EUROGRAPHICS 2012 / P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (*2012*), *Number 2*

Interactive Coherence-Based Façade Modeling

Przemyslaw Musialski [1,2,4] Michael Wimmer [1] Peter Wonka [3,4]

[1]TU Vienna [2]VRVis [3]KAUST [4]ASU

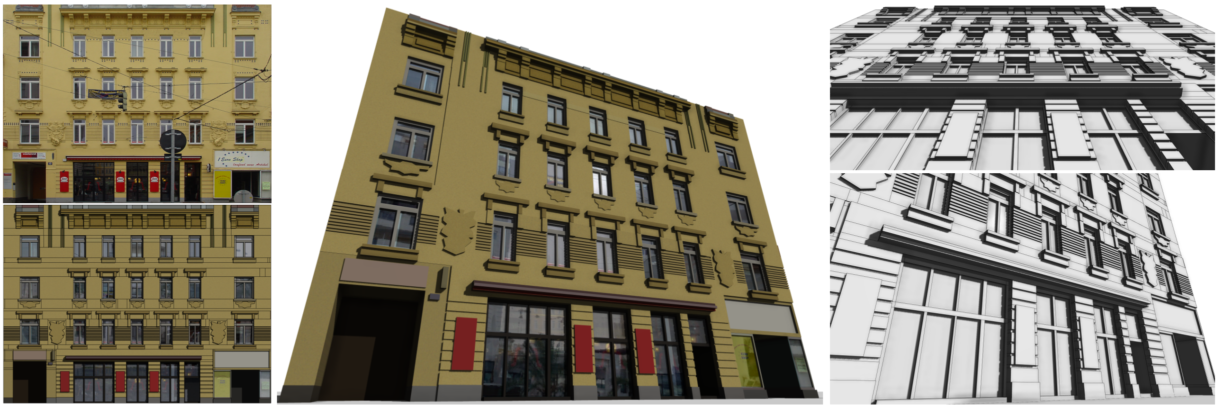


Figure 1: *Starting from an orthogonal input image (left top), the user interactively segments the façade into shapes (bottom left). Most split lines and symmetries are found automatically by the system, while the global façade structure is determined by the user. The input resolution is 1024 × 756, the number of visible shapes is 1346, and the modeling time is about 8 minutes (with material and depth assignments about 15 minutes).*

Abstract

We propose a novel interactive framework for modeling building façades from images. Our method is based on the notion of coherence-based editing which allows exploiting partial symmetries across the façade at any level of detail. The proposed workflow mixes manual interaction with automatic splitting and grouping operations based on unsupervised cluster analysis. In contrast to previous work, our approach leads to detailed 3d geometric models with up to several thousand regions per façade. We compare our modeling scheme to others and evaluate our approach in a user study with an experienced user and several novice users.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages

1. Introduction

The reconstruction of geometric models of building façades from photographs is an important tool in urban modeling scenarios. The main step in this process, segmenting a façade into its structural elements (henceforth called *shapes*), has been tackled by several automatic methods. All these methods share the property that they create models of low or intermediate level of detail and complexity.

However, many applications require models of high accuracy and geometric detail. For example, Figure 1 shows a façade which is not only structured into floors, columns, and windows, but it also contains detail for window elements, structure for different ornamentations, etc. Therefore, the research question we tackle in this paper is how to model a highly detailed façade from a photograph?

One approach would be to start with an automatic method like [XFZ*09, WFP10, SHFH11] and refine the resulting model to obtain the desired detail. However, it turns out that automatic façade subdivisions are not amenable to manual refinement: first, they contain numerous inaccuracies, which are hard and time-consuming to fix manually, and second, they often do not respect the major subdivision that would be done by a human designer, and on which such a designer would base further modeling actions.

In this paper, we follow a different approach: instead of starting with an initial automatic model, we propose a modeling paradigm where the human designer is in control of the modeling workflow, but is supported by automatic modeling tools. In particular, we have identified two types of operations where it is crucial to support the designer:

First, façade splitting in horizontal or vertical directions. For this operation, we contribute a robust automatic operator, which also provides a grouping of elements into similar types (e.g., wall, window, etc.). However, the most important aspect about this operator is that it is under the control of the user, so even if one automatic split operation produces an error, this is immediately recognized and can easily be corrected.

Second, interactive operations on groups of elements of a façade. Therefore, our contribution is that we propose a synchronized editing tool, which allows the designer to simultaneously work on a set of grouped façade elements, carrying out operations like inserting or moving a split, or assigning an attribute value.

The third contributions of this paper is a semi-automatic façade modeling workflow based on the concept of *coherence-based modeling* among particular façade elements. We have implemented a system that utilizes the proposed concept and the two mentioned tools that we call *automatic façade split* and *syncronized modeling*. Furthermore, we have carried out a large experiment where we compare our system against completely manual modeling, manual coherence-based modeling, as well as other façade modeling schemes, like the CGA-scheme and the interactive method proposed by Xiao et al. [XFT*08]. This experiment confirms our hypothesis that assisted manual modeling is preferable when high-quality models are required.

2. Related Work

Interactive Modeling. In their seminal work, Debevec et al. [DTM96] introduced a multi-view interactive method that combines *geometry-based modeling* with *image-based modeling* into one pipeline. More recently, Sinha et al. [SSS*08] presented an interactive system for generating textured 3d models of architecture from unordered sets of photographs which relies on camera parameters and point-clouds generated by a structure-from-motion process [SSS07] as a starting point. A further interactive image-based approach to façade modeling has been proposed by Xiao et al. [XFT*08], where façades are subdivided in a top-down manner and structured as a graph, and then followed by a bottom-up merging with the detection of reflectional symmetry. Furthermore, Jiang et al. [JTC09] extract 3d building-structure from a single image by utilizing the symmetry of the objects. Recently, more methods for interactive reconstruction from laser scan have been proposed [NSZ*10, ZSW*10], as well as a semi-automatic method which fuses laser scan data with photographs [LQS*11].

Another approach introduces *interactive inverse procedural modeling* to infer procedural grammars from photos of buildings [ARB07]. All these mentioned methods incorporate user interaction, and in general, provide lower level of detail as the method proposed in this paper.

