Bayesian Grammar Learning for Inverse Procedural Modeling

Anđelo Martinović and Luc Van Gool

Abstract

Within the fields of urban reconstruction and city modeling, shape grammars have emerged as a powerful tool for both synthesizing novel designs and reconstructing buildings. Traditionally, a human expert was required to write grammars for specific building styles, which limited the scope of method applicability. We present an approach to automatically learn two-dimensional attributed stochastic context-free grammars (2D-ASCFGs) from a set of labeled building facades. To this end, we use Bayesian Model Merging, a technique originally developed in the field of natural language processing, which we extend to the domain of two-dimensional languages. Given a set of labeled positive examples, we induce a grammar which can be sampled to create novel instances of the same building style. In addition, we demonstrate that our learned grammar can be used for parsing existing facade imagery. Experiments conducted on the dataset of Haussmannian buildings in Paris show that our parsing with learned grammars not only outperforms bottom-up classifiers but is also on par with approaches that use a manually designed style grammar.

1. Introduction

Over the last few years, there has been a flurry of approaches tackling the problem of urban modeling. Digital mapping of existing cities is reaching new heights as users can now browse detailed 3D models of cities instead of flat maps. In the entertainment industry, particularly for movies and games, there is an ever rising need for detailed and realistic models of virtual cities. Manual modeling of individual buildings usually provides good results, but the process is very time consuming and expensive.

Procedural modeling is an efficient way to create 3D models in a fast and scalable way. There, the structure of the object is encoded as a set of parametric rules. Models are generated by iteratively applying the rules on a starting shape. This approach was successfully applied on various categories of objects, such as plants, landscapes and architecture [25]. In urban procedural modeling, the knowledge of the building style and layout is most commonly encoded as a *shape grammar* [17]. A particular type of shape grammars for architectural modeling, the *split grammar,* was introduced by [27], and further refined in [11]. A specific building can then be represented as a particular derivation, or a parse tree of that grammar.

Figure 1: "Somewhere in Paris": a street with buildings sampled from our induced grammar.

Some approaches have used shape grammars as higher-order knowledge models for reconstruction of buildings. *Inverse procedural modeling (IPM)* is an umbrella term for approaches that attempt to discover the parametrized rules and the parameter values of the procedural model. Various methods have specialized this general IPM framework by assuming that the rules are known in advance, while the parameters are allowed to vary. This top-down model is then fitted to bottom-up cues derived from the data. Vanegas *et al*. [24] used a simple grammar for buildings that follow the Manhattan world assumption. A grammar was fitted from laser-scan data in [23]. Mathias *et al*. [10] reconstructed Greek Doric temples using template procedural models. An approach using reversible jump Markov Chain Monte Carlo (rjMCMC) for fitting split grammars to data was described in [16]. Teboul *et al*. [21] presented an efficient parsing scheme for Haussmannian shape grammars using Reinforcement Learning.

However, all of the methods mentioned above share a common drawback. They assume that a manually designed grammar is available from the outset. This is a serious con-

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.33

straint, as it limits the reconstruction techniques to a handful of building styles for which pre-written grammars exist. Creating style-specific grammars is a tedious and time-consuming process, which is usually performed only by a few experts in the field. So, a natural question arises: can we learn procedural grammars from data?

So far, the research in the field of general IPM has been limited to a small number of approaches. Learning L-systems from synthetic 2D vector data was tackled in [26]. Applications of general IPM in urban modelling started with Aliaga *et al*. [1], who presented an interactive method for extracting facade patterns from images. Bokeloh *et al*. [2] learned deterministic shape grammar rules from triangle meshes and point clouds. Attribute graph grammars [5] were presented as a method of top-down/bottom-up image parsing, though restricting the detected objects in scenes to rectangles.

In the field of formal grammar learning, a famous conclusion of Gold [3] states that no superfinite family of deterministic languages (including regular and context-free languages) can be identified in the limit. However, Horning [6] showed that the picture is not so grim for statistical grammar learning, and demonstrated that stochastic context-free grammars (SCFGs) can be learned from positive examples. Currently, one of the popular methods for learning SCFGs from data is Bayesian Model Merging [18], which makes the grammar induction problem tractable by introducing a Minimum Description Length (MDL) prior on the grammar structure. This approach was recently applied for learning probabilistic programs [7] and design patterns [20].

