1})$. After a few iterations, $\mathcal{R}(S)$ will contain all the potential splitter subsequences which might be heavily overlapped. Third, we merge all the subsequences in $\mathcal{R}(S)$ that are mergeable. For example, in Fig. 7, $\mathcal{R}(S) = \{\{p_1, \ldots, p_6\}, \{p_4, p_6, p_8\}, \{p_7, p_8, p_{10}, p_{11}, p_{12}\}, \{p_6, p_8, p_{10}\}, \{p_5, \ldots, p_8\}, \{p_3, p_4, p_6\}, \{p_9, p_{10}, p_{11}\}, \{p_8, p_9, p_{10}\}, \{p_7, p_8, p_9\}\}$. The above idea and procedure are documented in Algorithm 1 in details.

4.2.2 Stripe segmentation on a line

Even though $\mathcal{R}(S)$ contains the potential splitter subsequences, they are redundant and heavily overlapped with different repetitive patterns. After obtaining $\mathcal{R}(S)$, on each sample line, we try to extract a set of consistent and disjoint subsequence groups, to segment the stripes on every sample line independently. To discuss how to combine subsequences, we first present some definitions as follows.

Algorithm 1: Repeated subsequence extraction

```
foreach sample line do
    $\mathcal{R}(S) \leftarrow \Phi$;
    for $r$=maximum to 1 do
        foreach $l_i \in \mathcal{R}(S)$ do
            foreach $s \in \mathcal{R}(l_i$ expanded by $S^r)$ do
                if $l_i \subset s$ then
                    $l_i \leftarrow s$;
        foreach $s \in \mathcal{R}(S^r)$ do
            if no $l_i \in \mathcal{R}(S)$ s.t. $l_i \supseteq s$ then
                insert $s$ into $\mathcal{R}(S)$;
    merge all the mergeable subsequences in $\mathcal{R}(S)$;
```

Definition 2 Let l_1 and l_2 be two alternately repeated subsequences. If l_1 and l_2 are not mergeable and the spacings and textures between l_1 and l_2 are both similar, which are measured by average spacings of at most 7 adjacent splitters of l_1 and l_2 and adjacent texture similarity characterized by the two alternately repeated condition, l_1 and l_2 are called compatible. Let G_1 and G_2 be two sets of subsequences. If all pairs of $l_i \in G_1$ and $l_k \in G_2$ are either mergeable or compatible, G_1 and G_2 are called mergeable. If G_1 and G_2 are not mergeable and all pairs of $l_i \in G_1$ and $l_k \in G_2$ are not overlapped, G_1 and G_2 are called compatible.

The subsequences in $\mathcal{R}$ are first classified into different groups according to their compatibility (grouping step), and second, certain groups are extracted and combined to split the line (combination step).

In the grouping step, the subsequences in $l_i \in \mathcal{R}(S)$ are sorted in descending order according to $\text{len}(l_i) \times num(l_i)$, where $\text{len}(l_i)$ is the distance between the first and last splitters of l_i and $\text{num}(l_i)$ is the number of splitters in l_i, since longer and denser subsequences are more likely to be appropriate ones. The subsequences in $\mathcal{R}(S)$ are processed to classify them into different groups $\mathcal{G} = \{G_i\}$, satisfying $\bigcup_i G_i = \mathcal{R}(S)$. If $l_i, l_j \in G_k$, l_i and l_j must be compatible. This procedure is repeated until all $l_i \in \mathcal{R}(S)$ are processed. Once again, let us consider Fig. 7, the groups are $G_0 = \{\{p_1, \ldots, p_6\}, \{p_7, p_8, p_{10}, p_{11}, p_{12}\}\}$, $G_1 = \{p_4, p_6, p_8\}, G_2 = \{p_6, p_8, p_{10}\}, G_3 = \{p_5, \ldots, p_8\}, G_4 = \{p_3, p_4, p_6\}$, $G_5 = \{p_9, p_{10}, p_{11}\}$, $G_6 = \{p_8, p_9, p_{10}\}$, $G_7 = \{p_7, p_8, p_9\}$.

In the combination step, all $G_i \in \mathcal{G}$ are sorted in descending order according to $\left(\sum_{l_k \in G_i} \text{len}(l_k) \times \sum_{l_k \in G_i} \text{num}(l_k)\right) / \#(G_i)$, where $\#(G_i)$ is the number of sequences in G_i. They are processed in order, to extract disjoint and compatible groups, whose set is denoted by $\mathcal{L}$. Given a group $G_k \in \mathcal{G}$, if G_k is mergeable with some $G_i \in \mathcal{L}$ and compatible with all the other $G_i \in \mathcal{L}$, G_k is merged with all the mergeable groups. If G is not mergeable with any $G_i \in \mathcal{L}$ but compatible with all the $G_i \in \mathcal{L}$, G_k is inserted into $\mathcal{L}$. Otherwise G is discarded. This procedure is repeated until all $G_k \in \mathcal{G}$ are processed. Consider Fig. 7, $\mathcal{L} = \{\{p_1, \ldots, p_6\}, \{p_7, p_8, p_{10}, p_{11}, p_{12}\}\}$, and this is because the group G_0 is neither mergeable nor compatible with other groups. The aforementioned steps are documented in Algorithm 2 and the results are shown in Figs. 9 and 11b.

Algorithm 2: Stripe segmentation on a line

```
foreach sample line do
    //Grouping Step
    $\mathcal{G} \leftarrow \Phi$;
    sort $l_i \in \mathcal{R}(S)$ according to $len(l_i) \times num(l_i)$;
    foreach $l_i \in \mathcal{R}(S)$ do
        foreach $G_k \in \mathcal{G}$ do
            if $l_i$ is compatible with all $l_j \in G_k$ then
                $l_i$ is inserted into $G_k$;
        if $l_i$ has not been inserted into any $G_k \in \mathcal{G}$ then
            $l_i$ forms a new group $G$ and insert it into $\mathcal{G}$;
    //Combination Step
    $\mathcal{L} \leftarrow \Phi$;
    sort $G_i \in \mathcal{G}$ according to $\frac{\sum_{l_k \in G_i} len(l_k) \times \sum_{l_k \in G_i} num(l_k)}{\#(G_i)}$;
    foreach $G_i \in \mathcal{G}$ do
        if $G_i$ is mergeable with some $G_k \in \mathcal{L}$ and compatible with other $G_k \in \mathcal{L}$ then
            $G_i$ is merged with all mergeablemergeable $G_k \in \mathcal{L}$;
        else if $G_i$ is compatible with all $G_k \in \mathcal{L}$ then
            $G_i$ is inserted into $\mathcal{L}$;
```

4.2.3 Stripe segmentation on curved surface

After splitting on each sample line independently, some lines may not split appropriately (Fig. 11b). In this step, we combine the splitters on all the sample lines to vote for the most appropriate splitters to split the stripes eventually. We select a sample line (which is always the middle one) as the base line. Every splitter is traced along the stripe direction field to its corresponding position on the base line. Similar to the combination step on a single line, we also need to combine these groups to split the base line. However, this step is different in the following aspects. Since the positions of corresponding splitters from different sample lines are not strictly equal, we tolerate certain deviations between corresponding splitters while merging, and average their positions. After combination, we complete the gaps between the subsequences of $\mathcal{L}$ according to the splitters near the gaps. If it is irregular splitting, the gaps near the splitters with similar textures to the segmented stripes are filled. Finally, the groups except G_0 are discarded as non-repeated region (e.g., dashed regions in Fig. 29a). If it is regular splitting, we fill all the gaps according to the repetitive pattern of G_0. Finally, from these splitters, the complete stripes are traced along the stripe direction field to segment the entire regular and irregular surfaces as shown in Figs. 10 and 11c (the top façade is split separately), respectively.

