Learning grammars for architecture-specific facade parsing

Raghudeep Gadde, Renaud Marlet, Nikos Paragios

▶ **To cite this version:**

Raghudeep Gadde, Renaud Marlet, Nikos Paragios. Learning grammars for architecture-specific facade parsing. [Research Report] RR-8600, INRIA. 2014. hal-01069379v1

HAL Id: hal-01069379

https://inria.hal.science/hal-01069379v1

Submitted on 29 Sep 2014 (v1), last revised 2 Oct 2015 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

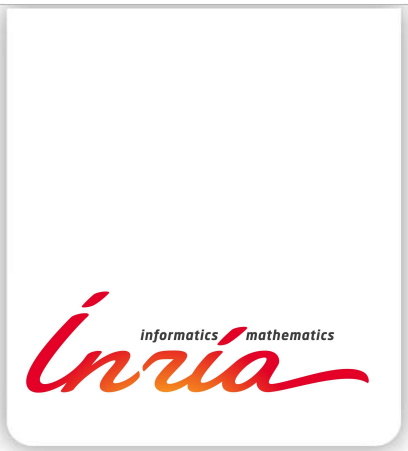
L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Learning grammars for architecture-specific facade parsing

Raghudeep Gadde , Renaud Marlet, Nikos Paragios

RESEARCH REPORT
N° 8600
September 2014
Project-Teams GALEN

ISSN 0249-6399 ISRN INRIA/RR--8600--FR+ENG



Learning grammars for architecture-specific facade parsing

Raghudeep Gadde * †, Renaud Marlet*, Nikos Paragios† ‡

Project-Teams GALEN

Research Report n° 8600 — September 2014 — 31 pages

Abstract: Parsing facade images requires optimal handcrafted grammar for a given class of buildings. Such a handcrafted grammar is often designed manually by experts. In this paper, we present a novel framework to learn a compact grammar from a set of ground-truth images. To this end, parse trees of ground-truth annotated images are obtained running existing inference algorithms with a simple, very general grammar. From these parse trees, repeated subtrees are sought and merged together to share derivations and produce a grammar with fewer rules. Furthermore, unsupervised clustering is performed on these rules, so that, rules corresponding to the same complex pattern are grouped together leading to a rich compact grammar. Experimental validation and comparison with the state-of-the-art grammar-based methods on four different datasets show that the learned grammar helps in much faster convergence while producing equal or more accurate parsing results compared to handcrafted grammars as well as grammars learned by other methods. Besides, we release a new dataset of facade images from Paris following the Art-deco style and demonstrate the general applicability and extreme potential of the proposed framework.

Key-words: grammar learning, facade parsing, subtree isomorphism, clustering

* IMAGINE, Ecole des Ponts Paris-Tech
† Center for Visual Computing, Ecole Centrale Paris
‡ GALEN Group, INRIA-Saclay, France

RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

1 rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing
Campus de l'École Polytechnique
91120 Palaiseau

Grammaires de spcifique l'architecture faade analyse apprentissage

Résumé : Dans cet article, nous prsentons un cadre nouveau apprendre une grammaire compacte d'un ensemble d'images de ground-truth.

Mots-clés : grammar learning, facade parsing, subtree isomorphism, clustering

1 Introduction

How building facades are segmented is of great interest in computer vision due to the number of applications and associated issues. Knowing the regularities in facade layout can be used in video games and movies to generate plausible urban landscapes with realistic rendering [20]. It can also guide the analysis of building images to construct semantized models that can be used for urban planning and in simulation tasks (e.g., for thermal performance evaluation or shadow casting studies) as well as to compact data for virtual navigation in cities.

Existing approaches for facade analysis use either conventional segmentation methods [6, 18] or grammar-driven recognition methods [19, 22, 29]. Conventional segmentation methods treat the problem as a pixel labeling task, with the possible addition of local regularity constraints related to building elements, but ignoring the global structural information in the architecture. On the contrary, methods based on shape grammars impose strong structural consistencies by considering only segments that follow a hierarchical decomposition corresponding to a combination of grammar rules. However, these methods require carefully handcrafted grammars to reach good performance. Besides, as many grammars as different architecture styles are required, and it is not clear who will write and finely tune them, with what expertise and at which cost, when there exists so many building styles.

In this work, we propose a method to automatically learn grammars from annotated images, which we illustrate on facade analysis. The grammars we learn are specific to the architecture style of the training samples. Using these grammars, we reach state-of-the-art parsing results, competing with handcrafted grammars. Thanks to our method, the tedious grammar writing and tuning task is turned into the much simpler and basic task of annotating facade images.

1.1 Related Work

Conventional segmentation techniques rely on grouping together consistent visual characteristics while imposing smoothness. Popular methods are based on active contours [12, 21], mean-shift clustering [5] and graph cuts [2, 13]. However, although they obtain very good pixel-wise scores, these techniques are not appropriate for a number of applications because they frequently produce segments that are inconsistent with basic architectural rules, e.g., irregular window sizes or alignments, or balconies shifted from associated windows. Moreover, as they label only what is visible, they are sensitive to occlusions, e.g., due to potted plants on windows and balconies, or to pervasive foreground objects in the street: trees, vehicles, pedestrians, street signs, lampposts, etc. As a result, important elements can be partially or totally missing from the produced segments, e.g., portions of wall or even complete windows. These methods may also be sensitive to variations of illumination such as cast shadows, night lighting and reflection, although the sensitivity can be partly reduced with larger training sets. Improvements have been obtained using ad hoc attributes that are specific to facade images [6, 18], yielding state-of-the-art pixel classification, but still breaking architectural rules.

A number of structural constraints can be hard-coded into several dynamic programming problems that can be solved efficiently and accurately, again improving the state of the art [4]. However, some structural rules are not expressed in this approach, such as the common vertical alignment constraints of windows. It also is difficult to adapt to new structures because the regularity is defined by hand, problem by problem.