Automatic Modeling. Also fully automatic reconstruction has been approached with the concept of *inverse procedural modeling*. Most methods are based on the Markov-Chain-Monte-Carlo (MCMC) optimization scheme: Alegre and Dellaert [AD04] proposed a specific set of grammar rules and an MCMC approach to optimize the parameters in order to fit a hierarchical model against the façade image. Dick et al. [DTC04] presented also a probabilistic MCMC approach which fits a predefined style grammar for certain architecture types. Korah and Rasmussen [KR07] introduced a method for automatic detection of window grids in ortho-rectified façade images based on MCMC optimization.

A further recent approach [KST*09] examines a rectified façade image in order to fit a hierarchical tree grammar. Teboul et al. [TSKP10] extend this work by combining a bottom-up segmentation through superpixels with top-down consistency checks coming from style rules and reinforcement learning [TKS*11].

Also more traditional image processing methods have been explored in order to detect the structure of façades: Lee and Nevatia [LN04] propose an automatic window detection method that uses edge-information. Müller et al. [MZWvG07] use also edges and mutual information to cut a single façade image into floors and tiles, and extract procedural rules. Xiao et al. [XFZ*09] provide an automatic multi-view image-based approach to generate coarse street-side 3d models. Musialski et al. [MRM*10] proposed a method to detect repetitive tiles in ortho-images and Wu et al. [WFP10] proposed a workflow that detects symmetries and boundaries of salient objects in general façade images. Recently Shen et al. [SHFH11] presented a method to detect concatenated grids in images or LiDAR scans of façades.

All these methods are generally limited to the detection of windows or grids of elements, thus their level of detail is far below the one presented here.

Procedural Modeling. Procedural architectural modeling is an approach to model urban environments by the means of rules defined by production systems [WWSR03, MWH*06, LWW08]. A recent state-of-the-art report [VAW*10] provides a comprehensive review of this topic.

Our approach is related to procedural modeling in that we also create a hierarchical representation of a scene, but we do not create more general rules.

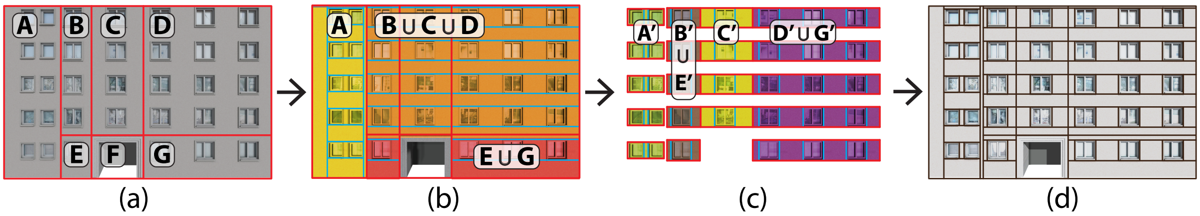


Figure 2: *Coherence-based modeling (Section 3). The input is split into 7 shapes (a). Then, mutually coherent shapes are grouped and split coherently: first in horizontal (b), then in vertical direction (c). In the final result, windows are separated (d).*

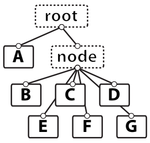
3. Coherence-Based Modeling

The input to our modeling system is a rectified image of a façade, which can be created from one or more images as described for instance in [MLS*10]. Note that we do not need to remove small occluders (like street lamps or signs in Fig. 1), as coherent editing will usually take care of this. The output of a modeling session is a segmentation of a façade into shapes (rectangles), where each shape can belong to one or more *groups*.

Our modeling scheme is based on the so-called *guillotine cutting*, which can be generated by applying simple axis aligned edge-to-edge cuts of rectangular regions (cf. Figure 2). This kind of splitting has a number of advantages: (1) it is quite intuitive and easy to generate with little interactive effort, (2) it ensures that the underlying rectangle is cut into strictly non-overlapping elements, and finally (3) it is easy to store and to maintain in a hierarchical tree data structure.

Many façade layouts can be fitted into such a scheme, which is also a popular concept in procedural modeling, but the resulting trees rarely capture the semantics of a building in an adequate way. In a classic article, Alexander [Ale65] gives a number of strong arguments why many design problems are not tree-like. For instance the resulting trees do not always provide an meaningful grouping of the façade elements according to their mutual similarities and symmetries. Xiao et al. [XFT*08] already proposed to generalize the subdivision tree to a directed acyclic graph by merging shapes that are adjacent and similar.

In contrast, we offer the user a concept that is even more flexible: a separate group structure that is not constrained by the original subdivision hierarchy. While this is very flexible, defining a meaningful grouping is a non-trivial task and so-far there are no algorithms that accomplish it fully automatically. It requires higher-order knowledge, hence, in our approach we incorporate the abilities of the user in order to provide these semantic cues. To help the user to structure the façade in a reasonable manner we introduce the concept of *coherence-based modeling*. In this context we define the term coherence as follows: regions that contain structural elements (e.g., windows) that can be separated by a common guillotine cut are considered as coherent. This does not imply that these regions are necessarily spatially adjacent nor that they exhibit a well defined symmetric pattern.



As an example consider the façade presented in Figure 2: This façade does not fit into a simple grid pattern, but it can be separated by 4 guillotine cuts into 7 distinct sub-regions (a), which results in a simple *shape tree* as shown in the figure to the left. The sub-elements contain partially coherent structures, in particular the windows, and incoherent structures, like the portal (shape (**F**)), which does not fit into the arrangement. Also the entire block (**A**) is not coherent with the rest since it contains windows which differ in size and appearance.

To further split the façade into rows and columns, regions (**B**), (**C**), and (**D**) are considered coherent and form a coherence group ($\mathbf{B} \cup \mathbf{C} \cup \mathbf{D}$). The same applies to the spatially disconnected but coherent elements (**E**) and (**G**), which form ($\mathbf{E} \cup \mathbf{G}$) as shown in Figure 2. Now, each grouping can be split (using manual or automatic tools) while keeping horizontal coherence. Note that for automatic splits, described in detail in Section 4.1, disconnected image pieces are combined to one *meta-image* that is then processed as one part.

