Is there a Procedural Logic to Architecture?

Julien Weissenberg   Hayko Riemenschneider   Mukta Prasad
Luc Van Gool
Computer Vision Lab, ETH Zurich
{julienw,hayko,mprasad,vangool}@vision.ee.ethz.ch

Abstract

Urban models are key to navigation, architecture and entertainment. Apart from visualizing façades, a number of tedious tasks remain largely manual (e.g. compression, generating new façade designs and structurally comparing façades for classification, retrieval and clustering).

We propose a novel procedural modelling method to automatically learn a grammar from a set of façades, generate new façade instances and compare façades. To deal with the difficulty of grammatical inference, we reformulate the problem. Instead of inferring a compromising, one-size-fits-all, single grammar for all tasks, we infer a model whose successive refinements are production rules tailored for each task. We demonstrate our automatic rule inference on datasets of two different architectural styles. Our method supercedes manual expert work and cuts the time required to build a procedural model of a façade from several days to a few milliseconds.

1. Introduction

In [15], Mitchell claims that architecture is structured by a certain logic which can be captured by formal grammars. Procedural modelling has extensively been used by architects, urban planners, the film and game industries, map makers and cultural heritage specialists to generate large-scale models of cities [26]. The introduction of procedural modelling has cut down the required amount of work to synthesize convincing models of cities, which used to take several man-years [17]. Procedural models are semantic and highly structured, and are very well-suited for simulations [2] and planning [11] compared to conventional 3D models. The main features of procedural models are that they are compact, editable, readable, semantic and advantageous for retrieval and fast graphics generation [10].

While whole virtual cities can be generated in minutes thanks to procedural modelling, it takes a lot more effort when it comes to constructing a model of an existing city. Existing inverse procedural modelling pipelines for example [25, 24, 14] require an expert to manually design a set of style-specific rules. Architects estimate that manual modelling takes them up to two days to model a single building —several years to model a city. In addition to the colossal amount of work involved, it also makes any update process very slow.

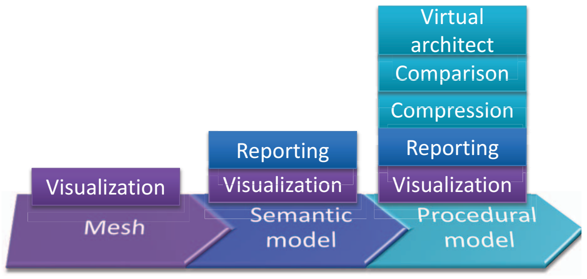

Figure 1. Pure mesh-based or semantic labelled models suffer from a limited field of uses. In this work we propose methods to automatically build procedural façade models in milliseconds for compression, comparison and new virtual façade creation.

The quality of 3D city modelling for visualization has dramatically improved over the past few years. Using Structure-from-Motion pipelines, works such as [28] have achieved a very high degree of realism. Yet, the resulting meshes are only good for visualisation. In parallel, semantic labelling has shown very encouraging results [13, 8]. However, the possibilities given by a set of labelled façades is limited compared to the set of applications offered by a procedural model (see Fig. 1). In this work, we transform a set of labelled façades into a procedural model, automatically. Additionally, we use the procedural models to generate new façade instances and compare façades layouts.

In a procedural model, rules are the backbone of the semantic information. They describe how architectural elements are grouped together and organised. The rules created for inverse procedural modelling should not only be able to generate a set of exemplar buildings in a style, but *specifically only* the possible buildings from a style. This poses the problem of finding a principled way of inferring such a rule set. As opposed to automatic inference, manual work takes a considerable amount of time and does not ensure consistency between the different models. Finally, the resulting rule set may contain a very large set of parameters, making the optimization intractable. To circumvent these problems, we propose a new formulation where the grammar is directly inferred from the labelling of a façade or a set of façades (see Fig. 2).

Our main contributions are: (1) an inverse procedural modelling pipeline where both rules and parameters are automatically inferred, (2) a method to measure the structural distance between façades for retrieval and clustering, (3) a method to synthesise new, non-existent façades from a model describing a set of façades in the same style, (4) a compression of the data by two orders of magnitude and a speedup of the processing time from days to milliseconds.

2. Related work

Façade modelling in form of image-based architectural modelling [1, 28] and semantic segmentation [3] is not relevant to our problem due to its restrictions to pure visualization and reporting. In fact, the work performed in image-base city modelling is a pre-processing step to our pipeline. Our goal is to use the procedural structure of façades for compression, comparison and virtual layout generation by exploiting shape grammars.

Shape grammars were intially proposed by Stiny *et al.* [23] as a generation tool for geometric paintings and sculptures. Later, Wonka *et al.* [27] and Müller *et al.* [18] proposed to use split shape grammars to describe architecture. The idea is to describe a building as a sequence of formal rules which gradually refine and add details to the model. Next we detail two main approaches that have been pursued to tackle architectural inverse procedural modelling using shape grammars, and then present related work in the field of grammatical inference.