To test the robustness of our segmentation algorithm, as shown in Fig. 12, we segment a façade of the model captured from two different views with about 3 million and 0.79 million pixels, respectively, and both examples result in correct segmentation with the same parameter setting.

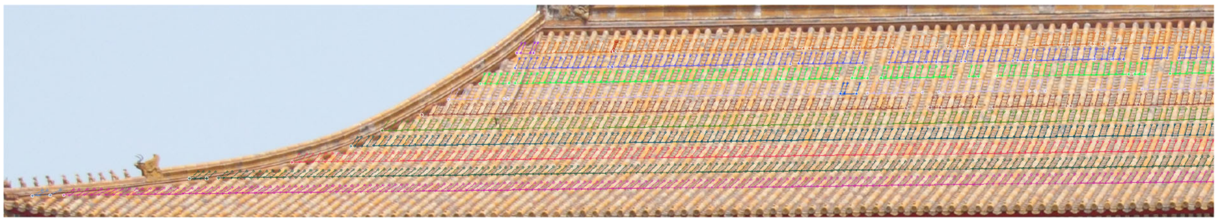


Fig. 9 Stripe segmentation on a line. Different groups of splitters are drawn in *different colors*

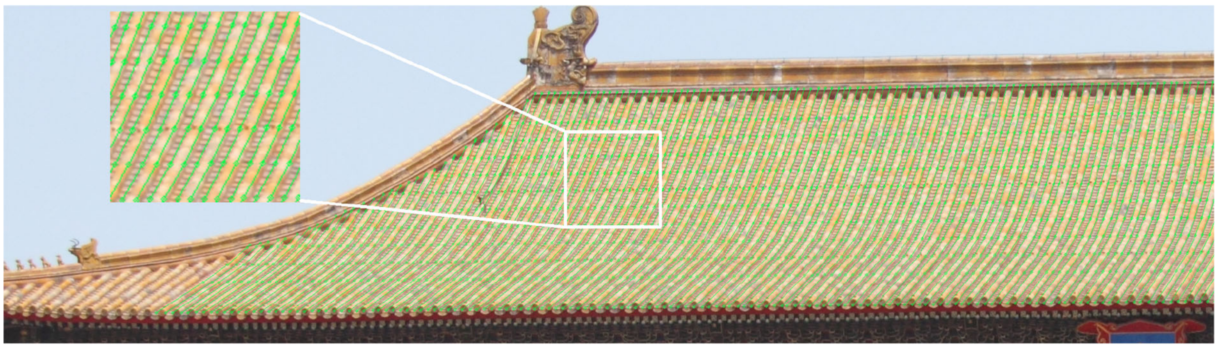


Fig. 10 Regular segmentation for roof tile. Each tile is regarded as a *curved stripe*

5 3D reconstruction

Geometry reconstruction from a single image is an ill-posed problem where additional clues are needed. Instead of using pyramid frustum or parallelepiped, making use of orthogonality between the stripe and the horizontal (vertical for modern buildings) direction, we calibrate the camera parameters initially. Later, we develop two 3D reconstruction methods for ancient Chinese architectures and curved modern façades, respectively.

Initial focal length recovery: We assume that the intrinsic parameters only have one degree of freedom (i.e., focal length). Since the parallel stripes are orthogonal to the horizontal (vertical) direction, using the inlier vanishing points, we refit the vanishing line of the parallel stripes in the form of $ax + v_y/v_x * a * y + 1 = 0$, where v_x and v_y are the x and y coordinates of the horizontal (vertical) direction vanishing points, respectively. Making use of the orthogonality, the camera focal length is $f = \sqrt{(v_x^2 + v_y^2)/(av_x + av_y^2/v_x)}$.

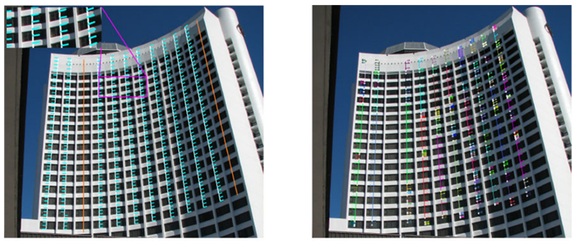
5.1 Ancient Chinese architecture reconstruction

Due to the symmetry of Chinese architectures, similar to Jiang [27], we flip the image horizontally to get a second virtual camera, which mimics the image captured at the symmetric position with respect to the symmetric plane of the architecture. Given the façade contour and horizontal vanishing point, we sweep horizontal lines to intersect the contour. Assuming a horizontal line intersects the contour at p_1 and p_2, we obtain two pairs of correspondences (p_1, p_2') and (p_2, p_1'), where p_1' and p_2' are the horizontally flipped p_1 and p_2 respectively. Then, based on these correspondences, we evaluate the projection matrices of the two cameras followed by bundle adjustment [37]. Finally, we get the refined projective matrices of the two images and the 3D contour of the building.

5.2 Curved modern building façade reconstruction

Different from Chinese architectures, the symmetric points on the contour of modern façade are co-planar, from which the fundamental matrix cannot be recovered. Thanks to the grid pattern detected in Sect. 4, we present a novel method to recover the 3D curved façade. First, the geometry is roughly recovered by developing rectangle patches successively. Second, the geometry and camera are refined in a non-linear least square procedure.