The newly generated elements are children of the respective shapes mentioned above: ($\mathbf{B}'$) are children of (**B**), ($\mathbf{E}'$) are children of (**E**), and so on. Shapes containing the wall between the floors are children too, but since they do not contain further structure, we do not consider them anymore. Elements containing structure (windows) can now again be combined such that vertical coherence among them can be exploited: ($\mathbf{A}'$), ($\mathbf{B}' \cup \mathbf{E}'$), ($\mathbf{C}'$) and ($\mathbf{D}' \cup \mathbf{G}'$) form four vertically coherent regions, where each can be split vertically as a whole (c). Applying the (automatic or manual) vertical cuts results in a separation of the particular windows (d). These can be grouped into a so-called *sync-group* and processed in a synchronized manner, such that only one instance has to be edited. We discuss this in more detail in Section 4.2.

Finally, note that the groups from step (c) overlap with groups from (b). This, in fact, is a key advantage of coherence-based modeling over previous methods, which define the façade as one rectilinear grid [MZWvG07, XFT*08] or as a hierarchical concatenation of such grids [SHFH11], The possibility to define groups composed of arbitrary elements combined with the coherence-constraints provides the user a very flexible yet well-defined toolkit for façade structuring.

4. Interactive Modeling Framework

In this section we describe the data structures and algorithms behind our system.

Shape: The basic building block of the modeling system is a *shape* S, which is defined as a rectangle with coordinates of its minimum corner (x,y) and its size (s_x,s_y). A shape can have an arbitrary number of attributes S_{a_i}, such as depth, material id, color etc, and it can belong to an arbitrary number of *groups*.

Shape Split: The main modeling operation is a horizontal or vertical *split operation* which splits a given shape along k split-lines (guillotine cuts) into $k+1$ child shapes. A single manual split is generated by clicking on a position in the selected shape (or group). Auto-splits are generated by clicking on the selected shape (or group); their number and locations are determined automatically (cf. Sec. 4.1).

Shape Tree: Repeated split operations on shapes create a *shape tree*. This tree completely describes the current subdivision and modeling state.

Group: A group is a set of shapes $\mathcal{S} = \{S_1,...,S_n\}$, regardless of their position in the shape tree. It can be built automatically (Section 4.1), or by user *selection* (Section 4.3). A group can be used to hold shapes with equal attributes (e.g., material id, depth, etc.), or to perform splits.

Group Split: Both manual and automatic splits can be performed across groups of shapes, like shown in Figure 2. A special case are *synchronized groups*, described in detail in Section 4.2, where splits are performed across shapes which are not naturally aligned in the spatial domain.

4.1. Automatic Façade Split

In order to automatically detect split-lines in façade images, one could rely on finding edges in the image subset under consideration. However, traditional edge detection is a local operation and thus sensitive to noise and parameter adjustments, so that too few or too many edges are easily generated. In our façade splitting operation, on the other hand, the partition is built with the *content* (interior) of the image, not with the *gradient* (as in edge detection methods).

Let us assume that we want to split an image with horizontal lines – for vertical splits, the following works directly on the transposed image. In cases where the underlying regions are spatially disconnected, like group $(\mathbf{E}\cup\mathbf{G})$ in Fig. 2, a meta-image is composed out of the pieces. Next, we interpret the pixel rows of the image as *feature vectors* (we concatenate the rgb-color channels to one vector), and perform *unsupervised clustering*, where the number of clusters is specified by the user. Simple clustering, however, usually leads to an oversegmentation because it does not take spatial locality into account. We therefore employ a *regularization* in the spatial domain that corrects the initial guess under the consideration of spatial coherence. A similar approach was first used by Zabith and Kolmogorov [ZK04] to segment images in a spatially coherent way using the alpha expansion algorithm. Since our method is reduced in the domain dimensionality (a 1d sequence of rows) we use the *dynamic programming* (DP) algorithm to determine the global optimum. Figure 3 depicts a schematic overview of this procedure, which we now describe in more detail.

Preprocessing. Since global illumination gradients or shadowing can adversely affect the vectors, we carry out a simple pre-filtering technique: For each channel of the input image I, we determine its low-frequency image G by filtering I with a Gaussian with large σ. We use a kernel of the size of $\frac{w}{2}\times\frac{h}{2}$ and $\sigma = \frac{\max(w,h)}{10}$. Then we subtract the low-frequency signature from the original: $I_{new} = I - G$. This procedure straightens the signal such that undesired global illumination gradients do not affect the clustering (cf. additional material).

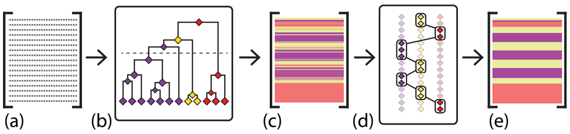


Figure 3: *Overview of the clustering procedure: (a) the input data as row feature vectors is (b) clustered by forming a linkage tree. The result of this stage (c) is regularized by dynamic programming (d), which build spatially coherent shapes (e).*

Clustering and Regularization. Having well-defined feature vectors $\mathbf{x}_i$, we use *hierarchical agglomerative clustering* [HTF09] in order to segment the image. We first compute the mutual distances of all point pairs using the normalized correlation coefficient distance: $d(\cdot,\cdot) = 1 - |\cdot,\cdot|_{\text{NCC}}$. Then we perform bottom-up agglomerative hierarchical cluster merging, which forms a linkage tree. We compare the clusters against each other using a recursive weighted average distance of two clusters. In particular, if a cluster c_p has been generated by merging two clusters c_s and c_t, then its distance $d(c_p,c_q)$ to an other cluster c_q is the average of the distance of c_q to the parent clusters c_s and c_t:

$$d(c_p,c_q) = \frac{1}{2}\left(d(c_s,c_q) + d(c_t,c_q)\right) .$$

In practice, we usually look for very few clusters, e.g., $k=2$ for a wall-window separation. We obtain the final separation by finding the smallest depth in the linkage tree at which a cut leaves a set $\mathcal{C} = \{c_1,\ldots,c_k\}$ of k cluster representatives. However, this clustering does not guarantee that the labeling is spatially coherent, in the sense that neighboring rows often have different labels, as illustrated in Figure 3 (c). This is due to the fact that neither the used metric nor the clustering algorithm itself take the spatial distribution of pixel rows into account.