The first inverse procedural modelling approaches assume that a grammar is given as input and infer the appropriate parameters to represent a given building [21, 24, 25]. These methods offer the advantage of systematically yielding architecturally sound buildings, but the actual correctness depends on whether the grammar can create the building at hand as an instantiation. Recent work relaxes the restrictions and use rather general architectural principles [13] or symmetries and repetitions [20] to infer the parameters.

In the second approach, both the rule set and the parameters are inferred. Inverse procedural modelling of L-systems has been tackled by Šťava *et al.* [22]. In [4], Bokeloh *et al.* produce procedural models of a mesh assuming it contains partial symmetries. Grammatical inference is also of interest in the analysis of natural language, music and genetics. In [9], de la Higuera *et al.* present a comprehensive survey about grammatical inference. In [6], Charikar *et al.* examine the smallest context-free grammar problem. They show that it is an NP-complete problem and compare grammar-based compression algorithms, such as [19].

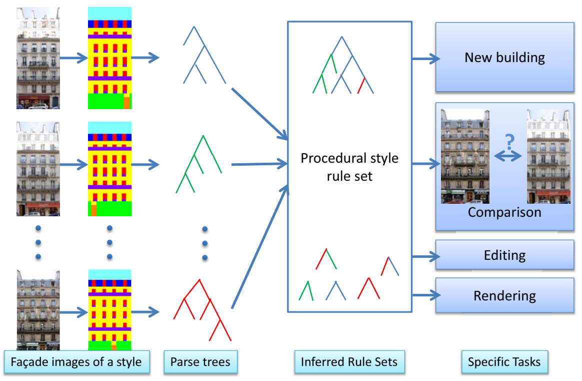

Figure 2. Our inverse procedural modelling pipeline. First, labellings of rectified façade pictures are the input to our method. A split tree is inferred for each façade. Those trees are combined and constitute a set of rules that describe the façade style. These inferred rules are then used for comparison, editing, virtual façade synthesis, and rendering.

However, to the best of our knowledge, grammatical inference has never been performed for more than one façade at a time and without any architectural style nor grid layout restrictions.

3. Approach

Our goal is to automatically derive a set of procedural rules to describe a set of façades of the same style, generate new ones and compare them for classification and retrieval. One or several Manhattan-world segmentations of façades are given as input, which can be obtained manually or automatically [3, 8, 13, 20].

Our weakly supervised algorithm for inverse procedural modelling provides as output, sets of style rules and for each façade, a set of parameters. We employ different representations for each different application.

3.1. Shape grammars for architecture

A shape grammar $\mathcal{G}$ is a context-free grammar consisting of a tuple $(\mathcal{N}, \Sigma, \mathcal{R}, \mathcal{S})$, where $\mathcal{N}$ is a finite set of non-terminal shapes, Σ is a finite set of terminal shapes, $\mathcal{R}$ is a finite set of production rules and $\mathcal{S}$ is a starting shape (axiom). A shape is enclosed in a 3D bounding box called scope. A terminal symbol shape can be a 3D model or a texture which we will refer to as asset (as in computer graphics terminology). An asset is hence an architectural element, such as a door, a window or a piece of wall. A production rule $r \in \mathcal{R}$ consists of a sequence of operations. An operation transforms a shape into other shapes. We consider *insert*, *horizontal* and *vertical* split operations. Insert operations insert an asset in the current scope, while split operations divide a shape into n newly created shapes. We distinguish between two kinds of split operations: binary and n-ary. We call a binary split a split between two elements such as

$$r_{binary} \rightarrow split(d)\{s_1 : r_1 | s_2 : r_2\} \tag{1}$$

where d is the splitting direction (horizontal or vertical), s_1 and s_2 are the sizes of the newly created shapes, and r_1 and r_2 their associated rules.

With shape grammars, a façade is described as a recursive splitting procedure from the root node (corresponding to the axiom $\mathcal{S}$) to the leaf nodes (each a single asset). Semantically higher level operations are compact notations for a series of lower level operations. Higher-level operations provide an *interpretation*, an *understanding* of the building. A number of implementations have been developed such as CGA [18], GML [12] and Teboul *et al.*'s [25].

Operations, and therefore rules, are parametric. Hence, a rule set plus a set of parameters generate a single façade. Altering one or the other will result in a different façade. We refer to a grammar instance as a set of shape grammar production rules plus a set of associated parameters.