As shown in Fig. 11, the parallel curved stripes are piecewise linear and consisting of rectangular patches, so given the intrinsic parameter matrix K, we can rotate the camera to rectify the rectangular patch successively by a homography $H = KRK^{-1}$ [37], where R is the rotation matrix to make


(a) (b)

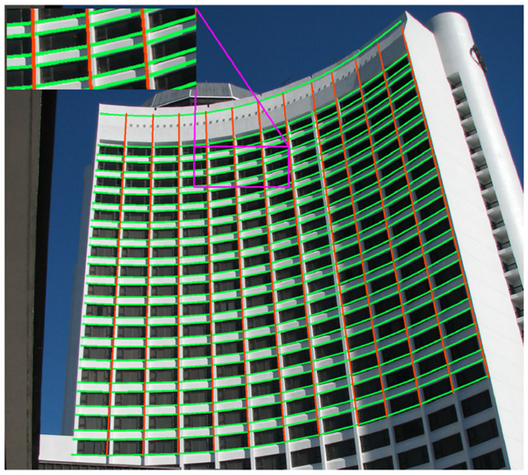

(c)

Fig. 11 **a** The floor direction field. **b** Segmentation on sample lines. **c** The irregular segmentation for hotel building façade into grids

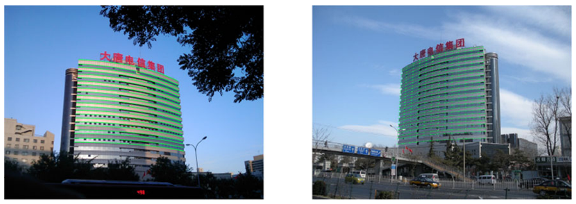

(a) (b)

Fig. 12 Façade segmentation of a model with two different views and different resolutions

H transform the two vanishing points of every rectangle to infinity. The angles are computed from the intrinsic matrix and the vanishing points. Thus, the initial 3D coordinates $\{V_{ij}\}_{i,j=0}^{m,n}$ of the corners (i.e., grid points) are derived, where i denotes the row (floor) index and j denotes the column index.

After the initialization, we compute the camera projection matrix $\mathcal{P}$ according to the 3D points $\{V_{ij}\}_{i,j=0}^{m,n}$ and their 2D projections $\{p_{ij}\}_{i,j=0}^{m,n}$. If the building is symmetric (it may be noted that whether the building is symmetric or not is specified by the user), we use principal component analysis to compute its symmetric plane P_{sym} and minimize Eq. (1) to optimize $\{V_{ij}\}_{i,j=0}^{m,n}$, $\mathcal{P}$ and P_{sym}. In order to enforce the

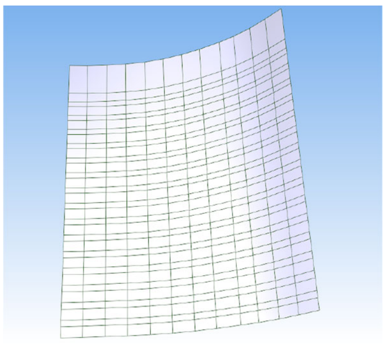


Fig. 13 The reconstructed hotel façade

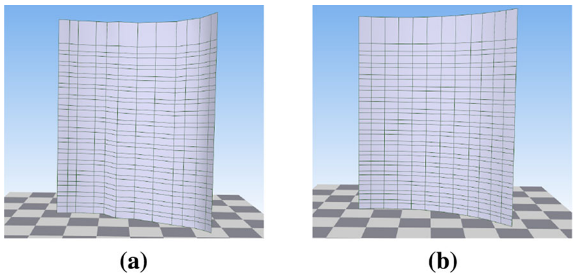


Fig. 14 **a** The initially recovered hotel façade, where we intentionally add more perturbations to investigate the convergence property of Eq. (1). **b** The optimized façade similar to the reconstructed façade in Fig. 13

symmetry, for every $V_{i_0 j_0} \in \{V_{ij}\}_{i,j=0}^{m,n}$, we generate a ghost point $V'_{i_0 j_0}$ to represent its symmetric point, which is initialized to the point on the 3D floor curve $\{V_{i_0 j}\}_{j=0}^{n}$ (i.e., the $V_{i_0 j_0}$'s row of the grid points) closest to the symmetric point of $V_{i_0 j_0}$.

$$d = \alpha \sum_{i,j} d_1^2 \left(\mathcal{P}V_{ij}, p_{ij}\right) + \beta \sum_{i,j} d_2^2 \left(\left(V_{ij} + V'_{ij}\right)/2\right) + \gamma \sum_{i,j} \theta^2 \left(V_{ij} - V'_{ij}\right) + \delta \sum_{i,j} d_3^2 \left(V'_{ij}\right), \quad (1)$$

where $d_1(\mathcal{P}V_{ij}, p_{ij})$ is the Euclidean distance from the projection of V_i to its corresponding image point p_{ij} (i.e., 3D-2D point correspondence), and $d_2((V_{ij} + V'_{ij})/2)$ is the distance from point $(V_{ij} + V'_{ij})/2$ to plane P_{sym} (i.e., the middle point of symmetric pairs should be on the symmetric plane). $\theta(V_{ij} - V'_{ij})$ is the sine of the angle between vector $V_{ij} - V'_{ij}$ and the normal of P_{sym} (i.e., the directional vector of symmetric pairs should be perpendicular to the symmetric plane). $d_3(V'_{ij})$ is the minimum distance from point V'_{ij} to the 3D floor curve $\{V_{ij}\}_{j=0}^{n}$ (i.e., the ghost symmetric points should be close to its located floor curve). The last three terms constrain the symmetry. In the case of non-symmetric building, only the first term is retained, which degenerates to ordinary bundle adjustment. We use Levenberg-Marquardt method to minimize Eq. (1) iteratively. In each iteration, the variables $\{V_{ij}, V'_{ij}\}_{i,j=0}^{m,n}$, P_{sym} and $\mathcal{P}$ are optimized independently in turn, while the others are fixed. In Eq. (1), the points on the same floor are enforced to lie on a common horizontal plane and the points vertically aligned are enforced to lie in a common vertical line. In our experiments, we set $\alpha = 4$, $\beta = 1$, $\gamma = 10$ and $\delta = 1$. Figure 13 shows the recovered 3D grid mesh of the building in Fig. 11c.

To investigate the convergence property of Eq. (1), we intentionally add more perturbations to the initial façade as shown in Fig. 14a. After optimization, the optimal façade recovered is shown in Fig. 14b similar to the reconstructed façade in Fig. 13.

6 Column segmentation

For Chinese architectures, the columns divide the room into bays (the areas between adjacent columns). We have to construct the bay, column and lintel components to meet the requirement of component generation and building variation. That is, we have to identify the columns for Chinese architectures and thus the bays and lintels are derived. For architectures with corridors, the inner bays are even occluded by outer columns, so we have to identify all the inner and outer columns. To ease the interactions, we present an interactive method to identify outer columns based on user-specified inner columns.

Initialization: The user first identifies the locations of the inner columns and draws the width of an arbitrary one. If the two side columns are not at the two ends, they are moved to the two ends for outer column segmentation. We sample a horizontal line at the middle height of the bays, to which the user-specified points are mapped along vertical direction and denoted by $\{c_0, \ldots, c_m\}$. Since the columns are bilaterally symmetric, the c_i and c_{m-i} are symmetric pairs and their 3D coordinates $\{C_0, \ldots, C_m\}$ are thus computed by triangulation [37]. Also, we fit the outer column plane P_{out} based on 3D points on the contour, and the inner column plane P_{in} is fitted on $\{C_0, \ldots, C_m\}$ constrained by the normal of P_{out}. For buildings without corridors, we draw the outer columns and the system determines whether the corridor exists automatically based on the ratio of the distance between the two fitted planes and the room height. Finally, $\{C_0, \ldots, C_m\}$ are projected onto P_{out} to get $\{C'_0, \ldots, C'_m\}$ as the initial positions of outer columns (please refer to yellow points in Fig. 15), which are further refined in the next step.