2. Our Approach

Inspired by recent successes of Bayesian Model Merging outside computer vision, we propose a novel approach of inducing procedural models, particularly *split grammars,* from a set of labeled images. We focus our discussion on facade modeling, since facades are mostly two-dimensional, and exhibit a logical hierarchy of elements.

The overview of our approach can be seen in Fig. 2. The input to our system is a set of facade images, which are semantically segmented into classes such as walls, windows, etc. In the first step we create a stochastic grammar which generates only the input examples with equal probabilities. However, we want to find a grammar that can also generalize to create novel designs. We formulate this problem as a search in the space of grammars, where the quality of a grammar is defined by its posterior probability given the data. As described in Sec. 5, this requires an optimal trade-off between the grammar description length (smaller grammars are preferred) and the likelihood of the input data. The latter is obtained by parsing the input examples with the candidate grammar.

Previous work has shown that image parsing with a known set of grammar rules is a difficult problem by itself [15, 22]. On the other hand, our grammar search procedure typically needs to evaluate a huge number of candidate grammars. This means that we have to parse the input examples in a very short time, lest the grammar search last indefinitely. Different authors have tackled this *curse of dimensionality* during parsing in different ways: assuming that the building exhibits a highly regular structure [12], using approximate inference such as MCMC [16], or exploiting grammar factorization [22]. Recent work by Riemenschneider *et al*. [15] has shown that it is possible to perform exact image parsing using dynamic programming if the image is reduced to an irregular lattice. This approach reduces the effective dimensionality of the problem, while not sacrificing much of the quality.

Following their example, we transform all of our input images into irregular lattices, casting our grammar search procedure into a lower dimensional space. In this space we use our own, modified version of the Earley-Stolcke parser [18], a technique from natural language processing adapted to parse 2D lattices instead of 1D strings. This dimensionality reduction enables the grammar search procedure to run within a reasonable time. Finally, in order to perform image parsing, the induced grammar is cast into the original space. This stochastic, parameterized grammar can either be used as a graphics tool for sampling building designs, or as a vision tool to alleviate image parsing of actual buildings.

Our contributions are: (1) A novel approach for inducing procedural split grammars from data. To the best of our knowledge, we are the first to present a principled approach for learning probabilistic two-dimensional split grammars from labeled images. (2) A generalization of the Earley-Stolcke SCFG parser to two dimensional lattices. (3) An adapted rjMCMC parser in the style of [19] for image-scale parsing. (4) An experimental evaluation suggesting that learned grammars can be as effective as human-written grammars for the task of facade parsing.

3. 2D-ASCFGs

We define a two-dimensional attributed stochastic context-free grammar (2D-ASCFG) as a tuple $G = (N, T, S, R, P, A)$, where N is a set of non-terminal symbols, T a set of terminal symbols, S the starting non-terminal symbol or axiom, R a set of production rules, $\{P(r), r \in R\}$ a set of rule probabilities and $\{A(r), r \in R\}$ a set of rule attributes.

Every symbol is associated with the corresponding shape, representing a rectangular region. Starting from the axiom, production rules subdivide the starting shape either in horizontal or vertical directions. We define the set R as a union of horizontal and vertical productions: $R = R_h \cup R_v$.

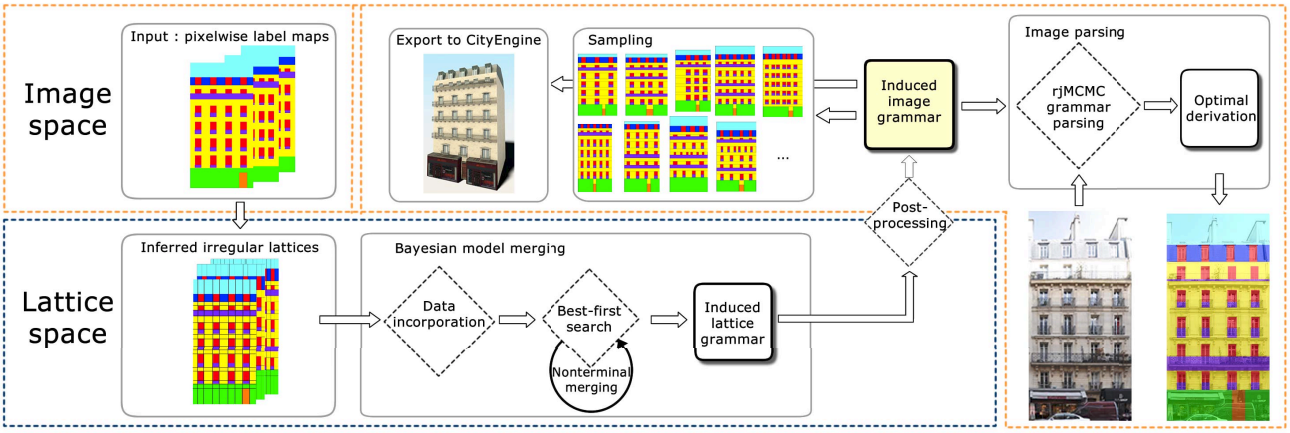

Figure 2: The overview of our approach.