3.2. What makes a good grammar?

The same façade can be represented by many different sequences of operations, therefore rule sets. Hence, we pay special attention to defining the properties a "good" grammar should fulfill. *The value of a grammar depends on its intended usage.* In the case of façade modelling, we consider visualization of an existing city, novel building generation, comparison between façades, reporting and compression. Bearing in mind the targeted applications, we propose to adopt the following criteria for the grammar generation. The grammar inference should be: consistent, fast, online and should produce an accurate, generative (within class), specific (between class) and compact grammar. Here, consistency means that two similar façade labellings will be described by two similar rule sets. By online algorithm, we mean that new façade instances can be added iteratively. This is a very important property, as in practice a city model needs to be constantly updated. Often, optimizing for one criteria will degrade the others. For instance, achieving Minimum Description Length (MDL) is likely to come at the cost of longer computations and the need to run an optimization over the whole dataset. Consequently, we propose to use different representations for different applications. The generative model, which creates new, virtual building instances for visualization, should be conservative and specific, as any error will be caught by the human eye. The analytic model, which can be used to compare buildings, should generalize such that it encompasses all façades from the style.

3.3. Façade parse tree generation

In this section, we describe the parsing algorithm which, starting from a labelling, encodes a façade as a binary split tree whose nodes correspond to façade regions, operations and parameters. The collection of trees is the starting point for all subsequent processing in Section 4. The parsing algorithm recursively splits the façade into smaller façade regions until all of them consist of a single asset. This can be seen as a top-down tree clustering of the façade.

Generating the split trees boils down to defining an energy function to select the split lines. The structure of the parse tree will solely be affected by the *order of the split*, determined according to this energy function. The rest of this section describes the energy function given in Eq. 3. Intuitively, the energy function aims at grouping assets which occur frequently next to each other and are not separated by a long edge. As the parsing is performed on labelled images, an edge designates a straight line that separates regions with different labels. The set of asset labels is $\mathcal{L}$.

In the rest of this section, d designates a direction, h means horizontal and v vertical. For each scope s, we start from a set of horizontal and vertical split line proposals $\mathcal{P}$ defined as

$$\mathcal{P} = \mathcal{P}_h \cup \mathcal{P}_v \tag{2}$$

where $\mathcal{P}_h$ is the set of horizontal split line proposals and $\mathcal{P}_v$ the set of vertical split line proposals. A line is a split proposal $p \in \mathcal{P}$ if a) it contains at least an edge collinear to the direction of p and b) it does not intersect with any edge perpendicular to p. The latter condition prevents from splitting across an asset. For simplicity, we only write the equations for horizontal split proposals, which can be easily be extended to vertical proposals. (x, y) refer to the coordinates of a pixel on a proposal line. For each split proposal $p_y \in \mathcal{P}$, we compute an energy function $f(p_y)$ using an edge length term e_x and an asset affinity v_d such that

$$f(p_y) = b^{(h)} b^{(p)} \left(\alpha \frac{1}{W} \sum_{x=0}^{W} e_y + (1-\alpha) \frac{1}{W} \sum_{x=0}^{W} v_d(x, y) \right) \tag{3}$$

where $b^{(h)}$ is a horizontal vs. vertical bias, $b^{(p)}$ is a parental bias, $\alpha \in [0, 1]$ is a weight between the edge support term and the affinity term, W is the size of the façade (the height respectively the width depending on the direction of p_y), $e_x = 0$ if a pixel y is an edge, 1 otherwise. The edge length term e_x rewards splitting along a longer line, while the affinity term v_d penalizes splitting between assets with stronger affinity. The biases are

$$b^{(h)} \begin{cases} \in]0,1] & \text{if } p \text{ is horizontal} \\ = 1 & \text{if } p \text{ is vertical} \end{cases} \tag{4}$$

$$b^{(p)} \begin{cases} \in]0,1] & \text{if } d(p) = d(a) \\ = 1 & \text{otherwise} \end{cases} \tag{5}$$

where $d(a)$ is the direction of the last accepted proposal (of the parent scope) and $d(p)$ the direction of p.

The second term sums over the affinity $v_d(x, y)$ between the asset at the edge pixel (x, y) and the asset at the nearest facing edge pixels $(x, \bar{y}^*)$, which is located according to

$$\bar{y}^* = \underset{\bar{y} \in \mathcal{P}_{\bar{\mathcal{A}}}}{\arg\min}(\Delta_{y,\bar{y}}) \tag{6}$$

where $\mathcal{P}_{\bar{\mathcal{A}}}$ is the set of split proposals which do not belong to the same asset. $\Delta_{y,\bar{y}}$ is the distance between proposals p_y and $p_{\bar{y}}$. The search space in Eq. 6 can be reduced by only considering edges located in the direction of the normal vector to the edge. The affinity $v_d(x, y)$ is defined as:

$$v_d(x, y) = c_d(l_{x,y}, l_{x,\bar{y}^*}) \tag{7}$$

where $l_{x,y}$ is the asset label at position (x, y) and $l_{x,\bar{y}^*}$ the asset label at the nearest facing edge.