In order to address this, we interpret the problem as a 1d labeling task, in which a smoothness term penalizes incoherent labelings of adjacent rows. In particular, 1d labeling can be efficiently solved using *dynamic programming*. Let $\mathcal{F}$ be a set of k labels and $f_x \in \mathcal{F}$ be the label assigned to row x. The dynamic programming algorithm solves for the optimal labeling by minimizing an energy E over all possible label assignments:

$$E(f) = \sum_{x_i \in \mathcal{X}} E_d(f_{x_i}) + \lambda \sum_{(x_i,x_j)\in\mathcal{N}} E_s(f_{x_i}, f_{x_j}) \quad \longrightarrow \text{min},$$

where λ expresses the relative weight of the two terms, $\mathcal{X} = \{x_1, \ldots, x_n\}$ is the set of all feature vectors, and $\mathcal{N} = \{(x_i,x_j) \big| (j-i) = 1\}$ is the set of all adjacent pairs of the vectors. The data term E_d expresses the cumulative distance of vectors x to their assigned cluster representatives $c_{f_x} \in \mathcal{C}$:

$$E_d\left(f_x\right) = d\left(x, c_{f_x}\right),$$

and the smoothness term E_s promotes spatial coherence for the cluster labels of adjacent rows:

$$E_s\left(f_{x_i}, f_{x_j}\right) = \begin{cases} 1 & \text{if } f_{x_i} \neq f_{x_j} \\ 0 & \text{otherwise.} \end{cases}$$

In DP terms, each row x_i corresponds to a stage, and each label $f\left(x_i\right)$ to a state. According to the *optimality principle*, we solve this problem by sequentially accumulating the energy over the stages followed by a reverse backtracking of a minimum energy path. Figure 3 (d) shows a schematic example with three labels per row and also indicates the optimal labeling calculated using dynamic programming as in [VC08].

4.2. Synchronized Modeling

The second fundamental concept in our modeling system is that of a *synchronized group*. Such a group describes a subset of the façade image pixels. Therefore, there are two restrictions when building such a group: (1) any coordinate on the façade may only be represented once in the group. (2) Each shape in the group has to have matching *topology*, i.e., the same number of children and the same number and arrangement of splits (cf. Figure 4). The *size* of the shapes is not relevant.

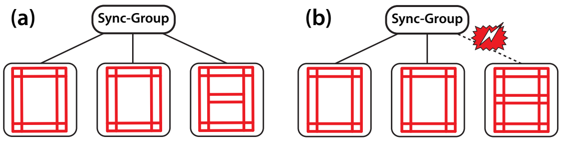


Figure 4: *Examples of sync-groups. A valid sync-group in (a), and an invalid sync-group in (b), where the right element has different topology at the same level.*

The first restriction is enforced by allowing only leaf shapes at grouping time. Then shapes can be sync-split and the splits are propagated to all shapes such that their relative positions with respect to their extent match each other. To fulfill the second restriction, if a grouped shape is split separately, the group remains if the split topology on the same group level is kept (cf. Figure 4, a), otherwise, the group is released (cf. Figure 4, b).

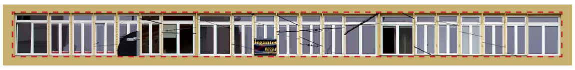


Figure 5: *The red-doted line indicates a meta-image generated over all windows of the façade shown in Fig. 1.*

Once a sync-group has been built, the following modeling operations can be performed on all shapes in the group in a "synchronized" fashion:

Manual Split (vertical or horizontal): By clicking at a position in one element of the group, each shape in the group is split at the corresponding *relative* position.

Automatic Split: The automatic façade split also takes the underlying image into account. In order to do so, we again create a meta-image which contains all shapes of the group arranged in an appropriate way for the following operation, like shown in Figure 5. The automatic split is then carried out on this meta-image, though this operation is often limited by too low resolution and noise in the underlying image.

Moving Splits: Since the topology of each group member matches, the user can select a split line in one shape, move it, and the corresponding split line in the other shapes will be updated to the same relative position.

Removing Splits: Two adjacent child nodes of a shape can be merged into one node by removing the split line between the two shapes. In synchronized mode, this operation will be carried out simultaneously for all shapes in the group.

Note that all operations still work on the individual shapes in the group independently. This means that a synchronized split will create a split in *each* shape in the group. This has the advantage that individual shapes can still be customized after a synchronized editing operation (see Figure 6).

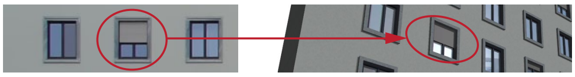


Figure 6: *After synchronized editing, each shape can be adjusted individually, like the window-shutter.*

4.3. Selecting Relevant Shapes

The splitting procedure provides a hierarchy of shapes. In order to perform editing the user needs to access shapes to determine what to group. We provide the following mechanisms to select relevant shapes (cf. Fig. 7):

Single Select (a): Clicking on a shape selects the shape itself. Clicking on a shape multiple times selects the successive parent shapes in the shape tree.

Multi Select (b): Clicking on a shape with the selection modifier (the CTRL key in our application) extends or reduces the selection by the clicked shape.

Cluster-Id Select (c): Clicking on a shape with the cluster modifier (the CTRL+ALT key in our application) selects all siblings of the shape in the shape tree that have the same cluster-id as the originally selected shape. The cluster-id is assigned to each shape in the automatic façade split stage (Section 4.1), thus it only works on regions split automatically. Repeating the above operation goes up in the shape tree in the search for the same cluster ids. This corresponds to selecting all "cousins" with the same cluster id for the next click, and so on. When the root of the shape tree is reached, *all* shapes with the same cluster id are selected.

Sync-Group Select (d): When a sync-group is currently activated, also the selection operation works in *synchronized mode*. This means that when the user clicks on one child of a shape using the group modifier (the SHIFT key in our application), then all corresponding children in the other shapes in the group are selected as well.

Group Subtraction/Addition (e): Selections saved in groups can be combined or subtracted from each other (for example, subtracting the glass elements from the window tiles to get the window frames).