On the contrary, methods based on shape grammars [1, 15, 19, 22, 23, 25, 26, 29] impose strong structural consistencies by considering only segments that follow a hierarchical decomposition corresponding to a combination of the rules defined in the grammar. Analyzing an image consists here in producing a parse tree whose associated segments fit as well as possible with the

observation. Mixed continuous-discrete inference is generally used to produce good parse trees. The inference of the structure of segments can also be separated from the optimization of their size and positions [16]. With this kind of methods, partially or fully occluded scene elements such as wall and windows can be recovered thanks to structural consistency. These methods are also less sensitive to changes of illumination. However, one of their most important limitation is the dependency in the grammar design, that is generally done and tuned manually.

Recently, a couple of methods have been proposed to automatically learn grammars from ground-truth image annotations [19,33]. Although it is common in natural language processing, grammatical inference is scarce in computer vision. To the best of our knowledge, these two methods are the only ones that can tackle the complexity of multi-class facade segmentation over a substantial training set. Both operate on split grammars. Split grammars, in 2D, feature grammar rules where a rectangle image is recursively split vertically or horizontally into subrectangles. We detail both approaches.

Martinovic and Van Gool's method [19] does not operate directly on the image but on an irregular lattice space similar to the one used by Riemenschneider et al. [22] for parsing. For each example in the training set, an instance-specific split grammar is constructed based on the lattice representation, alternating horizontal and vertical split rules. Putting all rules together yields a large grammar describing exactly the training set. These rules are then merged iteratively by a generalization operation, following a Bayesian model-merging technique. Each step of this iteration is relatively expensive because it requires considering as merging candidates all pairs of non-terminals and evaluating the corresponding grammar. After iterating, the resulting merged grammar is both smaller, which leads to faster parsing, and more general, to treat examples that are not in the training set. It seems this approach does not scale well as, in their experiments, the authors limit the training sets to "30 images to keep the induction time within reasonable bounds" [19].

Weissenberg et al. [33] presents an alternative technique to learn split grammars from images with ground-truth annotations. As in Martinovic and Van Gool's method, a parse tree is first constructed for each annotated image in the training set. However, the construction here operates directly in the image space, generating split rules iteratively based on an energy function expressing preference among split line candidates. Nested binary split rules in the same direction are then grouped together to form n-ary split rules. Finally, a compact grammar is generated by greedily merging grammar rules with identical structure (split direction and subcomponents) but different parameters (split positions). The work is validated by a study of the performance of grammar compression, an experiment in facade image retrieval and examples of virtual facade synthesis. But no experiment on using the generated grammars for parsing is reported.

1.2 Overview

Our method for automatically learning grammars from images with ground-truth annotations also operates on split grammars. As the above two methods [19,33], it first generates a large set of rules from the training set, and then compresses and generalizes them. However, it is based on different principles and relies on more powerful grammatical transformations.

An graphical overview of our approach is pictured in Figure 1. We first consider a small, simple-to-write and generic grammar that can describe many kinds of segmentations but that is not constrained enough to impose actual structural regularities. Using these generic rules and a standard parser for split grammars, we parse the training image annotations. It generates a set of parse trees that fit almost perfectly the ground-truth annotations and that can thus be considered as ground-truth parse trees. The instantiated grammar rules occurring in these parse trees are representative of the architecture style of the training sample. However, this set of

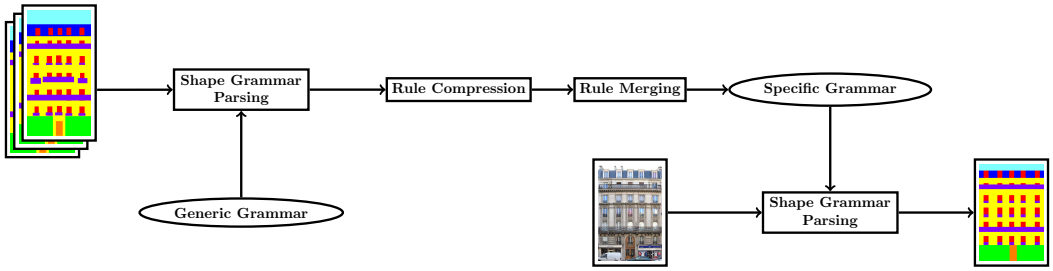


Figure 1: Overall pipeline of the framework.

instantiated rules cannot practically be used as a grammar within a parser because there are too many of them. Indeed, given the enormous combinatorial space to explore, current parsers require a moderate number of rules for inference to succeed. For this reasons, we perform two compression operations. First, we search for common subtrees in the parse trees and merge them into single rules. Second, we cluster rules using an appropriate similarity measure and factor each cluster around a single complex rule. This results in compact grammars that facilitate inference and generalize well the training samples.

In contrast with Weissenberg et al.'s method [33], our learned grammars can be used for parsing. Our learned grammars reach the performance of handcrafted grammars both in terms of parsing time and accuracy of resulting segmentation. On the Haussmannian dataset [29], it also outperforms the grammar generated by Martinovic and Van Gool's method [19] (for a different parser). Besides, our approach addresses the scalability issue of their method. Additionally, we believe our method is also simple to implement.

1.3 Contributions and Organization

The main contributions of our work are the following.

- We propose a new way to generate ground-truth parse trees based on generic grammars. Compared to current approaches, it provides less arbitrary and more deterministic structures.
- Contrary to other methods, the complex rule we generate may combine both horizontal and vertical splits, which captures richer patterns.
- Our rule generalization approach does not rely on a greedy iterative procedure, as in other methods. It is formulated as an unsupervised clustering problem, which is solved efficiently.
- Compared to the other approach that for learning grammars that can be used for parsing, our method scales to training sets with several hundreds of annotated images.
- We provide and discuss experiments on four datasets featuring different architectures styles, including a new Paris Art-deco dataset that we are going to make available to the community. The other datasets are standard and well-known for evaluating facade segmentation. We show that our learned grammars have an equal or better performance than handcrafted grammars or other automatically generated grammars, almost reaching the state-of-the-art of hard-coded segmenters.

The rest of this paper is organized as follows. Sections 2 and 3 briefly describe the concepts of shape grammar and image parsing. Section 4 discusses the kind of grammars we want to

learn. Section 5 presents a method to construct these parse trees and explains why ground-truth parse-tree rules cannot be used directly as a learned grammar. Section 6 details how rules extracted from the ground-truth parse trees can be efficiently compressed. Section 7 explains how to merge rules, which both generalizes and further compress them. Various experiments with this approach are reported and analyzed in Section 8. Section 9 concludes the paper.