Iterative refinement: The bays are always evenly distributed except for the center and/or two side ones. We first refine $\{C'_0, \ldots, C'_m\}$ based on the symmetry and uniformity. $\{C'_0, \ldots, C'_m\}$ are first parameterized by a scalar on their line, but we still use these symbols and minimize,

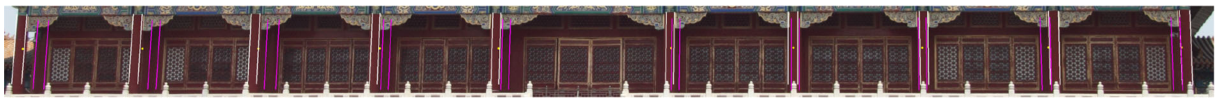


Fig. 15 Column segmentation. The *yellow points* are the projected points from user-specified inner column points to outer columns. After refinement, the final inner columns and outer columns are marked by *magenta lines* and *white line* respectively

$$\{C'_i\}_{i=1}^m = \underset{\{C'_i\},1\le i\le m}{\arg\min}\ \alpha \sum_{\text{uniform bays}} \|C'_i + C'_{i+2} - 2C'_{i+1}\|^2 + \beta \sum_{i<m/2} \|(C'_i + C'_{m-i}) - (C'_{i+1} + C'_{m-(i+1)})\|^2 + \sum_{1\le i\le m} \|C'_i - C'^0_i\|^2, \quad (2)$$

where the first term constrains the uniformities, and the second term constrains their symmetry. C'^0_i is the column location before refinement and the third term constrains the refined locations close to their source locations. We repeatedly refine $\{C'_i\}_{i=1}^m$ by above equation until the variations of C'_i are small enough or exceed maximum number of iterations. After refinement, $\{C'_0, \ldots, C'_m\}$ are projected onto the image to get $\{c'_0, \ldots, c'_m\}$. Since the inner and outer columns are aligned, the mapping between $\{c_0, \ldots, c_m\}$ and $\{c'_0, \ldots, c'_m\}$ is subject to a 1D homography. The columns are much smoother than the bays since the windows or doors are always decorated, meaning that the gradients on columns are smaller than that of bays. We iteratively refine the outer column positions based on the homography and image gradient alternately until the position variations are small enough or exceed the maximum number of iterations. In the gradient-based procedure, for every c'_i, given its column width w_i, we iteratively refine it to find the smallest gradient location. In each iteration, the search window is $s = w_i \exp(-2|c'_i - c_{\text{pre}}|/w_i)$, where c_{pre} is the column location after last iteration, which is initialized to c'_i. We find the new column center c_{new} by minimizing, $c_{\text{new}} = \arg\min_c\ g(c)$, $c_{\text{pre}} - s/2 \le c \le c_{\text{pre}} + s/2$, along with imposing c_{new} not to pass c_i, and $g(c)$ is the average image gradient in the interval $[c - w_i/2, c + w_i/2]$. The iteration stops if $|c_{\text{new}} - c_{\text{pre}}| \le 1$ or exceeding the maximum number of iterations. Finally, we compute the gradient on each vertical line of the room and their suppression radii in which their gradient is the largest. The lines with the top 5 % suppression radii are reserved as the candidate splitting lines. At last, the iteratively refined columns (just obtained in the prior step) and the intervals divided by the candidate splitting lines are combined to segment the columns (if a interval is covered more than 50 % by the refined columns, then mark it as column). The identified columns are shown in Fig. 15. The lintels are aligned with the columns in 3D, so they are segmented passingly by projecting the columns onto their planes.

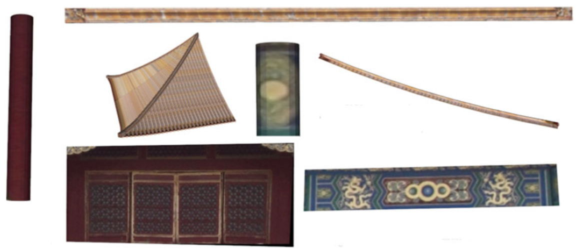


Fig. 16 A subset of the 3D components of the Meridian Gate

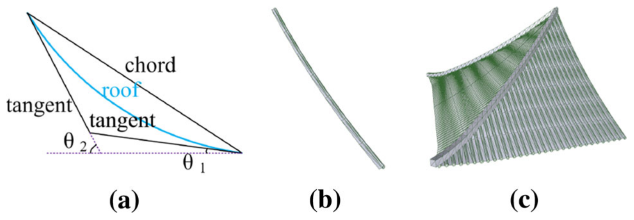


Fig. 17 Roof tile component. **a** Sketch drawing of the cross-section of tile. **b** Tile component. **c** Side roof component

7 Component, rule, and model generation

After image segmentation and 3D reconstruction, it now sets a stage to construct 3D architectural components, which are the basic elements to compose the final 3D models reusable for various model generation.

Chinese architecture components: Some of the components of the Meridian Gate are shown in Fig. 16. The inner columns divide the building into bays, which may be occluded by outer columns. To complete the occluded textures, making use of the symmetry of the bay, we flip the bay horizontally and register them followed by the use of graph cut texture synthesis [38] to stitch the two images. Finally, the repaired image is textured on a planar surface to form the bay component (see Bay in Fig. 16). The column and lintel column are simply formulated as a cylinder textured by their images, and the lintels are formulated as planar surfaces.

The curved roof is a bit complicated. We regard its formation as an arc sweeping horizontally. As shown in Fig. 17a, we compute the roof top ridge and bottom eave slop angle θ_1, θ_2, and the length of the chord. The roof depth equals to two times of the chord depth. For the Meridian Gate, $\theta_1 = 23.5°$ and $\theta_2 = 43.7°$. The tile curve is thus generated as shown in Fig. 17b. The side roof is constructed as a component shown in Fig. 17c, whose corner eave is tilted by a quadratic Bézier curve.

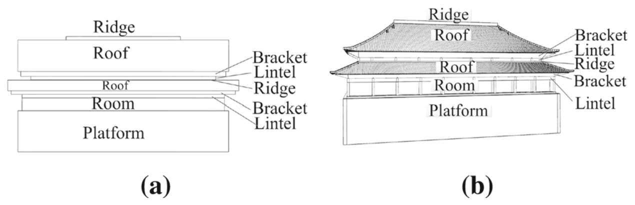


Fig. 18 **a** The mass models of the Meridian Gate model. **b** The mass models are split into components horizontally

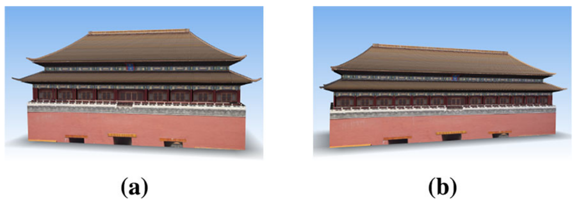


Fig. 19 **a** The Meridian Gate model. **b** The stretched Meridian Gate model

Modern architecture components: After segmenting the curved façade into grid pattern as shown in Fig. 11c, every rectangular patch forms a rectangle component textured by the image.