The co-occurrence c_d between asset pairs, where d is the direction, is computed across all façades and stored in two affinity matrices C_h and C_v. These matrices C_d are normalized weighted co-occurrence matrix of size $|\mathcal{L}| \times |\mathcal{L}|$, where $|\mathcal{L}|$ is the number of asset labels. Each value in C_d is defined by

$$c_d(l_{x,y}, l_{x,\bar{y}^*}) = \sum_{i=1}^{\nu} \frac{|\ y_i - \bar{y}_i{}^*\ |}{W} \tag{8}$$

where ν is the total number of pixels belonging to split proposal lines, and the proximity measure is the normalized distance between a pixel belonging to a split proposal p_y and its nearest facing asset edge located along $p_{\bar{y}^*}$. In order to reduce the number of computations, the affinity coefficients are pre-computed for each pixel over the façade in both horizontal and vertical directions.

At the end of this parsing step, we obtain a binary tree of split operations that describes the façade. This tree can be represented as set of binary split rules (see Eq. 1). The whole process is exactly lossless, i.e. the original labelling can be re-generated from the tree of rules.

In the next section we show how to use these parse trees to optimize for compression, retrieval and virtual generation of new façades.

4. Optimization of Shape Grammars

Given the general method to construct a parse tree of the façade layouts, it is our goal to optimize the grammar with respect to its production rules for the each of following specific applications, separately.

1. **Compression** reduces the size of the grammar by examining redundancies within and between parse trees.
2. **Comparison** defines structural features to create a retrieval metric capturing differences in façade layout.
3. **Virtual façade synthesis** analysis examples of façade types and creates a new set of consistent parameters to instantiate a façade of the same style.

4.1. Grammatical inference

In the previous section, we presented a way to infer a parse tree depicting a given façade. However, the inferred tree consists only of binary *split* operations, i.e. single operation rules. The grammatical inference phase is a succession of steps to group those single operation rules to infer a shorter representation. Note that the inferred grammar provides us with an understanding of the building as lower level operations are combined into fewer number of higher level operations. For instance, repetitions of the same rule across the building are detected. In addition, rendering performance benefits from using a compact grammar as fewer rules need to be evaluated at render time. The grammatical inference is a two step process. First, the binary split nodes in each parse trees are converted into n-ary split nodes. Then, more complex production rules are inferred by comparing n-ary split nodes over all parse trees.

4.1.1 Transformation to n-ary split nodes

Nested binary splits in the same direction (horizontal or vertical) are re-written as n-ary splits following

$$\begin{array}{l} \left\{ \begin{array}{lll} r_i & \rightarrow & split(d)\{s_j : r_j | s_k : r_k\} \\ r_j & \rightarrow & split(d)\{s_{j_1} : r_{j_1} | s_{j_2} : r_{j_2}\} \end{array} \right. \\ \qquad\quad \Leftrightarrow \\ \quad r_c \quad \rightarrow \quad split(d)\{s_{j_1} : r_{j_1} | s_{j_2} : r_{j_2} | s_k : r_k\} \end{array} \tag{9}$$

Since the position and size of each scope remains identical, this transformation is guaranteed to be lossless. Each such transformation reduces the total number of rules $|\mathcal{R}|$ by 1 (two rules were turned into one). It is performed multiple times by recursively exploring the parse tree until no more occurrences can be re-written.

4.1.2 Production rule inference

The production rule inference is based on a similar principle as [5, 19]. We recursively replace repeating structures with a rule. When two or more similar nodes are found, those are re-written as a single parametric rule.

To perform this transformation, each pairs of the nodes are first compared. Two nodes are considered similar if they have the same operation and similar children, regardless of their numerical parameters (i.e. size values). Formally, as shown in Eq. 10, r_α and r_β can be re-written as a parametric rule r_ϕ such that

$$\begin{array}{ccl} \left\{ \begin{array}{l} r_\alpha \\ r_\beta \end{array} \right. & \begin{array}{c} \rightarrow \\ \rightarrow \end{array} & \begin{array}{l} split(d)\{s_1 : r_1|...|s_k : r_k\} \\ split(d)\{s_I : r_1|...|s_K : r_k\} \end{array} \\ & \Leftrightarrow & \\ r_\phi(\mathbf{x}) & \rightarrow & split(d)\{x_1 : r_1|...|x_k : r_k\} \end{array} \tag{10}$$

Now a single *split* rule is to be invoked by changing the values of the parameter vector $\mathbf{x} = (x_1, x_2, ..., x_k)$. This transformation is guaranteed to losslessly preserve the layout of the scopes.