These productions correspond to standard horizontal and vertical split operators in split grammars. A production is of the form $X \rightarrow \lambda$, where $X \in N$ is called the left-hand-side (LHS), and $\lambda \in (N \cup T)^+$ is called the right-hand-side (RHS) of the production.

For every production we define $P(X \rightarrow \lambda)$ as the probability that the rule is selected in the top-down derivation from the grammar. For the grammar to be well-formed, the productions with X as LHS must satisfy the condition $\sum_\lambda P(X \rightarrow \lambda) = 1$. We additionally associate each grammar rule r with a set of attributes $A(r) = \{\alpha_i\}$. The elements of a single attribute are the relative sizes of the RHS shapes in respect to their parent shape, in the splitting direction: $\alpha_i = \{s_1, ..., s_{|\lambda|}\}, \sum_i s_i = 1$. These relative sizes sum up to one because RHS shapes always fill the entire shape of their parent.

We denote by τ a parse tree from the grammar, rooted on the axiom, its interior nodes corresponding to non-terminal symbols, and its exterior nodes to terminal symbols. The parse tree is obtained by applying a sequence of rules on the axiom and non-terminal nodes. A derivation from the grammar consists of the parse tree and the selected attributes at each node: $\delta = (\tau, \alpha)$. The probability of a single derivation is the product of all rule probabilities selected at each node s of the parse tree: $P(\delta) = \prod_{s \in \delta} P(r_s)$. The set of terminal nodes of a parse tree defines a lattice over an area. A lattice is a rectangular tesselation of 2D space, exactly filling the shape of the axiom. We define the likelihood of the grammar G generating a lattice l as $L(l|G) = \sum_{\delta \Rightarrow l} P(\delta)$, where we sum over the probabilities of all derivations that yield a particular lattice.

4. Bayesian Model Merging

To cast our grammar learning as an instance of Bayesian Model Merging, we need to define several methods:

• **Data incorporation**: given a body of data, build an initial grammar which generates only the input examples.

• **Model merging**: propose a candidate grammar by altering the structure of the currently best grammar.

• **Model evaluation**: evaluate the fitness of the candidate grammar compared to the currently best grammar.

• **Search**: use model merging to explore the grammar space, searching for the optimal grammar

4.1. Data Incorporation

We start with a set of n_f facade images, with each pixel labeled as one of the n_l terminal classes (window, wall, balcony, etc.) As already mentioned in Sec. 1, grammar induction would be infeasible in the image space due to the curse of dimensionality. To mitigate this issue, all input images are converted into lattices following an approach similar to [15]. Every rectangular region in the resulting two-dimensional tesselation of the image is labeled with the majority vote from the corresponding pixel labels.

For each lattice in the input set, we create an instance-specific split grammar, with terminal symbols corresponding to image labels. Non-terminal productions are created by alternatively splitting the image in horizontal and vertical directions, starting with the latter. All production probabilities are set to 1; all attributes are initialized to the relative sizes of right-hand side elements. For example, the first production splits the axiom into horizontal regions represented by newly instantiated non-terminals and parametrized by their height: $S \rightarrow X_i \dots X_n, p = 1, A = \{\{h(X_i), \dots, h(X_n)\}\}$, where the rule probability p is initialized to 1, but is allowed to change in the model search. The procedure is stopped at the level of a single lattice element, where we instantiate lexical productions, i.e. productions with a single terminal on the RHS: $X \rightarrow label, p = 1, A = \{\{1\}\}$. Lexical productions re-

main deterministic, as they only label the entire shape of the parent with the given terminal class.

Now we have a set of deterministic grammars G_i, each producing exactly one input lattice. The next step is to merge them into a single grammar by setting all of their axioms to the same symbol and aggregating all symbols and productions: $G_0 = (\cup N_i, \cup T_i, S, \cup R_i, \cup P_i, \cup A_i)$. The probabilities of the rules starting from the axiom are changed to $1/n_f$, which means that the grammar G_0 generates each of the input examples with the same probability.