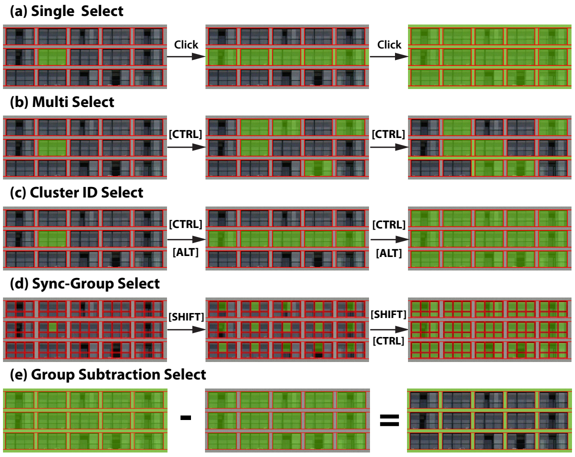


Figure 7: *Overview of the selection modi.*

Different modifiers can be combined, for example, multi-selection and sync-group selection shown in Figure 7 (d). While the number of different options appears large, in fact, a trained user is able to use them very fast and intuitively. We demonstrate these selections in Figure 7 and show their application in the accompanying video.

4.4. Approximating Complex Shapes

Usually, very fine detail is created by synced, but manual splits. At this point it might be difficult to express more complex elements of the façade by axis-aligned splits. In order to relax this constraint, we allow the user to add polygonal shapes at the bottom level of the shape tree. Each leaf node of the tree can contain a set of polygons, which can also be edited in synchronized manner over a grouped set of terminal nodes (Figure 8).

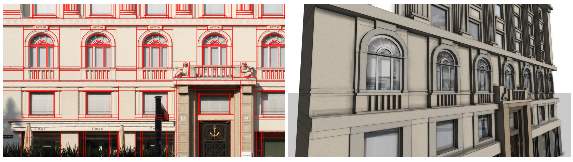


Figure 8: *Complex objects can be approximated by general polygons at the leaf-level of the shape tree.*

5. Evaluation

We have implemented the proposed interactive modeling workflow and used it to model a variety of façades with different properties. We evaluate our method using two test methodologies: a modeling test with an expert user to compare different modeling approaches; and a user study with novice users to evaluate the ease of use of our method.

5.1. Modeling Test

We compare five different modeling modes that are characterized as follows:

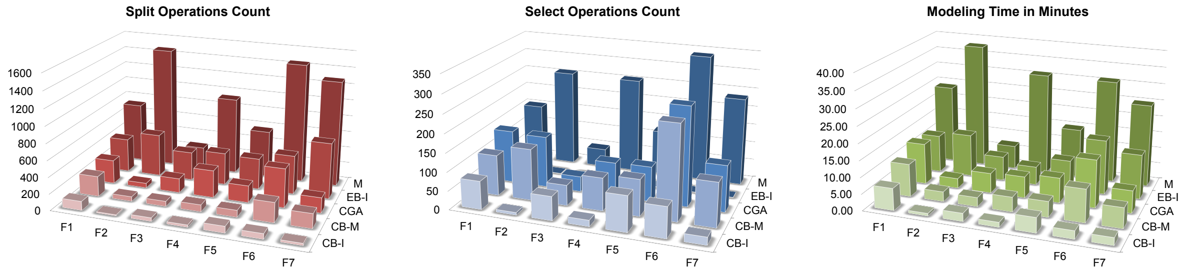
Manual (M): Hierarchical splits using our tool. All splitting lines are set manually; no coherence, no sync groups, no auto-split and no auto-selection.

Edge-Based-Interactive (EB-I): We have implemented the edge-driven method as described by Xiao et al. [XFT*08]. We implement the pipeline described in Sections 6.1 (edge detection), 6.2 (subdivision), and 6.3 (repetitive patterns/template matching) of that paper. Then we import the results of the automatic subdivision into our GUI and perform manual correction of the results using add-segment and remove-segment operations (both counted together in the table in Figure 9), practically equivalent to the operations described in Section 6.4 (interactive refinement) of [XFT*08]. For template matching, we allowed the user to indicate repetitive patterns like windows using a bounding rectangle and matched this over the façade using NCC, as described in [XFT*08].

CGA-Based (CGA): Hierarchical splits using our tool with all splitting lines set manually, and a subset of coherence-based editing tools is allowed in cases where elements are totally identical. In this test we basically manually simulate the CGA-grammar-based modeling scheme (but not the automatic method itself) of Müller et al. [MZWvG07].

Coherence-Based-Manual (CB-M): Manual splits and all coherence-based editing tools. All split lines are set manually but all coherence-based sync-splits are allowed. No auto-selection.

Coherence-Based-Interactive (CB-I): The same as previous but including auto-split and auto-selection, coherence split and sync split scheme. This is our full method.



Elements		M			EB-I			CGA			CB-M			CB-I		
		Edit	Select	Time	Edit	Select	Time	Edit	Select	Time	Edit	Select	Time	Edit	Select	Time
F1	960	725	153	24.00	411	1	11.50	288	142	12.50	246	109	10.00	110	76	6.75
F2	4351	1470	258	38.00	518	1	13.00	56	138	3.00	56	138	3.00	13	7	0.80
F3	308	262	49	7.20	348	1	7.40	174	41	6.10	63	55	2.80	44	64	2.50
F4	1742	936	256	31.00	392	1	5.60	333	92	6.50	84	88	4.50	27	17	1.50
F5	965	572	121	15.00	378	1	8.90	206	94	7.60	84	98	4.80	81	95	4.60
F6	1405	1448	339	31.00	470	1	16.50	474	265	14.50	243	252	10.00	66	82	2.50
F7	1858	1285	235	25.00	678	1	13.50	186	124	6.80	178	118	6.20	26	21	2.30

Figure 9: *Comparison of the number of split-operations (which include add and remove split-line), selection operations, (which include grouping) and of the modeling time. Note, there are no selections in the edge-based (EB-I) method.*

The purpose of the test is to answer the following two questions: (1) how much faster is our coherence-based method compared to others, and (2) how much does the automatic support help the user. In order to measure these dependencies, we use a metric which counts the number of mouse clicks performed with particular operations. The modeling operations we take into account are **(1) split-ops**: all operations which change the number of splits or shapes or their locations in the façade. These are in particular add-split, delete-split, move-split. Also our auto-split is a modeling-op, but it counts only once for each click. **(2) Selection-ops**: all clicks which are done to select and group shapes in order to perform some operations, like the coherence-split. Finally we measure the total **(3) modeling time**.