After each transformation, the number of rules is changed by the reduction of repeated rules as

$$|\mathcal{R}|_t = |\mathcal{R}|_{t-1} + 1 - \theta(\psi + 1) \tag{11}$$

where $|\mathcal{R}|_t$ is the total number of rules at the iteration t, $|\mathcal{R}|_{t-1}$ the total number of rules before the transformation, θ the number of occurrences of the rule and ψ the number of child rules. From Eq. 11, we note that the more similar rules exist, the more efficient the rule inference will be at reducing the total number of rules. In order to produce similar rules, the split tree should be as consistent as possible. Also, the more child rules ψ, the smaller the total number of rules $|\mathcal{R}|_t$ with respect to $|\mathcal{R}|_{t-1}$.

Comparing two nodes in the tree implies comparing their children. In order to avoid traversing the tree multiple times, the nodes are compared in a bottom-up fashion. First, only nodes whose children are *insert* operation are compared and replaced by production rules if possible. Later, nodes whose children are *insert* or production rules are considered until it is not possible to create any new production rule. Note that the parameters of the child rules are carried over the parameter vector of the newly inferred rule.

4.2. Compression

The rule set $\mathcal{R}$ is optimized for the smallest number of rules following the Minimum Description Length (MDL) principle by solving for

$$\underset{\alpha, b_p, b_h}{\arg\min}(|\mathcal{R}|) \tag{12}$$

where $\mathcal{R}$ is a rule set describing the input façades inferred by the method given in Sect. 3.3 to Sect. 4.2 , α, b_p and b_h are the parameters defined in Eq 3.

4.3. Comparing façades

Retrieval and clustering are two application examples for comparing façades. More formally, comparing façades means finding an adequate distance function δ as

$$\delta : \mathcal{R} \times \mathcal{R} \rightarrow \mathbb{R}^+ \tag{13}$$

As all façades of the same style are similar and share visual features, using a feature-based distance would be inappropriate. We propose two different distance measurements. The first is based on an MDL paradigm, while the second is a powerset-based distance derived from the parse trees.

The MDL-based approach measures the similarity of façades in terms of their **common rules**. As [7] we define that two façades A and B are more similar than A and C if

$$\frac{|\mathcal{R}_\mathcal{A} \cup \mathcal{R}_\mathcal{B}|}{|\mathcal{R}_\mathcal{A}| + |\mathcal{R}_\mathcal{B}|} < \frac{|\mathcal{R}_\mathcal{A} \cup \mathcal{R}_\mathcal{C}|}{|\mathcal{R}_\mathcal{A}| + |\mathcal{R}_\mathcal{C}|} \tag{14}$$

where $|\mathcal{R}_\mathcal{A}|$ is the number of rules used to describe façade A after compression. We create a histogram of the frequency of rules shared by two façades and use a χ^2 as an appropriate distance. We refer to this distance as δ_{common}.

In the second approach, we compare the binary split trees. To this end, we consider the **powerset** $\mathcal{PS}(\mathcal{O} \cup \mathcal{L})$ of the set of operations $\mathcal{O}$ and asset labels $\mathcal{L}$. For each façade parse tree, we draw a histogram which reflects the number of the elements of $\mathcal{PS}(\mathcal{O} \cup \mathcal{L})$ and use a χ^2 as a distance measure. We refer to this distance as $\delta_{powersets}$.

4.4. Virtual façade synthesis

In this section, we show how to create new, non-existent façades from a set of real building façades in the same style. To build virtual cities, experts manually model a few typical buildings and relax their parameters by assigning ranges from which the parameters are randomly drawn. This approach is inspiring since it produces valid buildings as the structures are not altered, and yet it delivers a good illusion of diversity.

At the end of the production rule generation, we have a set of parametric rules, each dependent on one or more parameter vectors $\mathbf{x}$. Each façade of the input set is identified by a starting rule and a parameter vector. The starting rules correspond to rules that split the whole façade, generally into floors and balcony layout. Each of these starting rules will then call the hierarchy of rules describing the structure of the façade. The parameter vectors specifies all the sizes used in the rules. To generate a new façade, we instantiate the starting rule with a new set of parameters. By doing so, we sample the parameter space while preserving

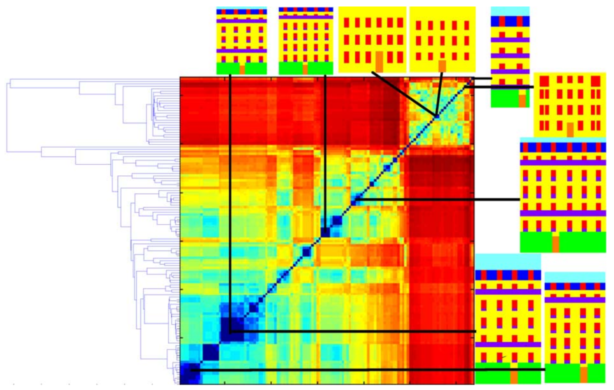
Figure 3. Distance matrix for the ECP2011 and Graz2012 datasets, re-ordered according to its dendrogram (log scale). Some of the associated labellings are shown on the right.

the structure. We have to make sure the new parameters will be consistent, i.e. it is important to preserve the correlations between the different vector variables. Assuming these parameters follow Gaussian distributions, the correlations are discovered by applying a PCA on the parameter vector set.