2 Split grammars in 2D

Split grammars were introduced by Wonka et al. [34] as a particular kind of shape grammars. The general idea of split grammars is to express the regularity of an object as a recursive decomposition where, at each level, a basic shape of a certain type is split into separate spatial regions that contain smaller basic shapes of some other types. A special case of split grammars in 2D considers as basic shape a labeled image rectangle, that is recursively split horizontally or vertically into labeled subrectangles.

2.1 The grammatical formalism

More formally, a labeled rectangle is denoted as $l(x, y, w, h)$, where l is the label of the rectangle, x, y are its coordinates and w, h are its width and height. A 2D binary split grammar is a 4-tuple $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$ where $\mathcal{N}$ is a set of non-terminal symbols, $\mathcal{T}$ is a set of terminal symbols (disjoint from $\mathcal{N}$), $\mathcal{R}$ is a set of production rules, and $S \in \mathcal{N}$ is a start symbol, also called axiom. A production rule R is of the form:

$$A \quad \rightarrow \quad B \quad \text{or} \tag{1}$$

$$A \quad \overset{a}{\rightarrow}_p \quad BC \tag{2}$$

The left-hand side of the arrow is a non-terminal $A \in \mathcal{N}$. The right-hand side consists of terminals or non-terminals $B, C \in \mathcal{N} \cup \mathcal{T}$. On the arrow, $a \in \{\mathsf{h}, \mathsf{v}\}$ indicates a split axis and $p > 0$ is the split position. The first kind of production rule (1) expresses a mere change of label of the rectangle. The second kind (2) expresses the fact that a rectangle $A(x, y, w, h)$ can be split along axis a at position p into two subrectangles of type B and C:

$$A(x, y, w, h) \quad \rightarrow \quad B(x, y, w, h) \tag{3}$$

$$A(x, y, w, h) \quad \overset{\mathsf{h}}{\rightarrow}_p \quad B(x, y, p, h)\, C(x + p, y, w - p, h) \tag{4}$$

$$A(x, y, w, h) \quad \overset{\mathsf{v}}{\rightarrow}_p \quad B(x, y, w, p)\, C(x, y + p, w, h - p) \tag{5}$$

If $a = \mathsf{h}$, the rectangle is split horizontally (with a vertical split line), which creates two adjacent subrectangles of the same height. If $a = \mathsf{v}$, the split is vertical (with a horizontal split line), which creates two subrectangles of the same width one on top of another. A rule of form (2) is only applicable if $p < h$ when $a = \mathsf{h}$, or $p < w$ when $a = \mathsf{v}$; it then creates two proper subrectangles (not with null height or width).

Terminals represent atomic elements, i.e., rectangles that contain only one or one type of object, e.g., a window, or part of a wall. By definition, a terminal never occurs on the left-hand side of a production rule. Non-terminals represent complex elements that can be broken down recursively into simpler elements until all of them are terminals, e.g., the floor of a building, which can be broken down into wall parts, windows and balconies.

The formalism of split grammars can be enriched with notations that facilitate the writing of grammars. Standard notations includes:

$$A_0 \xrightarrow{a} A_1 \dots A_n \Leftrightarrow \begin{cases} A_0 \xrightarrow{a} A_1 \dots X_1 \\ X_1 \xrightarrow{a} A_2 \dots X_2 \\ \dots \\ X_{n-2} \xrightarrow{a} A_{n-1} \dots A_n \end{cases} \tag{6}$$

$$(A_1 \dots A_n) \Leftrightarrow X \text{ with } X \xrightarrow{a} A_1 \dots A_n \tag{7}$$

$$A_1 \,|\, \dots \,|\, A_n \Leftrightarrow X \text{ with } \begin{cases} X \xrightarrow{a} A_1 \\ \dots \\ X \xrightarrow{a} A_n \end{cases} \tag{8}$$

$$A+ \Leftrightarrow X \text{ with } \begin{cases} X \xrightarrow{a} A \\ X \xrightarrow{a} AX \end{cases} \tag{9}$$

$$A?B \Leftrightarrow X \text{ with } \begin{cases} X \xrightarrow{a} AB \\ X \xrightarrow{a} B \end{cases} \tag{10}$$

$$A_0 \xrightarrow{a}_P A_1 \dots A_n \Leftrightarrow \begin{cases} A_0 \xrightarrow{a}_{p_1} A_1 \dots A_n \\ \dots \\ A_0 \xrightarrow{a}_{p_m} A_1 \dots A_n \\ \text{where } P = \{p1, \dots, p_m\} \end{cases} \tag{11}$$

where X, X_1, etc., are extra auxiliary non-terminals. Note that these are only abbreviations, not a change of paradigm. In particular, a *parameterized rule* $A \xrightarrow{a}_P BC$ is just a factorization of the *meta-rule* $A \xrightarrow{a} BC$ for all the parameters $p \in P$ of *instantiated rules* $A \xrightarrow{a}_p BC$.

This grammar formalism defines context-free shapes: a non-terminal is transformed according to the grammar independently of its context. Such a formalism cannot capture grid regularities, e.g., to model the alignment of windows both horizontally and vertically. Specific implementations, such as Teboul et al.'s parser [29], introduce grammar-level tags to indicate that the splits originating from a given non-terminal should be the same for all occurrences of the terminal. Floors with the same repeated window alignment are models with this kind of tag. We do not address such tags here. To support grid-like shapes, the grammar we want to learn have to encode rich patterns making repetition explicit.

2.2 Derivation trees

A derivation is a top-down view of the decomposition of an object via the grammar. It represents the process and result of recursively splitting a non-terminal into terminal elements. Unless otherwise specified a derivation originates from the start symbol S. Note that, in practice, a grammar generally contains several rules that have a same non-terminals A at the left-hand side of a rule. It introduces non-determinism as different rules can then be applied to split a given rectangle of type A.