Rules derivation: We integrate the image-based modeling method with the procedural modeling method. Finally, we derive GGA [9] shape grammar rules automatically to compose these 3D components to construct architectures and their variants. The rules construct mass (coarse) models at first, which forms the overall structure of the building followed by the top-down splitting to generate component details.

For Chinese architectures, first, as shown in Fig. 18a, the mass models are first subdivided vertically into a collection of one or more city wall, room, lintel, bracket set, roof, ridge, etc. Since the width differences of different partitions (e.g., roof and room) equal to their depth differences, the depths of other partitions are evaluated from their width differences to the roof. Second, as shown in Fig. 18b, for each mass model, we split them horizontally into specific components, such as columns, bays, lintels, etc. The side and rear faces are deployed as simple wall similar to most Chinese architectures. One of the advantages of procedural modeling is that, it can generate a suite of various models. The components (e.g., columns, roof tiles) are split by repeat split rules to fill as many elements as the space could afford. Finally, the model is generated from the rules and components. The generated Meridian Gate and one of its variants are shown in Fig. 19a, b respectively.

For modern buildings, first, as shown in Fig. 20a, the mass model is constructed as a right prism whose base face is derived from the recovered façade and depth is specified by the user. Second, as shown in Fig. 20b, every face of the prism is split vertically into floors to generate windows and walls, where the most repeated group $G_0 \in \mathcal{L}$ is split by repeat

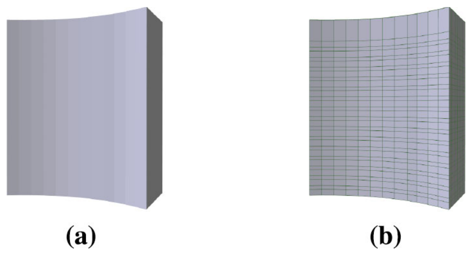


Fig. 20 **a** The mass models of the hotel model. **b** The mass models are split into floors and windows and walls

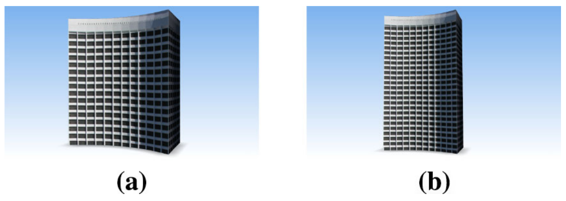


Fig. 21 **a** The hotel model. **b** The stretched hotel model

split rules to adaptively fill the spaces while the building height varies. The final hotel model is shown in Fig. 21a and a stretched building is shown in Fig. 21b, where the windows and walls grow adaptively.

8 Experimental results and evaluations

The images tested in this paper are either captured using our digital camera, or downloaded from the Internet. Their resolutions vary from about 2 million to 15 million pixels. Please refer to the additional material for more results.

We have compared the Meridian Gate model with the ground truth whose width (the room partition) is 60.05 m and depth is 25.00 m. The depth error of the result, whose width is 1.733 and depth is 0.754, is about 4.5 %.

The Imperial Ancestral Temple is one of the widest buildings with 11 bays. As shown in Fig. 1, its camera focal length is 3716.72. The slope angle of the top roof eave is 22.6° and ridge is 45°. The generated model width is 1.409 and depth is 0.614. Compared with the ground truth whose width is 66.79 m and depth is 29.09 m, the depth error is less than 0.1 %. Finally, the building and the deformed model are reconstructed, where the tiles and bays are generated adaptively.

Figure 22a shows a residential building with gable roof. We only draw its central façade due to occlusion, so only the central part is built as shown in Fig. 22b and the stretched model is shown in Fig. 22c. The roof is split into individual tiles as shown in Fig. 22d.

Figure 23a shows a building with gable and hip roof. The reconstructed model is shown in Fig. 23b. Different from the Meridian Gate image shot on a sunny day, this image is shot

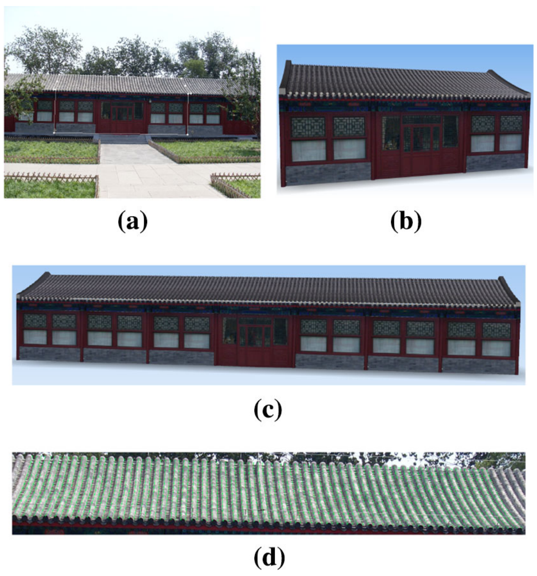


Fig. 22 **a** A residential building image with user-specified contour. **b** The reconstructed building. **c** The stretched building. **d** The roof is split into tiles

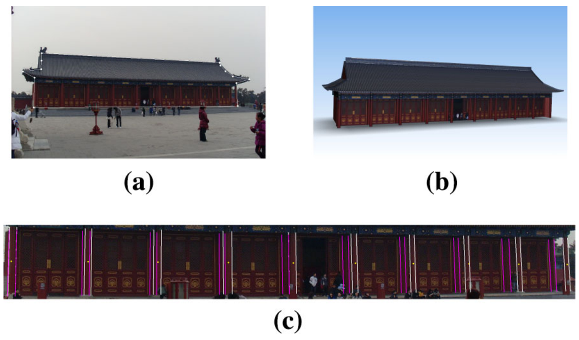


Fig. 23 **a** A gable and hip roof building. **b** The generated 3D building. **c** The identified columns

on a cloudy day resulting in more ambiguous color between the outer columns and inner bays. The identified columns are shown in Fig. 23c, which also facilitate the identification of lintels. As shown in Fig. 24, we also test two views of the same roof for segmentation, and both cases have resulted in correct results, demonstrating that our method is not affected by different view directions.

Figure 25a, c shows two pavilions. Different from previous model, they are rotationally symmetric. Instead of the cuboid mass model, their mass models are initialized as regular right prisms, which can be split into façades and columns. The generated models are shown in Fig. 25b, d.

As shown in Fig. 26a, the building with vague floor boundary is segmented adaptively, where the top and bottom of the façade is appropriately separated. The reconstructed building is shown in Fig. 26b and two deformed buildings are shown in Fig. 26c, d.