5. Evaluation

In this section, we evaluate the compression, façade retrieval and virtual façade generation. Further we discuss the losslessness and computational cost of the rule inference.

5.1. Experimental setup

In our setup we evaluate on two different datasets. First, the ECP2011 façades dataset [24] comprises 104 labelled images taken in rue Monge, a street in Paris. The architecture is representative for Haussmannian style, which is also found in other cities like Buenos Aires or Brussels. Second, a subset of Graz2012 [20] is selected, which consists of 30 annotated façades in Gruenderzeit style, which is widespread in Germany and Austria.

5.2. Losslessness and computational cost

Each step of the grammatical inference algorithm is perfectly lossless. In fact, one can regenerate the original labelled image by replacing the assets by colour patches.

As shown in Fig. 5, the computational cost of the inference algorithm (i.e. parsing, n-ary split compression, rule inference and data collection for statistics) is linear with respect to the number of input façades. This is a very important property, as it shows the inference algorithm scales up to the size of whole cities. Our implementation of the inference algorithm takes about 32 ms per façade on a single core of an Intel Core i7 930. Our method works online and is applicable in practice to model whole cites. A new façade can be added to the dataset at a linear cost. Finally, all steps in the inference algorithm can be parallelized as well.

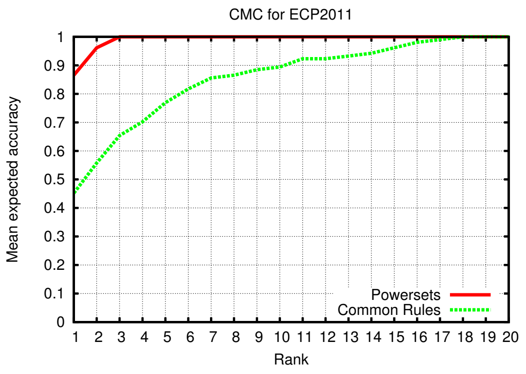

Figure 4. Cumulative Match Characteristics (CMC) for ECP2011 for semantic façade retrieval. The *powersets* distance better captures the structural similarities over *common rules* distance.

5.3. Compression

Our findings indicate that the parameters are robust for a large range. For example, for ECP2011 the minimal number of inferred rules is 68, and is found for all values of $\alpha \in [0.25, 1.0]$, $b_p \in [0.0625, 0.5]$ and $b_h \in [0.125, 1.0]$. In the results shown in this section, we use parameters $\alpha = 0.6$, $b_p = 0.25$ and $b_h = 0.5$.

The growth of the number of inferred rules with respect to the number of input façades is shown in Fig. 5 for ECP2011 and Graz2012. We can reduce the number of rules using compression by two orders of magnitude (notice the logarithmic scale). Especially, the more regular ECP2011 dataset shows a clear drop in rule growth. This shows that the core logic principles of the Haussmannian style can be explained after examining about 20 façades.

However, we also see that in addition to core principles within a style, *exceptions are the rule*. This translates to a continual growth of the number of rules in Fig. 5 when new façade samples are added. In the Graz2012 dataset, a large number of rules are ony used once. Further investigations would elucidate whether this stems from architectural diversity (needed for all applications) or annotation noise.

5.4. Façade comparison

The goal of façade retrieval is to compare a query facade to the set of known façades and determine the most similar ones. In our scenario we are not comparing appearance or the sizes of architectural elements, but the procedural layout of the façade. It is our goal to group façades which have the same layout in terms of floors, window columns, balconies and door placement.

Following the two distance measures δ_{common} and $\delta_{powersets}$ defined in Sect. 4.3, we evaluated a façade retrieval and clustering on the datasets[1]. Creating a ground truth for measuring distances is a tedious task due to the

[1]Examples of retrieval are shown in the supplementary material.