All the tests are performed under the same conditions and the resulting models are of the same level of detail (i.e. the number of shapes is similar). Modeling is done using our tool by a well-trained user (system expert) who already knows the test data sets. This ensures that we measure only the pure modeling time, not the time which is needed to think about how to divide the façade or how to use the GUI. The tested façades are shown in Figure 10.

Note that for Façades F1, F2, F3, F5 and F7, we did not take the ground-floor modeling time into account, since this part requires mostly manual modeling, which takes equally long for all of the tests. For the same reason, we do not measure the material and depth assignment times for all objects either. Thus, the number of elements and the total times reported in the table correspond to the 5th column (e) in Fig. 10. Values reported in column (f) are partially higher since these incorporate time for creation of additional detail (e.g., ground floor, ornaments), material groups and depth values.

Figure 9 shows the statistics and a graphical interpretation of the modeling test. It is evident that the major speedup comes from exploiting coherence, which of course depends on the repetitiveness of the façade. The second conclusion is that automatic splitting provides significant speedups on façades with many repetitions of the same objects: usually windows. This is especially evident by F2, but also others, like F4, F6 and F7 are sped up considerably using coherence, even compared to the CGA-like split scheme.

The edge-based-interactive method including template matching is consistently outperformed by the coherence-based methods, even if our automatic split is not used. The reason is that this method provides a vast number of misplaced segments that need to be manually removed. Even if identical windows can be grouped by the (notoriously unreliable) NCC-matcher, which provides a certain speedup, this task is still time-consuming, since it requires the user to instantly search for erroneously placed segments. Thus, we have found that cleaning up the splitting is even more tedious than starting with a fresh subdivision from scratch, which gives the user the possibility to add structural high-level knowledge.

5.2. User Study

Furthermore, we have also conducted a user study to check the response of novice users to our system. We have tested on 8 participants who were completely unexperienced with our tool. They were first instructed and supervised for about 15 minutes, then were allowed further 15 minutes to practice with the tool on one façade. After that we asked the participants to model three new façades which they had not seen before. Their goal was to reach a similar level of detail as in the previously supervised session in a time of maximally 5 minutes. All participants were asked to repeat the process three times in order to measure their learning progress.

Figure 11 shows the performance of the (novice) participants compared to an expert who was well trained in our system and who knew the data sets a priori. It can be seen that the ex-

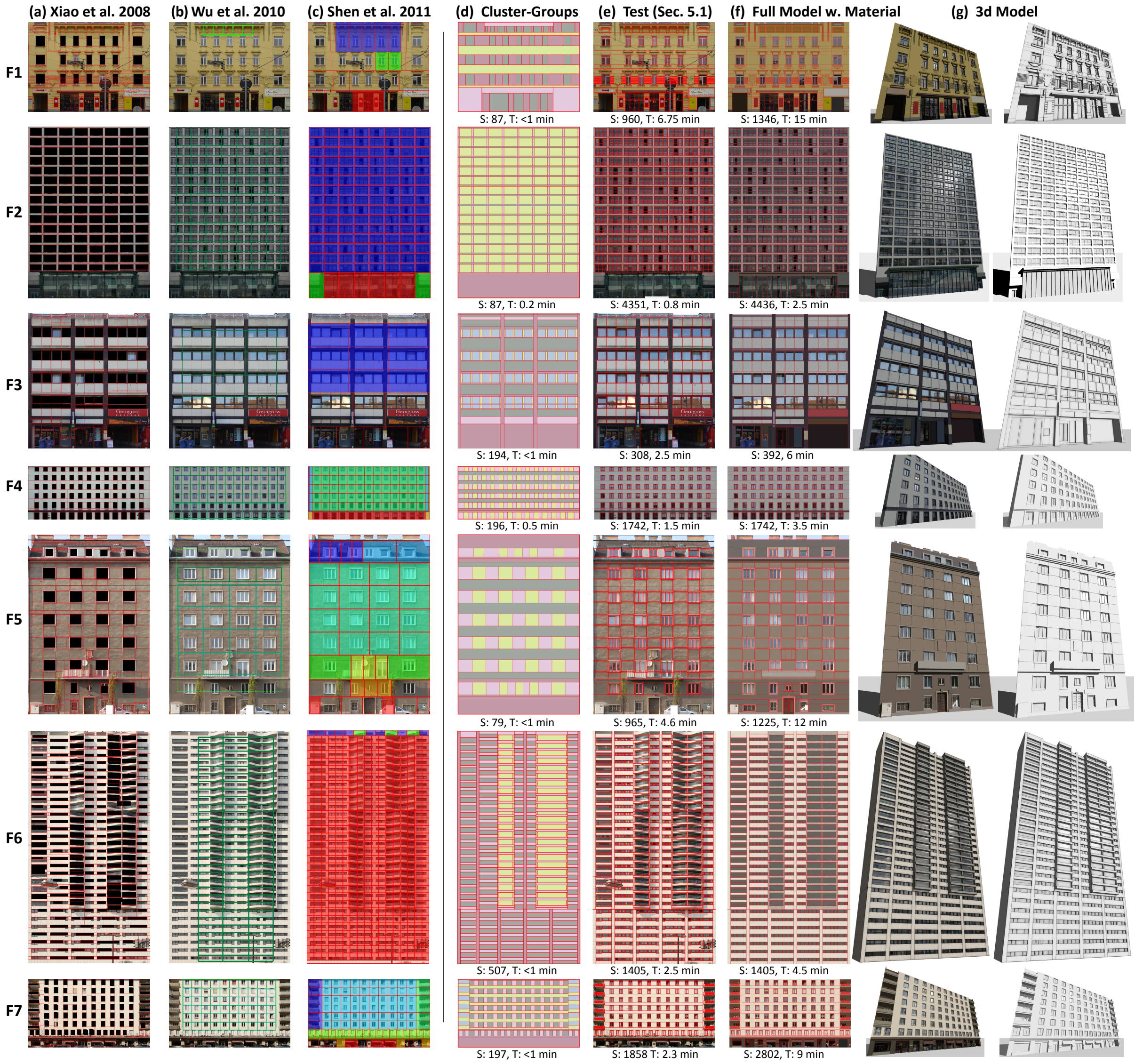


Figure 10: *From left to right: (a) our implementation of [XFT*08]. Note the holes in the windows mean that these windows have been previously recognized by the NCC matcher and have been grouped. Next, (b) results of [WFP10], followed by (c) results of [SHFH11]. Now follow our results: (d) elements grouped by cluster-id at a coarse level after the first few auto-split clicks, (e) results from the modeling test in Section 5.1 (note that for F1,F2,F3,F5 and F7 we did not take the ground-floor modeling time into account.), (f) our detailed results including ground floors, material, and depth assignments, and (g) finally a 3d rendering. S denotes the number of shapes and T the modeling time (figure best seen in closeup in the electronic version).*

pert needs a similar number of split operations, but performs them in less time and achieves slightly more detailed models. Nevertheless, the performance of novice users is in an acceptable range, especially compared to the manual modeling results from the previous test.