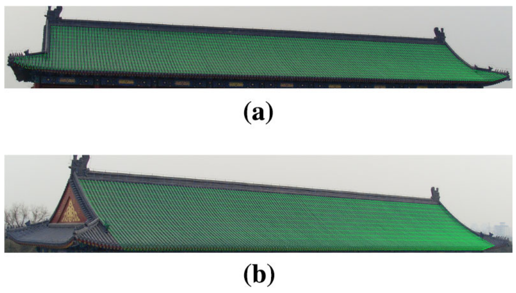


Fig. 24 Segmentation of the gable and hip roof from two different views

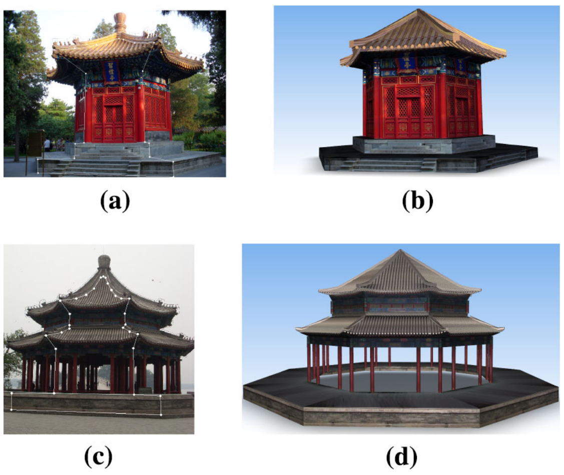


Fig. 25 **a** A pavilion with 6 faces. **b** The generated pavilion. **c** A two-story pavilion with 8 faces. **d** The generated pavilion

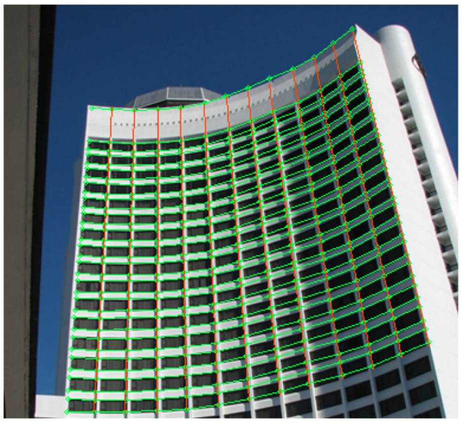


Fig. 26 **a** The building is split into floors, where the top and bottom of building is split adaptively. **b** The generated building. **c** The façade is flattened. **d** The façade is warped

The symmetry is not necessary. For a non-symmetric curved façade, which is also a generalized cylinder, as shown in Fig. 27, we can also segment and reconstruct the façade appropriately.

To test the robustness of our algorithm, as shown in Fig. 28, we compress Fig. 11c to 1/4 of the original resolution and

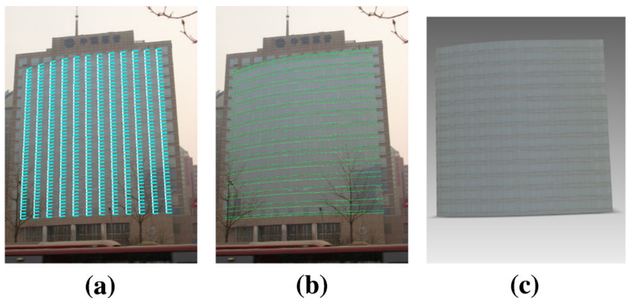


Fig. 27 A non-symmetric model. **a** The floor direction field. **b** The split floors. **c** The reconstructed model

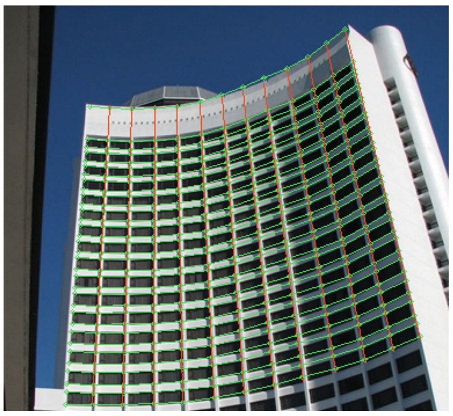


Fig. 28 Segmentation on an image with 1/4 resolution of Fig. 11c

segment the façade with the same parameter setting. Both have resulted in correct results.

Except for curved façades, the planar façade building UOB Plaza is shown in Fig. 29a, which is irregularly split. The window floors are classified into a repeated group *G* consisting of 4 continuous sequences of splitters while the dashed areas are split separately. The reconstructed building is shown in Fig. 29b and a shrunk building is shown in Fig. 29c, where the floors are generated adaptively. Figure 29d shows a stretched building where the floors are adjusted adaptively.

Except for some user-specified parameters, such as building styles (ancient or modern), number of building sides, etc, all the user interactions are simple and natural, and they are illustrated in the affiliated video. For ancient Chinese architectures, the user needs to draw building contour, identify building regions and their types, and identify inner columns. The interactions need about 2–3 min. For modern architectures, the user just needs to draw the contour, which is always within 1 min.

Comparison: Figure 30b shows our reconstructed Pavilion of Manifest Benevolence (also known as TiRenGe) from the input image (Fig. 30a). Jiang et al. [27] also presented an interactive method and have constructed the same model. Their method, however, needs to draw a frustum, roof curves, roof tiles, walls, columns, and others, for calibration and

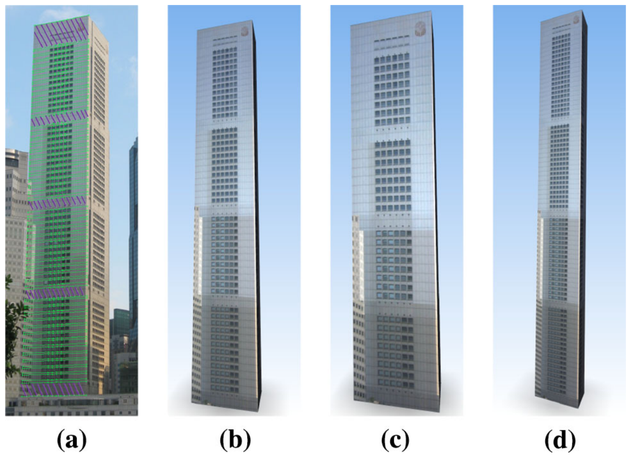


Fig. 29 The UOB Plaza. **a** Though the *bottom left part* is occluded, the building is adaptively split into repeated floors, which is irregular due to the *purple dashed regions*. **b** The reconstructed UOB building. **c** The shrunk building. **d** The stretched building

building generation. Nonetheless, they did not explicitly construct architectural components, so their building is only a static model. In contrast, our building can be extended adaptively (Fig. 30c) and its components can be substituted because of our component-aware modeling capability. As shown in Fig. 30d, the bays are changed to the ones of the Meridian Gate and the gable and hip roof building, respectively. As shown in Fig. 30e, the lintels of Jiang et al.'s method have exhibited mis-alignment (highlighted inside circles) because they are never segmented into individual components. Furthermore, the inner columns are not reconstructed either. In contrast, we construct the appropriate columns and lintels as shown in Fig. 30f. Our automatic tile segmentation algorithm produces more accurate tile density compared with Jiang's manual segmentation.