4.2. Merging

A new grammar is proposed by selecting two non-terminals X_1 and X_2 from the current grammar and replacing them with a new non-terminal Y. This operation has two effects on the grammar. First, all the RHS occurrences of X_1 and X_2 are replaced by Y:

$$\begin{array}{lll} Z_1 \to \mu_1 X_1 \lambda_1 & \underset{\dashrightarrow}{merge} & Z_1 \to \mu_1 Y \lambda_1 \\ Z_2 \to \mu_2 X_2 \lambda_2 & & Z_2 \to \mu_2 Y \lambda_2 \end{array}$$

where $\mu, \lambda \in (N \cup T)^+$. If $Z_1 = Z_2, \mu_1 = \mu_2, \lambda_1 = \lambda_2$, then the two resulting productions are merged in one. In that case, the attribute set of the new production is defined as the union of the attributes of the old productions.

Second, all the productions where X_1 and X_2 appear on the LHS are replaced with Y, as well:

$$\begin{array}{lll} X_1 \to \lambda_1 & \underset{\dashrightarrow}{merge} & Y \to \lambda_1 \\ X_2 \to \lambda_2 & & Y \to \lambda_2 \end{array}$$

Again, if $\lambda_1 = \lambda_2$, only one production is created. If we create a production $Y \to Y$, we delete it from the grammar.

The merging operation basically states that in the resulting grammar two previously different symbols may be used interchangeably, although with different probabilities. The only restriction that we place on the merging operations is that X_1 and X_2 have to be "label-compatible", meaning that the sets of terminal symbols reachable from both nodes have to be equal. In this way we prevent nonsensical merges, e.g. merging two non-terminals representing sky and door regions, respectively. We also improve the speed of the inference procedure by restricting the search space.

4.3. Evaluating Candidate Grammars

Our goal is to find the grammar model G that yields the best trade-off between the fit to the input data D and a general preference for simpler models. From a Bayesian perspective, we want to maximize the posterior $P(G|D)$, which is proportional to the product of the grammar prior $P(G)$ and a likelihood term $P(D|G)$. We can decompose the grammar model into a structure part G_S (representing grammar symbols and rules) and the parameter part θ_g(rule probabilities): $G = (G_S, \theta_g)$.

The model prior $P(G)$ then factorizes to $P(G_s)P(\theta_g|G_s)$, the product of priors over structure and parameters. To define the prior over the grammar structure we follow a Minimum Description Length (MDL) principle. The grammar's description length $DL(G_s)$ is calculated by a simple encoding of productions, where every occurrence of a non-terminal in a production contributes with $log\,|N|$ bits, $|N|$ being the total number of non-terminals in the grammar. Then, the structure prior is defined as $P(G_s) = e^{-DL(G_s)}$. We use symmetrical Dirichlet parameter priors, as all productions with the same LHS form a multinomial distribution.

In [18] it was shown that in order to calculate the posterior over the model structure $P(G_s|D) \propto P(G_s)P(D|G_s)$, one needs to integrate over the parameter prior:

$$P(D|G_s) = \int_{\theta_g} P(\theta_g|G_s)P(D|G_s, \theta_g)d\theta_g \qquad (1)$$

Fortunately, we can approximate this integral with the ML estimate of $P(D|G_s)$ by using the Viterbi assumption. This basically means that we assume that every input sample is generated by a single derivation tree of the grammar. The likelihood of a single input example is then the product of all rule probabilities used in the Viterbi derivation. Since Viterbi derivations and rule probabilities θ_g depend on each other, we use the Expectation-Maximization procedure to find the optimal values for θ_g. In the E-step, starting from an estimate for θ_g, the expected usage counts $\hat{c}(X \to \lambda)$ for each rule are calculated. This is done by finding Viterbi derivations for all input data and counting the number of times every rule was used. In the M-step, the rule probabilities $\hat{\theta}_g$ are re-estimated using the formula:

$$\hat{P}(X \to \lambda) = \frac{\hat{c}(X \to \lambda)}{\sum_\mu \hat{c}(X \to \mu)} \qquad (2)$$

where μ iterates over all possible LHS choices for X. The process is iterated until convergence.

4.4. 2D Earley Parsing

In order to find the Viterbi derivations of each input lattice in the E-step, we use a modified version of the Earley-Stolcke parser [18], which we extended from parsing strings to parsing 2D lattices. To the best of our knowledge, we are the first to create an Earley parser for two dimensional SCFGs. We provide its implementation details in a technical report [9].