More formally, given some rectangular image of size $W \times H$, the basic shape $S(0, 0, W, H)$ is recursively transformed or split into subrectangles as defined by production rules in the grammar. At any point of this process, the input image is tiled into rectangles that have a label in $\mathcal{T} \cup \mathcal{N}$, which provides a semantic interpretation in terms of labeled segments. In theory, this process may not terminate because of possible recursive rules; in practice, binary rules reduce the size of rectangles and lead to bounded derivations. A derivation is complete when no more rule can be applied. In theory, some non-terminal basic shapes $A(x, y, w, h)$ may remain as underived

because the productions rules with A on the left-hand side cannot apply due to split positions p incompatible with the current rectangle. In practice, the grammar is generally designed such that the remaining basic shapes are all labeled in $\mathcal{T}$. The language generated by the grammar, i.e., the set of shapes represented by the grammar, is the set of all possible tilings with terminals only as labels, that can be obtained by a derivation process from S. Alternatives in the production rules generally create a combinatorial explosion of the possible tilings.

A derivation can be represented as a tree with a production rule at each node. The root node is a rule whose left-hand side is the start symbol S, and at any level of the tree, a non-leaf node holding rule $A \to B$ has one child whose rule has B as left-hand side, and a non-leaf node bearing rule $A \xrightarrow{a}_p B\,C$ has two children whose rule have B and C as left-hand side. Such a derivation tree is also called a parse tree. It can be seen a as tree-shaped graphical model associated to the image, that is constructed dynamically rather than fixed. A complete subtree, a.k.a. bottom-up subtree, is a subtree that does not have a non-terminal node at leaf nodes (in right-hand sides for derivation trees).

For simple rules as binary splits, a derivation tree is isomorphic to its derived tree, which has the same tree structure but holds as nodes terminals and non-terminals rather than rules: a non-leaf node holds the non-terminal at the left-hand side of the corresponding rule, and its children hold the right-hand side components of the rule. (In the following, we actually picture derived trees rather than derivation trees for readability.)

Parsing an image consists in looking for the best derivation that explains the image. It is generally based on low-level pixel classifiers and or detectors, that produce a set of probabilities, for each pixel, to be of given types $l \in \mathcal{T}$. A parser typically defines the score of a given parse tree based on a comparison between the "observed" pixel classification probabilities and the "expected", regularized pixel class as defined by the rectangular tiling associated to the parse tree. The goal of the parser is to find a parse tree that minimizes (or maximizes) this score. This search is extremely difficult due to the combinatorial explosion of the possible parse trees.

For this reason, existing inference methods require carefully handcrafted grammars that heavily reduce the search space while mostly preserving the applicability of the grammar to parse targeted images. This allows parser to produce good results within reasonable time limits. Our goal is thus not just to generate a grammar that is compatible with a dataset of images that is representative of an architecture style. It is also to produce a grammar than leads to efficient parsing. For this, the generated grammar has to be as deterministic and specific as possible while preserving enough generality to handle possible cases that are not in the training set.

3 Shape grammar parsing

Image parsing with grammars is a complex and challenging optimization task for two reasons. One, the number of unknown parameters to infer is not fixed and evolves during the optimization process, and two, the inference process involves both discrete (specific derivation rules) and continuous variables (derivation parameters). Prominent methods to tackle this problem are based on (i) reversible jump Markov chain Monte Carlo [23], (ii) evolutionary computation algorithms [26] and (iii) Markov decision processes, in particular reinforcement learning [29]. We focus on the latter in the following as we rely on such a technology for our experiments. Our results are mode general though (see also Section 4).

3.1 Principles of Reinforcement Learning

In reinforcement learning (RL) [27], an agent interacts with an unknown environment while choosing actions that maximize its cumulative reward. The unknown environment is modeled

as a Markov Decision Process (MDP). RL assumes that the environment can be described by a finite number of states, a set of actions and rules that determine the transition between states including reward R for that transition. The agent can perform an action a on some state s leading the environment to a new state s' with some reward. This reward indicates how good the action was from the current state s. Then the optimal value function $Q^*(s,a)$ for taking action a in state s advancing to state s' is defined as

$$Q^*(s,a) = \mathbb{E}[R(s,a) + \gamma \max_{a'} Q^*(s',a')] \tag{12}$$

The parameter γ is a discount factor and represents how much weight we give to the rewards that we will come across in the future. This optimal value function can be approximated using the Q-learning algorithm [27]. The values of all state-action combinations are stored and updated based on samples (s,a,r,s') with a learning rate α according to

$$Q(s,a) \longleftarrow \alpha(r + \max_{a'} Q(s',a')) + (1-\alpha)Q(s,a) \tag{13}$$

From the value function, the greedy policy chooses the action with the maximum value for the current state, $\pi(s) = \arg\max_a Q(s,a)$. For a more detailed description please refer to [27].

3.2 Reinforcement learning for parsing

Reinforcement learning has been successfully applied to solve the shape parsing problem an as optimization of a top-down geometry (from binary split grammars) of the facade, to fit the bottom-up *merit* responses of a pixel-wise classifier [29]. The pixelwise merit $m(l,x,y)$ provides initial semantic information based on classifiers from the image-level features. It expresses the likelihood that the pixel at coordinates x,y is labeled l. The parsing engine is the agent which can be modeled as a MDP. The state s of the agent is (T,N) where T is a derivation tree and N refers to the non-terminal node that is currently being processed. The agent's action a at state s can be any of the grammar rule that is applicable to N, leading to the next state $s' = (T', N')$. The agent's actions are decided by the policy function reward $\pi(s,a) = P(a|s)$, the probability of choosing action a at state s. The agent's goal is to maximize the rewards that are being obtained from its actions. If multiple non-terminal nodes are generated, N refers to the leftmost non-terminal. Otherwise, N becomes the first unprocessed non-terminal encountered while backtracking in the tree. The value function $v^\pi(s)$ and action-value function $Q^\pi(s,a)$ can be calculated as follows.

$$Q^\pi(s,a) = R(s,a) + v^\pi(s(a)) \tag{14}$$

$$v^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a) \tag{15}$$

The different states are the several non-terminal shapes in the grammar for which the rewards are expressed as the sum of its descendant rewards. The goal is to choose a set grammar rules that maximize the reward for the *axiom*. We refer the reader to [29] for more details.