5.3. Visual Comparison

Figure 10 shows a number of models created using our system, compared to the results of others. All input images were taken using a handheld digital camera, rectified using a homography from four points and resized to about 1.0 MP.

First column (a): our implementation of [XFT*08], as described in Section 5.1. **Second column (b)**: results of the software of [WFP10]. The **third column (c)** shows the results of the recent method of [SHFH11] provided courtesy of the authors. The **forth column (d)** shows the results of our modeling approach after few initial clicks with manual

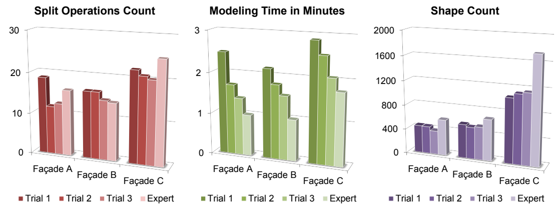


Figure 11: *Results of the user study conducted on 8 participants. The charts compare the performance of novice users (in three trials on each façade) versus a system expert.*

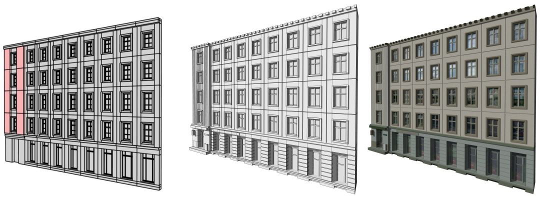


Figure 12: *Comparison of the results of the method of [MZWvG07] (left, 409 shapes) to our method (middle and right, 1,878 shapes). Left image courtesy of Müller et al.*

and automatic split-tools. Further, the **fifth column (e)** contains the results of the modeling test described in Section 5.1. Finally, the **sixth column (f)** shows the models with additional details (ground floors, ornaments, etc.) and materials and depth applied, and the **seventh column (g)** a 3d rendering. Note that the results in the first three columns have been obtained automatically, thus a direct comparison to our method should be regarded with caution.

6. Discussion

Limitations. The robustness of the automatic façade split is limited by the quality of the input image. If the image has too low resolution, strong image gradients, strong occlusions reflections, or ambiguous regions that cannot be separated using clustering, the method falls back to manual editing.

At this point the advantage of our method over automatic approaches is that the user can correct an error immediately as it happens: either by using the UNDO-function and providing new, manual splits, or by moving, removing or adding additional splits. Thus, such a manual intervention is well integrated into the modeling workflow.

Another limitation is the generally missing depth. Since our system is currently purely single-image, the depth contained in the images has to be deduced by the user. In general, if multiple input images are available, depth from stereo reconstruction (e.g., as in [XFT*08]) could be added and then refined by editing operations. Also a combination with LiDAR would be possible [LQS*11], but we consider this problem currently as out of scope.

High Detail vs. Low Detail, Interactive vs. Automatic. The geometric detail is an essential attribute for façade reconstruction. In this paper we aim for high detail. The reconstruction by Müller et al. shown in Figure 12 has 409 shapes (counting a matched window as one shape), and is comparable in detail to the results of Xiao et al. [XFT*08]. For example, processing the façade in Figure 12 by an automatic window and door classifier [TKS*11] would result in around 190 shapes. A similar number would be obtained by the fully automatic method by Xiao et al. [XFZ*09], as the segmentation is coarser than [XFT*08]. The same applies to other recent methods [WFP10, SHFH11] that provide even coarser subdivisions (cf. Fig. 10).

Our result has 1,878 shapes (Figure 12), which is about an order of magnitude higher. We believe that interactive modeling is the key to achieving this level of quality and detail, which is required by many applications. In general, we believe it is a future research direction to study how to best integrate automatic and interactive methods in order to improve the performance of the modeling process and the quality of the results.

Global Optimization vs. Individual Elements. Another open question is whether it is desirable to globally optimize the entire façade structure in order to set all instances of the same object to exactly the same size, or to keep the local variations of each particular element.

7. Conclusions

This paper has presented an interactive modeling framework for creating high-quality, detailed 2.5d models of building façades from single input images. The main contribution is the combination of automatic and user-driven methods according to their strengths: a human can quickly recognize the main façade structure even for very irregular façades, while our segmentation-based automatic façade split operation helps to quickly partition the façade into its components. We have evaluated the approach in a user study and show that both modeling time and modeling operations compare favorably to previous work.

There are two key insights of the evaluation: (1) When segmenting façades, it is not a good idea to rely too much on splitting heuristics. Instead, cues like symmetry [MZWvG07] or coherence, as proposed here, should be incorporated. (2) It is not efficient to first use an automatic algorithm and try to clean up the errors. That usually takes longer than starting from scratch and doing everything manually. Indeed, starting with an automatic method and then refining the results even increases the actual load of manual work: because potential errors are distributed arbitrarily over the façade, they have to be first localized and then subsequently fixed. For this reason the user spends a lot of time locating errors and thinking how to bring the current splitting into a reasonable subdivision.

In contrast, our proposed coherence-based cutting scheme allows the user to decompose the façade into well-defined regions in advance. The time of the user is now invested into thinking about how to divide a non-trivial façade into a meaningful structure. This is a difficult task and it has not been covered by automatic façade subdivision approaches.

We believe that with the plethora of research in automatic computer vision algorithms, it will become equally important to study the efficient integration of automatic processing and user interaction.