Using Earley's parser instead of more common CKY parsing [28] has a number of advantages. Its worst-case complexity is cubic in the size of the input, but it can perform substantially better for many well-known grammar

classes. Another appealing property is that it places no restrictions on the form of the grammar. This sets us apart from previous work which either requires the grammar to be in a Chomsky Normal Form [21], or that the rules have to satisfy optimal substructure property [15].

4.5. Search in Model Space

In order to define a flexible search procedure, we modify the posterior calculation with a global prior weight w, which gives us control over the balance between the likelihood and the prior. Utilizing the Boltzmann's transformation, we transform the posterior maximization into an energy minimization:

$$E(G|D) = -w\, log\, P(G) - log\, P(D|G) \qquad (3)$$

By setting w to a low value, we decrease the influence of the prior, thereby making the search procedure stop earlier. For larger values of w, we increase the tendency to generalize beyond the data. The influence of global prior weight w on induced grammar size is shown in Table 1.

Starting from the initial grammar, we follow a greedy best-first approach: in each iteration, every pair of non-terminals is considered for merging, and all of the candidate grammars are evaluated. The candidate with the minimum energy is accepted if it has lower energy than the current grammar. The rule probabilities are learned in each step using the EM procedure presented in 4.3.

The described method produced satisfactory results in our experiments. Of course, one may imagine more intricate ways of searching through the grammar space, e.g. by using a beam search or a random walk algorithm. We leave this for future work.

4.6. Final Model

The grammar resulting from the search procedure is still limited to the lattice space. To cast the grammar back in the image space, we perform two post-processing steps.

First, we collapse sequences of the same non-terminal symbol in a production to a single symbol with correspondingly modified attributes, for example:

$$X \to \lambda Y Y \mu \quad \xrightarrow{Collapse} \quad X \to \lambda Y \mu$$
$$A = \{\{s_1, y_1, y_2, s_2\}\} \qquad A = \{\{s_1, y_1{+}y_2, s_2\}\}$$

Second, for every production $p = (X \to \lambda_1 \dots \lambda_k)$, we fit a $(k-1)$-variate Gaussian distribution $\phi(A) = \mathcal{N}(\bar{\mu}, \hat{\Sigma})$ to the set of its attributes $A(p) = \{\alpha_1 \dots \alpha_n\}$:

$$\bar{\mu} = \frac{1}{n}\sum_{j=1}^{n}\alpha_j \qquad (4)$$

$$\hat{\Sigma} = \frac{1}{n-1}\sum_{j=1}^{n}(\alpha_j - \bar{\mu})(\alpha_j - \bar{\mu})^T \qquad (5)$$

	Initial grammar G_0	Induced, $w = 0.3$	Induced, $w = 1.0$
$\|N\|$	126.8 ± 6.61	26.6 ± 0.89	14 ± 0.0
$\|R_h\|$	121.8 ± 6.61	65 ± 6.70	27.8 ± 2.68
$\|R_v\|$	33 ± 0.0	15.6 ± 2.60	11 ± 1.41

Table 1: Size comparison: initial grammar created by grammar incorporation, and two inferred grammars with prior weights of $w = 0.3$ and $w = 1.0$.

This enables us to sample productions with continuous attributes, by sampling directly from the estimated size distribution. Note that every production with the RHS size of k has $k-1$ degrees of freedom. If $k = 1$, we are dealing with a lexical production, for which no distribution is estimated since they have the relative size of 1 by definition.

5. Parsing in Image Space

The grammar induced in the previous section is now amenable for image-scale parsing. However, two main problems arise when trying to design an efficient optimization method. First, we cannot use exact methods such as dynamic programming as we allow our attributes to take on continuous values. Second, due to the stochastic nature of the grammar, the number of attributes can change. In order to tackle the first issue, we use a Markov Chain Monte Carlo approach, which reduces the optimization to sampling. However, as the MCMC operates over a fixed-dimensional space, we must consider its extension in the form of Reversible jump MCMC (rjMCMC). Talton [19] presented a rjMCMC-based method to parse parametric, stochastic, context-free grammars, given a high-level specification of the desired model. However, their method requires that only terminal symbols of the grammar contain descriptive continuous parameters. In contrast, we present a modified version of [19] that lifts this constraint. We also use a different likelihood computation, utilizing a pixel-based classifier to calculate the terminal merit.