4 What grammars to learn?

Given a training set made of annotated images, we want to learn a grammar that is able to "parse well" unannotated images that are similar to those in the training set. We consider three main aspects to a "good parsing": accuracy, speed (and more generally resource consumption)

as well as stability. Indeed, because the solution space is irregular and huge, most parsers only explore a small portion of this space and can be caught in local minima, yielding sub-optimal results. Besides, most parsers also include randomized procedures and are thus non-deterministic. To quantify the stability of parsing, which is also a indicator of predictability, we measure the variance of parsing accuracy and parsing speed across a test set.

As explained below, the quantity and the nature of choices in a grammar are crucial regarding its performance. There are two sources of alternatives in a split grammar: structural choices (possible combinations of meta-rules) and parameter choices for each of these rules (split positions of instantiated rules).

Accuracy is bounded by the language generated by the grammar. If the grammar is too coarse, it will not be able to express some structural or parametric variations of the objects, leading by force to sub-optimal segmentations. For instance, Teboul et al.'s manual grammar for Haussmannian buildings [29] does not allow wall areas between shop and door, imposes that roof windows are as high as the whole roof, and admits only two kinds of balconies: balcony running over the whole facade width, or balconies being attached to one single window and having exactly the same width. Conversely, if a grammar is too expressive, for instance to possibly cover rare or merely hypothesized cases, the solution space might be too large to search and inaccurate solutions can be produced, although better solutions could exist within the grammar. Speed and stability are also reduced in this case. A balance thus has to be found in the ability of the grammar to cover possible variations. This observation is not restricted to structural choices; it also applies to parameter variation. For example, Teboul et al.'s Haussmannian grammar discretizes split positions with a step of 3 pixels. Although it intrinsically implies sub-optimal splits, it actually results in a better overall accuracy thanks to the reduction of the search space (for a given bound on the number of episodes of the parse).

Besides, different grammars may generate exactly the same language. Yet, some of these grammars may lead to more efficient parses than others. In particular, grammars that impose derivation choices at a time where parsing cues are weak or missing necessitate more backtracking to recover from wrong early choices. Also, some (ambiguous) grammars may have several different analyses for a given shape. It leads to a useless increase of the search space. Some choices may be easier to make than others though.

Note that these properties are not intrinsic to grammars. They also depend on the actual parser that is used. In the following, we consider the case of Teboul et al.'s parser based on reinforcement learning (see Section 3), which is available from the authors and which we have used in our experiments. However, we believe that the general reasoning as well as the qualitative results would be the same with another top-down parser based on a randomized search over the structural and parameter space. In particular, we do not want to rely on any kind of repetition tag to possibly impose identical subderivations, as in Teboul et al.'s parser, e.g., to model grid-like shapes.

As our goal is to prevent experts from having to manually write and tune grammars, our learned grammars should ideally have a similar or better performance than hand-written grammars, regarding accuracy, speed and stability. The difficulties when writing a grammar concerns the control of the expressive power, the specific encoding of complex patterns, and the tuning of parameters to express likely sizes. They have to be addressed automatically.

Finally, as we not only want to learn the structure of objects but also possible object sizes, we assume that all images (in the both the training and the learning sets) present the object more or less at the same scale, i.e., the same number of unit length per pixel. For instance, images in the Haussmannian dataset [29] have been specifically designed to be scaled according to that principle. Images in other datasets have consistent sizes but do not enforce a strict rescaling.

5 Generation of ground-truth parse trees

As observed with formal languages and natural languages, a particularly appropriate data model to learn a grammar is the parse tree. However, training data in our case only consist of annotated images. Parse trees thus have first to be generated from these images.

An annotated image is a pair of images consisting of a real picture and a label image of the same size. In a label image, each pixel is assigned a label in $\mathcal{T}$ identifying the type of the underlying element at the same location in the real image. Label images express the ground-truth segmentation of the corresponding real images.

5.1 Arbitrary, prior-less splits

Different techniques have been proposed to build parse trees from label images [19, 33]. As mentioned in the introduction, Martinovic and Van Gool [19] first tile the label image using the horizontal and vertical axis of segment boundaries, and then merge iteratively these tiles, constructing rules and parse trees on the fly. Weissenberg et al. [33] prefer to define a energy that gives a score to split lines candidates. They use a greedy strategy to recursively split the image using best splits.

These methods have one advantage, which can also be a drawback: they assume no specific knowledge. The problem is that a given label image can be compatible with several parse trees. For instance, a facade with a grid of windows can be analyzed as a set of floors containing rows of windows or as a set of columns of windows, or even as a combination of both. Imposing a minimum description length (MDL) [19] is not enough to single out one particular grammar. As a result, very different parse trees can be generated for very similar facades, resulting in suboptimal factorizations in the learned grammar. A bias can also be imposed to choose specific types of parse trees, e.g., favoring horizontal splits over vertical splits or favoring split axis alternations [33]. But it is hard to control in order to guarantee similar analyses for similar images. To prevent arbitrary analyses of label images, we propose to generate ground-truth parse trees using a *generic grammar*.

5.2 The idea of a generic grammar

The idea is that the generic grammar should be very small and simple, to be written rapidly with no particular expertise and no tuning required: it should not defeat the purpose of automatically learning a full-fledged specific grammar with adapted parameters from annotated images. More than that, it should actually be able to explain a wide range of structures. The same generic grammar should thus make sense for different datasets, e.g., corresponding to different architecture styles.

Note that such a generic grammar only makes sense for the learning task, to generate meaningful parse trees. It cannot be used practically to parse actual facade images, for two reasons. First, it is so general that the solution space would be too large to search, leading to time-consuming and suboptimal parses, and thus to inaccurate and unreliable segmentations. Second, the generic grammar would not be constrained to the specific structure of the learning set. e.g., to a particular architecture style. It thus could not regularize facade analysis with respect to noise (clutter), occlusions or variations of illuminations. However, as shown below, a generic grammar is appropriate to parse ground-truth annotations and generate corresponding ground-truth parse trees.