Acknowledgements

The authors thank Georg Stonawski for supporting this project and Franz Spitaler for his tireless efforts in implementing the rendering frontend. This project has been partially funded by FFG (825842-FIT-IT-VC) and NSF (CCF 0811790 and CCF 0643822).

References

[AD04] Alegre F., Dellaert F.: A Probabilistic Approach to the Semantic Interpretation of Building Facades. In *International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres, 2004* (2004). 2

[Ale65] Alexander C.: A city is not a tree. In *Architectural Forum* (1965), vol. 122, pp. 58–62. 3

[ARB07] Aliaga D. G., Rosen P. A., Bekins D. R.: Style grammars for interactive visualization of architecture. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 786–97. 2

[DTC04] Dick A., Torr P. H. S., Cipolla R.: Modelling and Interpretation of Architecture from Several Images. *International Journal of Computer Vision 60*, 2 (Nov. 2004), 111–134. 2

[DTM96] Debevec P. E., Taylor C. J., Malik J.: Modeling and rendering architecture from photographs. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96* (1996), pp. 11–20. 2

[HTF09] Hastie T., Tibshirani R., Friedman J. H.: *The elements of statistical learning: data mining, inference, and prediction*, 2 ed. Springer, 2009. 4

[JTC09] Jiang N., Tan P., Cheong L.-F.: Symmetric architecture modeling with a single image. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 1. 2

[KR07] Korah T., Rasmussen C.: 2D Lattice Extraction from Structured Environments. In *2007 IEEE International Conference on Image Processing* (2007), vol. 2, IEEE, pp. II – 61–II – 64. 2

[KST*09] Koutsourakis P., Simon L., Teboul O., Tziritas G., Paragios N.: Single view reconstruction using shape grammars for urban environments. In *2009 IEEE 12th International Conference on Computer Vision* (Sept. 2009), IEEE, pp. 1795–1802. 2

[LN04] Lee S. C., Nevatia R.: Extraction and integration of window in a 3D building model from ground view images. In *2004 IEEE Computer Vision and Pattern Recognition, 2004. CVPR 2004.* (2004), vol. 2, IEEE, pp. 113–120. 2

[LQS*11] Li Y., Qian Zheng, Sharf A., Cohen-Or D., Chen B., Mitra N. J.: 2D-3D Fusion for Layer Decomposition of Urban Facades. In *2011 IEEE 13th International Conference on Computer Vision* (Nov. 2011), IEEE, p. to appear. 2, 9

[LWW08] Lipp M., Wonka P., Wimmer M.: Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 1. 2

[MLS*10] Musialski P., Luksch C., Schwärzler M., Buchetics M., Maierhofer S., Purgathofer W.: Interactive Multi-View Façade Image Editing. In *Vision, Modeling, and Visualization (VMV)* (2010), pp. 131–138. 3

[MRM*10] Musialski P., Recheis M., Maierhofer S., Wonka P., Purgathofer W.: Tiling of ortho-rectified facade images. In *26th Spring Conference on Computer Graphics - SCCG '10* (New York, USA, May 2010), ACM Press, p. 117. 2

[MWH*06] Müller P., Wonka P., Haegler S., Ulmer A., van Gool L.: Procedural modeling of buildings. *ACM Transactions on Graphics 25*, 3 (July 2006), 614. 2

[MZWvG07] Müller P., Zeng G., Wonka P., van Gool L.: Image-based procedural modeling of facades. *ACM Transactions on Graphics 26*, 3 (July 2007), 85. 2, 3, 6, 9

[NSZ*10] Nan L., Sharf A., Zhang H., Cohen-Or D., Chen B.: SmartBoxes for interactive urban reconstruction. *ACM Transactions on Graphics 29*, 4 (July 2010), 1. 2

[SHFH11] Shen C.-H., Huang S.-S., Fu H., Hu S.-M.: Adaptive partitioning of urban facades. *ACM Transactions on Graphics 30*, 6 (Dec. 2011), 1. 1, 2, 3, 8, 9

[SSS07] Snavely N., Seitz S. M., Szeliski R.: Modeling the World from Internet Photo Collections. *International Journal of Computer Vision 80*, 2 (Dec. 2007), 189–210. 2

[SSS*08] Sinha S. N., Steedly D., Szeliski R., Agrawala M., Pollefeys M.: Interactive 3D architectural modeling from unordered photo collections. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 1. 2

[TKS*11] Teboul O., Kokkinos I., Simon L., Koutsourakis P., Paragios N.: Shape grammar parsing via Reinforcement Learning. In *CVPR 2011* (June 2011), IEEE, pp. 2273–2280. 2, 9

[TSKP10] Teboul O., Simon L., Koutsourakis P., Paragios N.: Segmentation of building facades using procedural shape priors. In *2010 IEEE Computer Vision and Pattern Recognition* (June 2010), IEEE, pp. 3105–3112. 2

[VAW*10] Vanegas C. A., Aliaga D. G., Wonka P., Müller P., Waddell P. A., Watson B.: Modelling the Appearance and Behaviour of Urban Spaces. *Computer Graphics Forum 29*, 1 (Mar. 2010), 25–42. 2

[VC08] Velho L., Carvalho P. C. P.: *Mathematical optimization in computer graphics and vision.* Morgan Kaufmann, 2008. 5

[WFP10] Wu C., Frahm J.-M., Pollefeys M.: Detecting large repetitive structures with salient boundaries. In *Proceedings ECCV 2010* (2010), vol. 6312, pp. 142–155. 1, 2, 8, 9

[WWSR03] Wonka P., Wimmer M., Sillion F., Ribarsky W.: Instant architecture. *ACM Transactions on Graphics 22*, 3 (July 2003), 669. 2

[XFT*08] Xiao J., Fang T., Tan P., Zhao P., Ofek E., Quan L.: Image-based façade modeling. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 1. 2, 3, 6, 8, 9

[XFZ*09] Xiao J., Fang T., Zhao P., Lhuillier M., Quan L.: Image-based street-side city modeling. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 1. 1, 2, 9

[ZK04] Zabih R., Kolmogorov V.: Spatially coherent clustering using graph cuts. In *Proceedings of the 2004 IEEE CVPR 2004.* (2004), IEEE, pp. 437–444. 4

[ZSW*10] 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 Transactions on Graphics 29*, 4 (July 2010), 1. 2