5.1. Grammar Parsing via rjMCMC

For a given test image, our task is to find the derivation from the grammar that has the best fit to the image. Similarly to Sec. 4.3, we define a posterior of the derivation δ given the image:

$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)} \qquad (6)$$

where ϕ is defined in Sec. 4.6. Note that we have factorized the prior into a rule and attribute term over all non-terminal

nodes s of the derivation tree. We can ignore the normalizing constant for the purposes of maximization and define the energy through Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute} \quad (7)$$

The energy that we want to minimize is composed of three terms. The *rule term* is calculated by summing up the negative log probabilities of all rules r_s selected in the derivation. The *attribute term* measures the discrepancy between the proposed attributes and the expected values of attribute distributions estimated in Sec. 4.6. To calculate the *image term*, we use the Random Forest pixel classifier of [22], which outputs the label probability distribution P_{RF} for each pixel in the image.

$$E^{image} = \sum_{t\in\delta} \sum_{x_i\in t} -log\, P_{RF}(l_t|x_i) \quad (8)$$

The sum is defined over all terminals t in the derivation tree. Integral images are used to rapidly calculate the inner summation of pixel energies over the rectangular region of each terminal symbol. By making this choice of image support, we can make a direct comparison to the approach of [21].

5.1.1 Search

We utilize the standard rjMCMC formulation with Metropolis-Hastings (MH) update from [4]. The chain is initialized with a random derivation $\delta = (\tau, \alpha)$ from the grammar. We define α as a concatenation of all selected attribute elements (i.e. relative RHS sizes) in a pre-order traversal of tree τ. To ease the discussion, we will refer to α as the *parameter vector*.

In every MH iteration, the chain is evolved by performing either a dimension-preserving "diffusion" move, or a dimension-altering "jump" move [19]. In the diffusion move, a random node is selected in the derivation tree, and its corresponding parameters are resampled from a Gaussian proposal distribution, centered on the current parameters. Since the proposal function is symmetric, the acceptance probability for the move reduces to:

$$\rho_{\delta\to\delta'} = min\{1, \frac{p(\delta'|I)}{p(\delta|I)}\} = min\{1, e^{-(E_{\delta'}-E_\delta)}\} \quad (9)$$

In the jump move, again a random node h is selected in the derivation tree, and a new rule is sampled from all rules applicable to the current LHS. If the RHS size of the new production is different from the old one, we have to re-derive the entire tree under h. This changes not only the topology of the derivation tree τ, but also the parameter vector from some n-dimensional α to m-dimensional α'. In order for the jump move to be reversible, we need to define a dimension matching function, which casts both chain states into a common space. This can be done by supplementing the α and α' with additional parameter vectors u and u', such that $n + |u| = m + |u'|$.

We shall now define this mapping. Let k be the index of the first parameter of node h in the concatenated vector α, d_1 the number of parameters in the subtree underneath the node h, and d_2 the number of parameters in the subtree after resampling the rule at h. Let us also define u and u' as vectors of d_1 and d_2 uniformly sampled numbers in the interval $[0, 1]$, respectively. We can now write the mapping as follows:

$$\alpha'_i = \begin{cases} \alpha_i & , i \in [1, l] \\ u_{i-l} & , i \in [l+1, l+d_2] \\ \alpha_{i-d_2+d_1} & , i \in [l+d_2+1, m] \end{cases}$$
$$u'_i = \alpha_{i+l} \quad , i \in [1, d_2] \quad (10)$$

The reverse mapping is obtained from Eq. 10 by swapping (α, u, m, d_2) with (α', u', n, d_1). This choice of dimension function allows us to express the acceptance probability of the jump move in a simple way[1]:

$$\rho_{\delta\to\delta'} = min\{1, \frac{q_{\tau'}(h)}{q_\tau(h)} e^{-[(E^{img}_{\delta'}+E^{attr}_{\delta'})-(E^{img}_{\delta}+E^{attr}_{\delta})]}\} \quad (11)$$

where $q_\tau(h)$ is the probability of choosing a non-terminal h in a tree τ.

The chain is guaranteed to converge to the true posterior as the number of iterations goes to infinity. In practice, the random walk is stopped after a certain number of iterations. Similar to [19], we use parallel tempering to improve the speed of convergence. Eight chains are run in parallel, with temperature quotient between chains set to 1.3. For jump moves, we employ the technique of *delayed rejection*: a diffusion move is attempted immediately after a jump move, and two moves are accepted or rejected in unison.