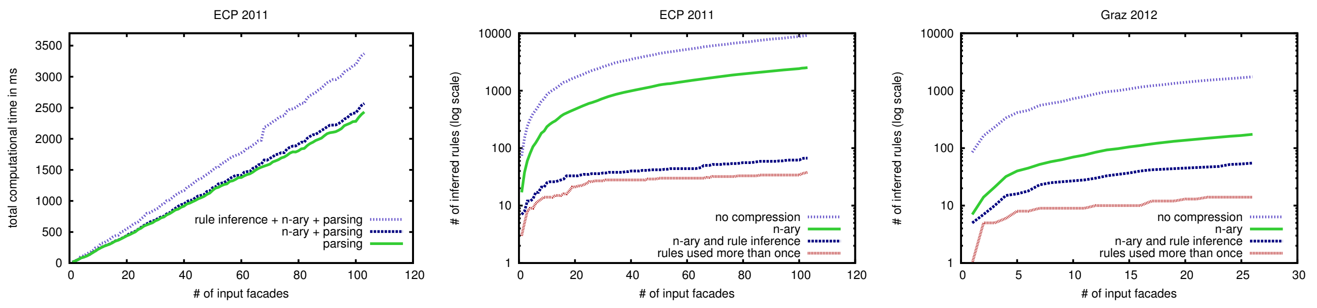

Figure 5. Computational time with respect to the number of Haussmannian façades given as input (left); number of rules with respect to the number of Haussmannian (centre) and Gruenderzeit (right) façades given as input, using no compression, n-ary compression and n-ary and rule inference compression (log scale).

number of pairs to evaluate (104^2 for the ECP2011 dataset). We defined a gold standard distance function as the number of architectural changes in the number of floors, window columns, door placement and location of running balconies. We manually annotated each façade with its related data[2].

For retrieval, we evaluate the distance measures δ_{common} and $\delta_{powersets}$ and count if the ground truth façades with no architectural changes (deemed identical) are retrieved in the top-K ranking. This measure is typically known in identification retrieval as the Cumulative Match Characteristics (CMC) [16] and shows how many retrieved results to look at before finding the desired result.

Optimizing the parameters in Eq. 3 in order to maximize this CMC retrieval score at rank $k = 20$ for the ECP2011 dataset gives $\alpha = 0.5$, $b_p = 0.0625$ and $b_h = 0.125$.

In Fig. 4, we compare the two methods detailed in Sect. 4.3. Using the powerset method, we see that retrieving the exact instance within $k = 1$ has a mean expectation accuracy of 87% whereas within $k = 2$ all the correct façades are retrieved. The common rule approach does not yield such good results in comparison (56% at $k = 2$). This strongly supports our claim that different representations are suitable for different applications.

For clustering, we use the distance measure $\delta_{powersets}$ and can show distinct groups between and within each of the façade datasets for Haussmannian and Gruenderzeit styles, as indicated by the dendrogram and the linked heat maps as shown in Fig. 3. We can see that the distance measurement effectively accounts for structural changes. The distinction between the two styles is clear due to the differences in the frequency of asset types. For instance, shops and balconies are more common in Haussmannian. Consequently, the two logics can be automatically separated and hence two style grammars can be inferred.

5.5. Virtual façade synthesis

For virtual façade synthesis, the quality of the sampling improves when the number of parameter vector instances

[2]The ground truth annotation is available on the author's website.

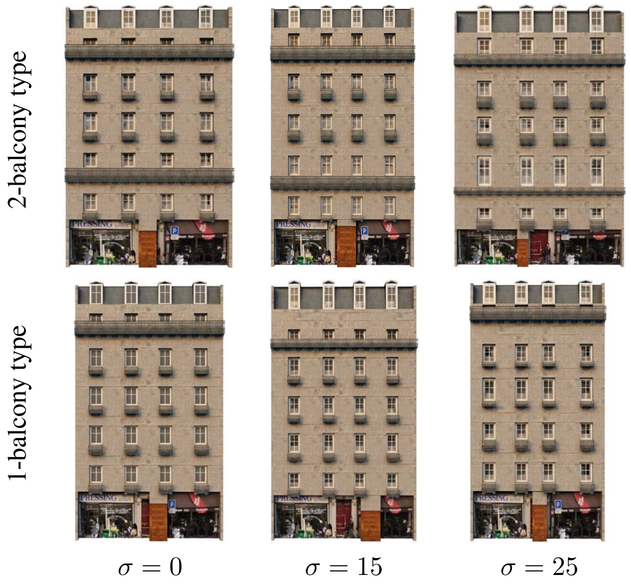

Figure 6. Rows correspond to different façade structure type (i.e. different starting rules), columns correspond to different σ values which influence parameter variations.

associated with each rule increases. Hence, the optimization of the grammatical inference for virtual façade synthesis follows the same objective function as for compression (see Eq. 12). Examples of virtual façades are shown in Fig. 6. Notice the greater variance of the door position, heights of the balconies and roof, in contrast to the variance of the width of windows. Here textures and colours were not randomized to emphasize the structural changes only. Stiny *et al.* [23], Mitchell [15], and Wonka [27] have developed an ontology for architecture. We were able to judge how well shape grammars are suited for *capturing* a set of real examples from a style by evaluating compression, comparison and synthesis. The generated rule set for Haussmannian summarizes the main features of the style. **The most frequent rules correspond to:** 7 floors (including the ground and roof floor), 4 window columns, running balconies on the 2nd and 5th floors and shops on the ground floor.