Tables 1 shows a simple generic grammar that has the same structural expressive power as Teboul et al.'s Haussmannian grammar [28]. Table 2 shows another example of a generic grammar

Simple generic grammar $\mathcal{G}_{\text{sgen}}$		
Axiom	$\xrightarrow{\text{v}}$	GroundFloor Floors RoofFloor sky
GroundFloor	$\xrightarrow{\text{h}}$	shop door shop
Floors	$\xrightarrow{\text{v}}$	wall (Floor wall)+
Floor	$\xrightarrow{\text{h}}$	wall (BalcWins wall)+
Floor	$\xrightarrow{\text{v}}$	balcony WinFloor
WinFloor	$\xrightarrow{\text{h}}$	wall (windows wall)+
BalcWin	$\xrightarrow{\text{v}}$	balcony window
RoofFloor	$\xrightarrow{\text{v}}$	roof (window roof)+

Table 1: The meta-rules of a simple generic grammar that has the same structural expressive power as Teboul et al.'s Haussmannian grammar [29].

that can derive a wide range of facade images comprising the following elements: *wall*, *window*, *balcony*, *roof*, *shop*, *door* and *sky*. Note that these grammars are unambiguous: any resulting segmentation only has one single parse with the grammar. Note also that only the meta-rules are shown. The split parameters of the actual generic grammars are $P = \{1, \ldots, W-1\}$ for horizontal-split rules and $P = \{1, \ldots, H-1\}$ for vertical-split rules.

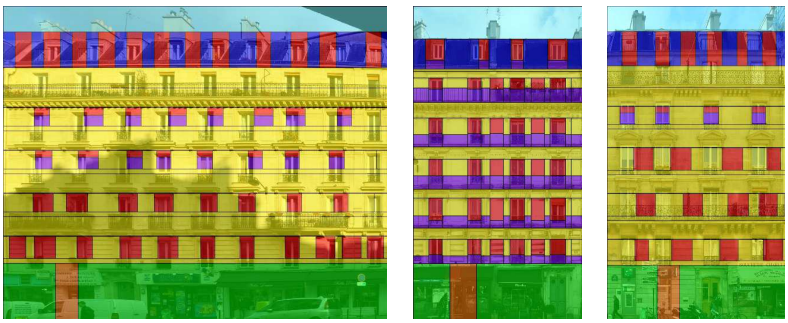


Figure 2: Segmentation maps after 5000 iterations of RL parsing [29] with the generic grammar of Table 1.

As an illustration of the inappropriateness of such grammars to directly parse real pictures, Figure 2 shows a few examples of parses using the simple generic grammar from Table 1 after 5000 episodes of Teboul et al.'s parser [29]. Actually, 5000 episodes are not enough for convergence. This is in contrast with the hand-written compact grammar for Haussmannian facades, where convergence is typically observed within 2000 episodes of RL parsing [29]. Moreover, with this generic grammar, some global structural consistency such as window alignment in columns are not modeled.

5.3 Ground-truth parse trees from a generic grammar

To produce a ground-truth parse tree, we feed the parser described in Section 3 with the generic grammar and the ground-truth label image I_{gt}. Additionally, we replace the usual merit function

Generic grammar $\mathcal{G}_{\mathsf{gen}}$		
Axiom	$\xrightarrow{\mathsf{v}}$	GroundFloor Floors RoofFloor sky
GroundFloor	$\xrightarrow{\mathsf{h}}$	shop? wall DoorWall wall shop?
DoorWall	$\xrightarrow{\mathsf{v}}$	door wall
Floors	$\xrightarrow{\mathsf{v}}$	wall (Floor wall)+
Floor	$\xrightarrow{\mathsf{h}}$	wall (BalcWins wall)+
BalcWins	$\xrightarrow{\mathsf{v}}$	window | balcony Windows
Windows	$\xrightarrow{\mathsf{h}}$	window | wall (window wall)+
RoofFloor	$\xrightarrow{\mathsf{v}}$	roof? RoofWins roof?
RoofWins	$\xrightarrow{\mathsf{h}}$	roof (BalcWin roof)+
BalcWin	$\xrightarrow{\mathsf{v}}$	balcony? window

Table 2: The meta-rules of a generic grammar that can possibly express many facades, with the following segment types: *door*, *shop*, *balcony*, *window*, *wall*, *roof* and *sky*.

based on a pixel classifier by the label image itself:

$$m(l,x,y) = \begin{cases} 1 & \text{if } I_{\mathsf{gt}}(x,y) = l \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

With this definition, the merit of a parse tree is equal to the number of corresponding pixels that are assigned the same label than in the ground-truth annotation. The parser tries to maximize this merit, and thus to produce a parse tree whose associated label image matches as much as possible the ground-truth label image.

Although the parser, equipped with the generic grammar, cannot parse real images (in a reasonable time), it is able to successfully parse the ground-truth label images. The reason is that these images are much more regular and much less noisy than the distribution of label probabilities given by the merit function for real images. It leads to sharper parsing scores and greatly contributes to pruning the search tree. Moreover, as the sampling distribution of split positions in the parser is based on image gradients, the sharp annotations also leads to a small number of sharp peaks in the gradients. There are less decisions to make and good choices are tried first. Some empirical data on parse tree generation using the generic grammar in Table 2 are given in Section 8.

Note that, at least in theory, any parser could actually be used with the same kind of input for generating ground-truth parse trees. For the same reasons as above, we believe that the convergence would be similarly good with other parsing schemes, e.g., based on rjMCMC [19]. Although we could experiment with only one parser (Teboul et al.'s parser [29], that is publicly available), we believe our approach is not tied to a single parser but has general applicability.

5.4 Direct use of parse-tree rules

A grammar specific to the images of a ground-truth training set can be simply produced by just gathering all the rules (including their split parameters) present in the corresponding parse trees. We note such a grammar $\mathcal{G}_{\mathsf{gt}}$.

When generating parse trees using a generic grammar $\mathcal{G}_{\mathsf{gen}}$, the number of meta-rules present in the trees and thus in $\mathcal{G}_{\mathsf{gt}}$ is bounded by the number of meta-rules in the generic grammar $\mathcal{G}_{\mathsf{gen}}$. However, the number of actual rules (with specific split parameters) can be several orders of magnitude larger, as there can be $H-1$ or $W-1$ different instances of a single meta-rule. It grows initially more or less linearly with the number of training images, until most instances relevant for the training set have been encountered. (See also Section 8.)