Zhang et al. [31] presented a low-rank based algorithm to unwrap generalized cylindrical surface. However, when the region is large beyond polynomial model or violation of the low-rank assumption, their method may fail to unwrap the surface. In contrast, our method is not restricted by the polynomial model or the low-rank assumption. As shown in Fig. 31, we rectify the generalized cylindrical surface successfully, which fails by Zhang's algorithm though using different images.

Limitations: Currently, our system may have several limitations that call for improvement. For Chinese architectures, since we rely on the tile directions for calibration, the roof should be captured completely with enough resolution. The camera should not be positioned near the symmetry plane of a building, because we would need a wide baseline between the source and the mirrored image. Meanwhile, the image plane should not be parallel to the façade to avoid the degeneracy of vanishing point at infinity. Moreover, our method is

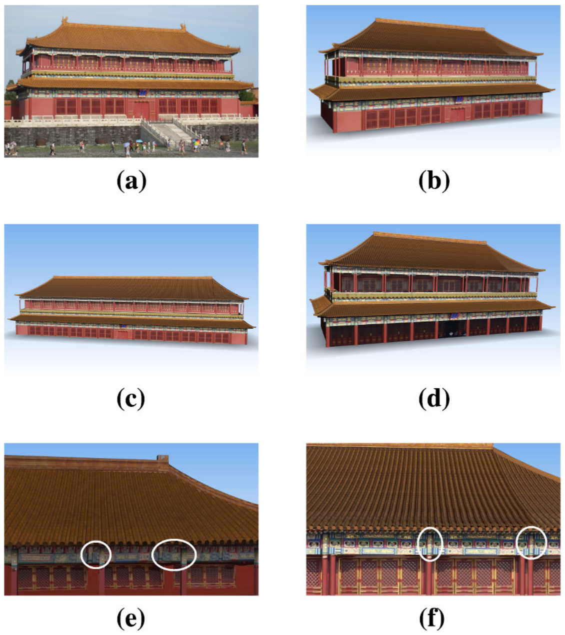


Fig. 30 The TiRenGe model. **a** The input image. **b** Our reconstructed model. **c** Our extended model. **d** The deformed model with different bays. **e** A local close-up view of Jiang's model [27], the mis-alignment is highlighted (Thanks to Ping Tan for providing their final model). **f** The same localized view of our model for the comparison purpose

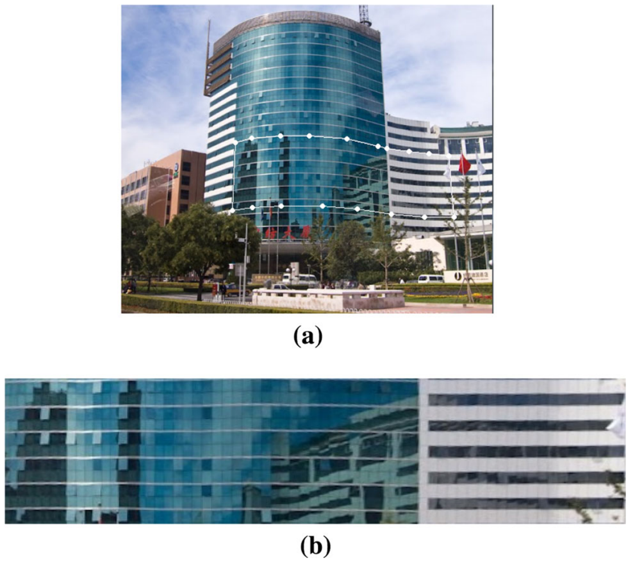


Fig. 31 Generalized cylindrical surface rectification. **a** The source image along with user drawn contour. **b** The rectified façade. While Zhang's algorithm [31] fails to rectify the same façade though different images

inadequate for perfectly round buildings, such as the Temple of Heaven (in Beijing). It is also hard to handle complicated foundations (e.g., the foundation of the Imperial Ancestral Temple) and bumpy façades (e.g., balcony) of modern buildings from a single image.

9 Conclusion and future work

We have detailed a component-aware modeling framework to reconstruct architectures from a single image. The core of the system is an adaptive curved stripe segmentation algorithm to segment building tiles and floors, based on which the 3D dimensions are recovered from a single image. Especially for Chinese architectures, the outer columns are identified based on user-specified inner columns. Finally, the 3D components as well as shape rules are generated, from which the 3D models and their variants are constructed. Our method not only generates varying 3D models, but also extracts reusable 3D architectural components towards new building construction of various styles and versatile and massive urban production at city-scale.

Our system could be further improved along several directions in the near future. To reduce users' burden for costly and extensive interaction, the building contour could be segmented by the image matting approach and the building partitions could be recognized by computer vision techniques. The two alternately stripe segmentation algorithm may be further generalized to adequately handle n-alternately repeated pattern segmentation problems.

Acknowledgments The authors would like to thank the anonymous reviewers for their constructive comments that help improve this paper. This work is supported in part by National Natural Science Foundation of China (Grant No. 61300068, 61190120, 61190121, 61190125, 61300067, 61272348, 61202235, 61073078), National Science Foundation of USA (Grant No. IIS-0949467, IIS-1047715, and IIS-1049448), the National High-tech R&D Program of China (Grant No. 012AA011503), Postdoctoral Science Foundation of China (Grant No. 2013M530512), Ph.D. Program Foundation of Ministry of Education of China (Grant No. 20111102110018), National Key Technology R&D Program of China (Grant No. 2014BAK18B01).

References

1. Zheng, Q., Sharf, A., Wan, G., Li, Y., Mitra, N.J., Cohen-Or, D., Chen, B.: Non-local scan consolidation for 3d urban scenes. ACM Trans. Graph. 29, 94:1–94:9 (2010)
2. Poullis, C., You, S.: Photorealistic large-scale urban city model reconstruction. IEEE Trans. Vis. Comput. Graph. **15**, 654–669 (2009)
3. Xiao, J., Fang, T., Zhao, P., Lhuillier, M., Quan, L.: Image-based street-side city modeling. ACM Trans. Graph. **28**(5), 1–12 (2009)
4. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Towards internet-scale multi-view stereo. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1434–1441 (2010)
5. Musialski, P., Wimmer, M., Wonka, P.: Interactive coherence-based façade modeling, Computer Graphics Forum (Proceedings of EUROGRAPHICS 2012) 31(2), 661–670 (2012)
6. AlHalawani, S., Yang, Y.-L., Liu, H., Mitra, N.J.: Interactive facades analysis and synthesis of semi-regular facades. Comput. Graph. Forum **32**(2), 215–224 (2013)
7. Musialski, P., Wonka, P., Aliaga, D.G., Wimmer, M., van Gool, L., Purgathofer, W.: A survey of urban reconstruction. In: EURO-