6. Conclusion

In this work we show that procedural models provide a much larger flexibility than pure mesh-based or semantic labelled representations by enabling compression, façade comparison and new virtual façade synthesis. Our method starts by a binary split procedure on labelled image to create parse trees and consequent procedural rule sets. The final grammar models are optimized on the requirements for compression and virtual synthesis (minimum number of rules inspired by MDL) and retrieval (best ranking performance inspired by bag-of-words models).

Our evaluations confirm that a single grammar model is not enough. The optimization results for compression and retrieval produce different models with different performances. In case of retrieval the performance nearly doubles with a more tailored grammar model. In all, our method removes the need for manual expert work and cuts time to build a procedural façade model from days to milliseconds.

The benefits of our procedural knowledge can be used to highlight the atypical parts in a façade and automatically complete occluded areas. Also the generated grammar rules could be translated to human language to teach architectural principles to humans.

Future work entails lifting the process to 3D and include depth as well as entire buildings into the grammar models. We will also investigate improving noisy semantic image labelling methods with the inferred grammar rules and build joint labelling and grammar inference methods. Finally, the answer to the title is yes.

Acknowledgments. This work was supported by the European Research Counsil (ERC) under the projects VarCity (#273940) and 3D-COFORM (#231809).

References

[1] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys. Towards urban 3d reconstruction from video. In *3DPVT*, 2006. 2

[2] G. Aschwanden, S. Haegler, J. Halatsch, R. Jeker, G. Schmitt, and L. Van Gool. Evaluation of 3D city models using automatic placed urban agents. In *CONVR*, 2009. 1

[3] A. Berg, F. Grabler, and J. Malik. Parsing images of architectural scenes. In *ICCV*, 2007. 2

[4] M. Bokeloh, M. Wand, and H. Seidel. A connection between partial symmetry and inverse procedural modeling. In *SIGGRAPH*, 2010. 2

[5] G. Busatto, M. Lohrey, and S. Maneth. Grammar-based tree compression. Technical report, EPFL, 2004. 5

[6] M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat. The smallest grammar problem. *TIT*, 2005. 2

[7] R. Cilibrasi and P. Vitányi. Clustering by compression. *TIT*, 2005. 5

[8] D. Dai, M. Prasad, G. Schmitt, and L. Van Gool. Learning domain knowledge for facade labeling. In *ECCV*, 2012. 1, 2

[9] C. De La Higuera. A bibliographical study of grammatical inference. *PR*, 2005. 2

[10] S. Haegler, P. Wonka, S. M. Arisona, L. Van Gool, and P. Müller. Grammar-based encoding of facades. In *EGSR*, 2010. 1

[11] J. Halatsch, A. Kunze, and G. Schmitt. Using shape grammars for master planning. *Design Computing and Cognition*, 2008. 1

[12] S. Havemann. *Generative Mesh Modeling*. PhD Thesis, TU Braunschweig, 2005. 3

[13] A. Martinović, M. Mathias, J. Weissenberg, and L. Van Gool. A three-layered approach to facade parsing. In *ECCV*, 2012. 1, 2

[14] M. Mathias, A. Martinović, J. Weissenberg, S. Haegler, and L. Van Gool. Automatic architectural style recognition. In *3D-ARCH*, 2011. 1

[15] W. J. Mitchell. *The Logic of Architecture: Design, Computation, and Cognition*. MIT Press, 1990. 1, 7

[16] H. Moon and P. Phillips. Computational and performance aspects of PCA-based face-recognition algorithms. *Perception*, 2001. 7

[17] P. Müller. *Procedural modeling of buildings*. PhD Thesis, ETH Zurich, 2010. 1

[18] P. Müller, G. Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of facades. In *SIGGRAPH*, 2007. 2, 3

[19] C. Nevill-Manning and I. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *JAIR*, 1997. 2, 5

[20] H. Riemenschneider, U. Krispel, W. Thaller, M. Donoser, S. Havemann, D. Fellner, and H. Bischof. Irregular lattices for complex shape grammar facade parsing. In *CVPR*, 2012. 2, 6

[21] N. Ripperda and C. Brenner. Reconstruction of façade structures using a formal grammar and RjMCMC. *DAGM*, 2006. 2

[22] O. Št'ava, B. Beneš, R. Měch, D. Aliaga, and P. Krištof. Inverse Procedural Modeling by Automatic Generation of L-systems. In *EGSR*, 2010. 2

[23] G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. *IFIP*, 1972. 2, 7

[24] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *CVPR*, 2011. 1, 2, 6

[25] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape prior. In *CVPR*, 2010. 1, 2, 3

[26] B. Watson, P. Müller, P. Wonka, C. Sexton, O. Veryovka, and A. Fuller. Procedural urban modeling in practice. *CGA*, 2008. 1

[27] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. Instant architecture. *ACM Graphics*, 2003. 2, 7

[28] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. In *SIGGRAPH Asia*, 2009. 1, 2