Such a ground-truth grammar typically comprises most of the rules that are useful to parse an object similar to those in the training set. Even if a few optimal rules are missing because the corresponding split positions do not occur in the training set, close split positions are enough in practice to provide a reasonably good parse. Otherwise it means that the object is not similar to those in the training set.

However, as shown in Section 8, the ground-truth grammar $\mathcal{G}_{\mathsf{gt}}$ cannot be used practically for parsing. It requires a large amount of time for convergence and often results in sub-optimal parses. The reason is that it is too large, which yields a huge space to search. It is also actually too general because it accepts any combination of parse fragments associated to different objects. For instance, for buildings, different floors may have different windows alignments and even different numbers of windows, even if, in the training set, all facades have perfect (but different) window grid alignments. The architecture style is thus not captured.

On the contrary, if we create new instances of non-terminals for the rules of each ground-truth parse tree, i.e., if we generate independent sets of rules for each tree, then the only possible parses are those in the ground-truth. An architecture style can somehow be captured in this way, but the grammar totally overfits the learning data: any new object can only be analyzed as an object of the ground truth. Consequently, previously unseen objects are parsed very inaccurately.

To produce a sensible grammar suitable for parsing objects similar to the ones in the training set, we need the grammar to be general enough not to overfit the data and specific enough to capture the structure of the objects. It should be large enough to cover some unseen cases but small enough to ensure efficient parsing.

Our approach to produce such a grammar consists in compacting the ground-truth grammar by freezing certain rule patterns while preserving enough generality. We do that in two steps. First, we perform a compression of ground-truth derivation, identifying exact repetitions in parse trees. Then we merge similar subtrees of the parse trees. These operations are described in the following two sections.

Our approach consists first in identifying repetitions in each parse tree individually, and consider them as fixed specific patterns. This captures intra-object regularity in the learning set and improves convergence, but it is too restrictive to generalize to unseen objects. In a second step, we cluster these fixed patterns according to a similarity measure and merge them, introducing appropriate generalization.

6 Rule compression

We first consider repetition in a single derivation. More precisely, given a parse tree, we look for complete subtrees that repeat. It identifies patterns within a single object. Specific rules are then introduced to freeze these patterns. (Incomplete subtrees and inter-object patterns are treated in Section 7.)

6.1 Freezing repeated patterns

Many subtrees can repeat within a single parse tree, revealing different levels of structural and parametric regularity. For instance, in a building, there may be several identical instances of

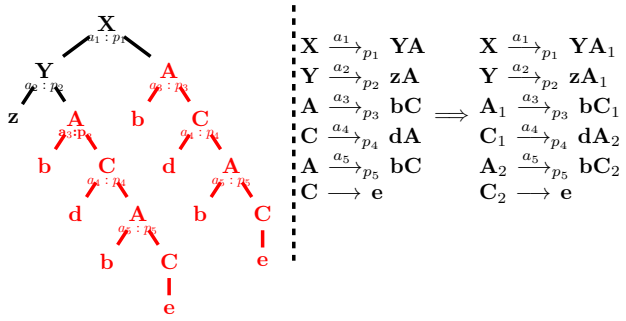


Figure 3: Example of rule compression.

windows with balcony, or several repeated floors with the same layout. We are interested in the largest and most repeating patterns, which we hypothesize are the more likely to be characteristic of a more widespread regularity. More formally, we look for complete subtrees that maximize the number of repeated nodes:

$$\underset{\substack{U \\ subtree(U,T) \\ nbocc(U,T)\geq 2}}{\arg\max} \quad nbocc(U,T)\, size(U) \tag{17}$$

where:

- $subtree(U,T)$ says that U is a complete subtree of T,
- $nbocc(U,T)$ is the number of occurrences of U in T,
- $size(U)$ is the number of nodes in U.

Repetition of subtrees here takes into account both structure and parameters. Two instantiated rules $A \xrightarrow{a}_{p} BC$ and $A' \xrightarrow{a'}_{p'} B'C'$ occurring in the parse tree are identical if $A = A'$, $B = B'$, $C = C'$, $a = a'$ and $p = p'$. However, noise, discretization discrepancies as well as inaccuracies when constructing the ground-truth annotations may result in identical meta-rules $A \xrightarrow{a} BC$ in the parse tree, but with slightly different parameters p, p'. For this reason, we actually consider two instantiated rules to be identical if they stem from the same meta-rule and their parameters p, p' differ only by a few pixels.

Given a repeating subtree U, we then create new rules that represent the pattern only. For this, we duplicate all rules in the subtree, renaming all non-terminals to make sure they are only used once in a rule left-hand side. (In terms of derived trees rather than derivation trees, we rename all non-leaf nodes.) In the following, we note A_i a renamed non-terminal created from an original non-terminal A. The renaming creates as many instances $A_1, \ldots, A_n$ as there are occurrences of A in the subtree. An example of such a transformation is pictured on Figure 3.

This removes non-determinism, if any, wherever the pattern is used. As choices inside the pattern are frozen, the language generated by the resulting grammar rules, for a single parse tree, is smaller. It is as if we had introduced a new, complex n-ary rule representing the whole pattern. Note that this operation is much more general than the rule compression transformation of Weissenberg et al. [33], that only combines splits along one direction, horizontal or vertical. For instance, their transformation cannot handle a floor pattern (which requires horizontal splits) having identical windows with balcony (which requires a vertical split). Another advantage is

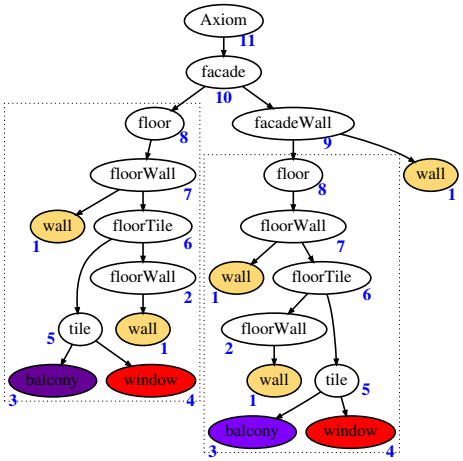


Figure 4: Certificates of a parse tree with 2 repeating subtrees.

that we capture rich patterns into complex rules without the need to change the underlying formalism (splits remain binary) nor the parsers that implement it. As a matter of fact, in all our experiments, we reuse Teboul et al.'s binary split parser as is [29] (cf. Section 3).