6. Results

In all grammar learning experiments, the training set was limited to 30 images to keep the induction time within reasonable bounds. In image parsing experiments, w is set to 0.3, and rjMCMC search is run for 100k iterations. The process is repeated 5 times, and the minimum energy chain state is selected as the result.

[1] The proof of Eq. 11 is given in the supplementary material.

6.1. Parsing Existing Facades

To show that our grammar learning is usable on real-world examples, we use the well-established Ecole Centrale Paris (ECP) facade parsing dataset [13], which contains 104 images of Haussmannian-style buildings. We use the 5-fold cross-validation experimental setup from [8].

In Table 2 we compare the accuracies achieved by four different semantic facade segmentation methods. Each method was evaluated on the ground truth annotations from [8]. We evaluate the accuracy in terms of class-wise and total pixel averages. As a baseline, we use the MAP estimation of the Random Forest classifier, provided by [22]. Our approach clearly outperforms the baseline in the total pixel accuracy and all but one class. Since the RF classifier output is used in both our method and the RL-based approach of [21], our methods are directly comparable. The results that we obtain show that learned grammars can be just as effective in facade parsing as their manually written counterparts, even outperforming them in some cases.

To put the results in context, we also show the performance of the state of the art (SOA) method in facade parsing [8]. However, as the SOA method uses segment classification and object detectors, it is not strictly comparable, since we use pixel classification cues. A promising direction for future work would be to learn grammars from the output of methods such as [8], eliminating the need for labeled ground truth images.

6.2. Generating Novel Designs

The advantage of having a grammar for a certain style of buildings is that we can easily sample new designs from it. In this scenario, we generate a random derivation from the grammar by starting from the grammar axiom as the first node of the tree. At each node, we sample a rule based on its probability in the grammar. The relative sizes of the RHS are sampled from the estimated Gaussian distribution ϕ. Finally, the terminal symbols are replaced with instances of architectural elements from a 3D shape and texture library. We rendered a whole street of buildings sampled from our induced grammar in CityEngine [14]. The results are shown in Fig. 4, where we also demonstrate the effect of the prior weight parameter w on the generalization capabilities of the grammar. In Fig. 4b, we had intentionally set the prior weight too high, hence all compatible non-terminal symbols were merged, leading to an excessively general grammar. With the proper choice of w, we can find a good trade-off between the data fit and generalization, as shown in Fig. 4a.

Class	**RF**[22]	**RL**[21]	**Ours**	**SOA**[8]
Window	29	62	66	75
Wall	58	82	80	88
Balcony	35	58	49	70
Door	79	47	50	67
Roof	51	66	71	74
Sky	73	95	91	97
Shop	20	88	81	93
Overall	48.55	74.71	74.82	84.17

Table 2: Per-class and overall pixel accuracy (in percent) on the ECP dataset: RF - Random Forest. RL - Manually designed grammar. SOA - State of the art.

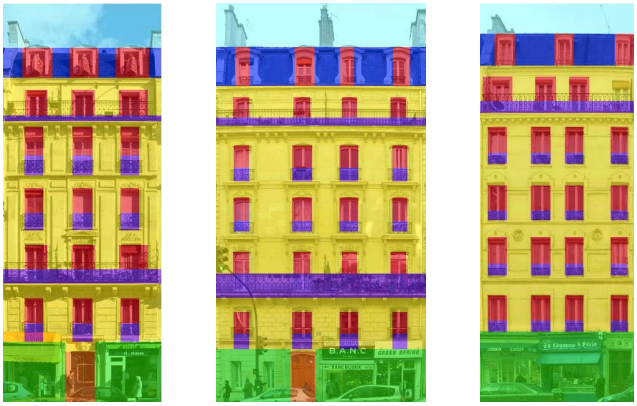

Figure 3: Example images from the ECP dataset parsed with our induced grammar. Note that the output is not restricted to a grid as in [21].

7. Conclusion and Future Work

In this work we introduced a principled way of learning procedural split grammars from labeled data. The validity of our approach is demonstrated using an example of urban modeling. Our induced procedural grammar not only generates new buildings of the same style, but also achieves exceptional results in facade parsing, outperforming similar approaches which require a manually designed set of grammar rules.

In future work, other strategies for the design of grammar merging operators will be explored, undoubtedly requiring elaborate search strategies. Furthermore, more complex shape grammars could be inferred by extending the Earley parser, which is currently limited to grid-like designs. We will also investigate the feasibility of designing an iterative approach for image parsing. In each step of this approach, a more refined grammar is inferred through initial labeling, augmenting in turn the labeling in subsequent iterations.