GRAPHICS 2012 State of the Art Reports, Eurographics Association, pp. 146–177 (2012)
8. Wonka, P., Wimmer, M., Sillion, F., Ribarsky, W.: Instant architecture. ACM Trans. Graph. **22**(3), 669–677 (2003)
9. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. ACM Trans. Graph. **25**(3), 614–623 (2006)
10. Lipp, M., Wonka, P., Wimmer, M.: Interactive visual editing of grammars for procedural architecture. ACM Trans. Graph. **27**(3), 1–10 (2008)
11. Talton, J.O., Lou, Y., Lesser, S., Duke, J., Měch, R., Koltun, V.: Metropolis procedural modeling. ACM Trans. Graph. 30(2), 11:1–11:14 (2011)
12. Hou, F., Qi, Y., Qin, H.: Drawing-based procedural modeling of chinese architectures. IEEE Trans. Vis. Comput. Graph. **18**(1), 30–42 (2012)
13. Kelly, T., Wonka, P.: Interactive architectural modeling with procedural extrusions. ACM Trans. Graph. 30, 14:1–14:15 (2011)
14. Merrell, P., Manocha, D.: Model synthesis: a general procedural modeling algorithm. IEEE Trans. Vis. Comput. Graph. **17**(6), 715–728 (2011)
15. Bokeloh, M., Wand, M., Seidel, H.-P.: A connection between partial symmetry and inverse procedural modeling. ACM Trans. Graph. 29, 104:1–104:10 (2010)
16. Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S.M.: Multi-view stereo for community photo collections. In: ICCV, pp. 1–8 (2007)
17. Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building rome in a day. In: IEEE International Conference on Computer Vision, pp. 72–79 (2009)
18. Pollefeys, M., Nistér, D., Frahm, J.M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewénius, H., Yang, R., Welch, G., Towles, H.: Detailed real-time urban 3d reconstruction from video. Int. J. Comput. Vis. **78**, 143–167 (2008)
19. Sinha, S.N., Steedly, D., Szeliski, R., Agrawala, M., Pollefeys, M.: Interactive 3d architectural modeling from unordered photo collections. ACM Trans. Graph. 27(5), 159:1–159:10 (2008)
20. Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., Quan, L.: Image-based façade modeling. ACM Trans. Graph. 27(5), 161:1–161:10 (2008)
21. Zhao, P., Fang, T., Xiao, J., Zhang, H., Zhao, Q., Quan, L.: Rectilinear parsing of architecture in urban environment. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 342–349 (2010)
22. Vanegas, C., Aliaga, D., Beneš, B.: Building reconstruction using manhattan-world grammars. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 358–365 (2010)
23. Ceylan, D., Mitra, N.J., Zheng, Y., Pauly, M.: Coupled structure-from-motion and 3d symmetry detection for urban facades. ACM Trans. Graph. 33(1), 2:1–2:15 (2014)
24. Hong, W., Yang, A.Y., Huang, K., Ma, Y.: On symmetry and multiple-view geometry: structure, pose, and calibration from a single image. Int. J. Comput. Vis. **60**, 241–265 (2004)
25. Liebowitz, D., Criminisi, A., Zisserman, A.: Creating architectural models from images. In: Annual Conference of the European Association for Computer Graphics (Eurographics), vol. 18, pp. 39–50 (1999)
26. Wilczkowiak, M., Sturm, P., Boyer, E.: Using geometric constraints through parallelepipeds for calibration and 3d modeling. IEEE Trans. Pattern Anal. Mach. Intell. **27**(2), 194–207 (2005)
27. Jiang, N., Tan, P., Cheong, L.-F.: Symmetric architecture modeling with a single image. ACM Trans. Graph. **28**(5), 1–8 (2009)
28. Wu, C., Frahm, J., Pollefeys, M.: Repetition-based dense single-view reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3113–3120 (2011)
29. Chen, X., Kang, S.B., Xu, Y.-Q., Dorsey, J., Shum, H.-Y.: Sketching reality: realistic interpretation of architectural designs. ACM Trans. Graph. **27**(2), 1–15 (2008)
30. Prasad, M., Fitzgibbon, A.: Single view reconstruction of curved surfaces. In: IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1345–1354 (2006)
31. Zhang, Z., Liang, X., Ma, Y.: Unwrapping low-rank textures on generalized cylindrical surfaces. In: Proceedings of the 2011 International Conference on Computer Vision, IEEE Computer Society, Washington, DC, USA, pp. 1347–1354 (2011)
32. Liu, Y., Hel-Or, H., Kaplan, C.S., Gool, L.J.V.: Computational symmetry in computer vision and computer graphics. Found. Trends Comput. Graph. Vis. **5**(1–2), 1–195 (2010)
33. Mitra, N.J., Pauly, M., Wand, M., Ceylan, D.: Symmetry in 3d geometry: extraction and applications. Comput. Graph. Forum **32**(6), 1–23 (2013)
34. Müller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. ACM Trans. Graph. **26**(3), 85 (2007)
35. Wu, C., Frahm, J.-M., Pollefeys, M.: Detecting large repetitive structures with salient boundaries. In: Proceedings of the 11th European conference on Computer vision: Part II, ECCV'10, Springer, Berlin, Heidelberg, pp. 142–155 (2010)
36. C.-H. Shen, S.-S. Huang, H. Fu, S.-M. Hu, Adaptive partitioning of urban facades, ACM Trans. Graph. 30 (6) (2011) 184:1–184:10
37. Hartley, R.I., Zisserman, A.: Multiple view geometry in computer vision, 2nd Edn. Cambridge University Press. ISBN: 0521540518 (2004)
38. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. ACM Trans. Graph. **22**, 277–286 (2003)

Fei Hou received his Ph.D. degree of computer science from Beihang University in 2012. He was a post doctoral researcher at Beihang University from 2012 to 2014 and currently he is a post doctoral researcher at Nanyang Technological University. His research interests are medical image processing, image-based modeling and data vectorization etc.

Hong Qin received the BS and MS degrees in computer science from Peking University, China. He received the PhD degree in computer science from the University of Toronto (UofT) in 1995. He is a full professor of computer science in the Department of Computer Science at State University of New York at Stony Brook (Stony Brook University). During his years at the University of Toronto, he received UofT Open Doctoral Fellowship. He was also a recipient of NSF CAREER Award from the US National Science Foundation (NSF), Honda Initiation Award, and Alfred P. Sloan Research Fellow by the Sloan Foundation. Currently, he serves as an associate editor for The Visual Computer, Graphical Models, and Journal of Computer Science and Technology. His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer-aided geometric design, humancomputer interaction, visualization, and scientific computing. Detailed information about him can be found from his website: http://www.cs.sunysb.edu/~qin. He is a senior member of the IEEE and the IEEE Computer Society.

Yue Qi is a professor in the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University, China. He received the BS degree and PhD degree in system engineering from National University of Defense Technology in 1991 and 2001 respectively, the MS degree in computer science from University of Science and Technology of China in 1995. His research interests include virtual reality, augmented reality and computer graphics, with an emphasis on geometry and appearance modeling. He is a member of the IEEE and the IEEE Computer Society.