Acknowledgement. This work was supported by ERC Advanced Grant VarCity and Research Programme of the Fund for Scientific Research - Flanders (Belgium) (FWO - G.0004.08).

(a) w=0.3 (b) w=1.0

Figure 4: Generating a scene with different grammars. (a) Samples from the Bayes-optimal grammar. (b) The grammar is over-generalizing due to high prior weight.

References

[1] D. G. Aliaga, P. A. Rosen, and D. R. Bekins. Style grammars for interactive visualization of architecture. *TVCG*, 13(4), 2007. 2

[2] M. Bokeloh, M. Wand, and H.-P. Seidel. A connection between partial symmetry and inverse procedural modeling. *SIGGRAPH*, 29(4), 2010. 2

[3] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5), 1967. 2

[4] P. J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4), 1995. 6

[5] F. Han and S.-C. Zhu. Bottom-up/top-down image parsing with attribute grammar. *IEEE TPAMI*, 31(1), 2009. 2

[6] J. J. Horning. *A study of grammatical inference*. PhD thesis, Stanford, CA, USA, 1969. AAI7010465. 2

[7] I. Hwang, A. Stuhlmüller, and N. D. Goodman. Inducing probabilistic programs by bayesian program merging. *CoRR*, arXiv:1110.5667, 2011. 2

[8] A. Martinović, M. Mathias, J. Weissenberg, and L. Van Gool. A three-layered approach to facade parsing. In *ECCV*, 2012. 7

[9] A. Martinović and L. Van Gool. Earley parsing for 2D stochastic context free grammars. Technical Report KUL/ESAT/PSI/1301, KU Leuven, 2013. 4

[10] M. Mathias, A. Martinović, J. Weissenberg, and L. Van Gool. Procedural 3D building reconstruction using shape grammars and detectors. In *3DIMPVT*, 2011. 1

[11] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *SIGGRAPH*, 25(3), 2006. 1

[12] P. Muller, G. Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of facades. *SIGGRAPH*, 26(3), 2007. 2

[13] Olivier Teboul. Ecole Centrale Paris Facades Database. http://www.mas.ecp.fr/vision/Personnel/teboul/data.php, 2010. 7

[14] Procedural. CityEngine. http://www.procedural.com/, 2010. 7

[15] H. Riemenschneider, U. Krispel, W. Thaller, M. Donoser, S. Havemann, D. W. Fellner, and H. Bischof. Irregular lattices for complex shape grammar facade parsing. In *CVPR*, 2012. 2, 3, 5

[16] N. Ripperda and C. Brenner. Reconstruction of façade structures using a formal grammar and rjmcmc. In *DAGM*, 2006. 1, 2

[17] G. Stiny. Pictorial and formal aspects of shape and shape grammars, 1975. Birkhauser Verlag, Basel. 1

[18] A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, University of California at Berkeley, 1994. 2, 4

[19] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. Metropolis procedural modeling. *SIGGRAPH*, 30(2), 2011. 2, 5, 6

[20] J. O. Talton, L. Yang, R. Kumar, M. Lim, N. D. Goodman, and R. Měch. Learning design patterns with bayesian grammar induction. In *UIST*, 2012. 2

[21] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *CVPR*, 2011. 1, 5, 6, 7

[22] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape priors. In *CVPR*, 2010. 2, 6, 7

[23] A. Toshev, P. Mordohai, and B. Taskar. Detecting and parsing architecture at city scale from range data. In *CVPR*, 2010. 1

[24] C. Vanegas, D. Aliaga, and B. Beneš. Building reconstruction using manhattan-world grammars. In *CVPR*, 2010. 1

[25] C. A. Vanegas, D. G. Aliaga, P. Wonka, P. Müller, P. Waddell, and B. Watson. Modelling the appearance and behaviour of urban spaces. *Comput. Graph. Forum*, 29(1), 2010. 1

[26] O. Šťava, B. Beneš, R. Měch, D. G. Aliaga, and P. Krištof. Inverse procedural modeling by automatic generation of l-systems. *Computer Graphics Forum*, 29(2), 2010. 2

[27] P. Wonka, M. Wimmer, F. X. Sillion, and W. Ribarsky. Instant architecture. *SIGGRAPH*, 22(3), 2003. 1

[28] D. H. Younger. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2), 1967. 4