6.2 Finding repetition via subtree isomorphism

The number of complete subtrees of a tree T is equal to the number of nodes in T. The simplest and most naive way to find the largest repeating complete subtrees in T is to compare each subtree with all the other subtrees, which is computationally expensive. Several efficient algorithms have been proposed to discover most frequent subtrees in ordered trees [3], making the search mostly linear in the size of the tree. A family of popular approaches turns the issue into a substring matching problem [17, 35]. We prefer to rely on a proposition of Valiente [32] — actually a variant of a folklore method recalled by Flajolet et al. [9] —, which is simple and can be adapted to approximate matching, as required to give some tolerance in rule parameter comparison.

Valiente's algorithm for subtree isomorphism computes a certificate for each subtree in a forest, which is a number between 1 and (at most) the number of nodes in T. The certificate is such that two subtrees have the same certificate iff they are identical. Certificates thus provide a partition of the set of subtrees into isomorphic equivalent classes. The assignment of certificates to subtrees is based on a bottom-up traversal of the tree (see Algorithm 1). When considering a new node, a signature is made from the label of the node and the certificates of its sons, if any ($n \in \{0, 1, 2\}$). A hash table then maps this signature to the associated certificate. If the signature has not been encountered yet, a new certificate is created and used. For example, in Figure 4, both "floor" nodes have a certificate of 8, indicating that both have the same complete subtree starting from these nodes. The complexity is linear on average in the number of nodes in the tree.

Labels in a parse tree are grammar rules, of the form $A \to B$ or $A \overset{a}{\to}_p BC$. Subtree isomorphism between two nodes requires that labels to be equal, i.e., strict rule equality. To perform approximate matching, leaving some tolerance in split position, only the meta-rule part $A \overset{a}{\to} BC$ is used as label in the signature; the parameter p is left out. The hash table now does

Algorithm 1 : Subtree isomorphism

1: $H \leftarrow \emptyset$ *// hash table mapping signatures to certificates*
2: $c_{\mathsf{new}} \leftarrow 0$ *// counter to make new certificates*
3: **for all** u node of T, in bottom-up order **do**
4: let $(c_1, \ldots, c_n)$ be the certificates of u sons, if any
5: $s \leftarrow (\text{label}(u), c_1, \ldots, c_n)$ *// signature of subtree at u*
6: **if** $s \notin Dom(H)$ **then** *// if signature unknown yet*
7: $c_{\mathsf{new}} \leftarrow c_{\mathsf{new}} + 1$ *// new fresh certificate*
8: $H[s] \leftarrow c_{\mathsf{new}}$
9: **end if**
10: $c \leftarrow H[s]$ *// certificate assigned to subtree at u*
11: **end for**

Ensure: identical subtrees $\Leftrightarrow$ identical certificates

Algorithm 2 : getCertificate(p)

1: let $(p_i, c_i, m_i, n_i)_{1 \leq i \leq N}$ be $H[s]$ *// $N = 0$ if undefined*
2: **for all** $1 \leq i \leq N$ **do** *// loop skipped if $N = 0$*
3: **if** $|p - p_i| \leq \theta$ **then** *// approximate equality*
4: $m_i \leftarrow m_i + p$ *// sum of positions similar to p_i*
5: $n_i \leftarrow n_i + 1$ *// number of positions similar to p_i*
6: **return** c_i *// certificate of p is same as p_i*
7: **end if**
8: **end for**
9: $c_{\mathsf{new}} \leftarrow c_{\mathsf{new}} + 1$ *// new fresh certificate*
10: $H[s] \leftarrow (H[s], (p, c_{\mathsf{new}}, p, 1))$ *// new entry added for p*
11: **return** c_{new} *// certificate of p is c_{new}*

not only contain a single certificate c for a given signature s; it contains an association between possibly several positions p_i and corresponding certificates c_i. This allows positions close to p_i to be considered as identical and to be given the same certificate c_i. Moreover, rather than use p_i when generating the pattern rules, we use the average of all positions assimilated to p_i. For this, we also store in the hash table, along with p_i and c_i, the sum m_i of all encountered positions similar to p_i as well as the number n_i of such positions. Later on, when the pattern is used to generate actual grammar rules, it gives access to the mean position m_i/n_i of all positions similar to p_i. This corresponds to replacing lines 6-10 in Algorithm 1 by $c \leftarrow$ getCertificate(p), with $p = 0$ when $n < 2$, and where getCertificate(p) is defined in Algorithm 2.

To find a complete subtree U in T that maximizes term (17), we actually also record the number of times the certificate of U is used. It counts the number of occurrences $nbocc(U, T)$ of subtree U in tree T. After such a U is found, new rules are generated as defined in Section 6.1. The hash table is updated accordingly, and the search for repeated subtrees is iterated.

At this stage a subtree-reduced grammar can be obtained and used for inference. Compared to the generic split grammar (with all possible parameters) or to the parse-tree grammar (set of rules occurring in ground-truth parse trees), the compressed grammar is much smaller in terms of complex rules (counting as one a whole rule pattern) and much more deterministic. Inference is thus much faster. However, the size of the compressed grammar mostly grows linearly with the number of learning images. The reason is that there is no inter-object sharing and no sharing between *similar* patterns, as opposed to identical ones. In fact, in the case of buildings, we

would like to group all facades having the same architectural style independently of the number and values of corresponding attributes. For instance, a 4-window floor could be grouped with a 5-window floor given that the derivation of the former would be a similar subderivation of the latter. This is achieved by the rule merging stage.

7 Rule merging

The rule compression stage (cf. Section 6) freezes intra-object patterns, restricting rule usage. It also drastically reduces the size of the parse trees and of the corresponding grammar. This size reduction allows more complex transformations, which would otherwise be computationally expensive, to capture richer patterns. The rule merging stage described in this section, to be performed after rule compression, is such a transformation. It captures inter-object patterns and generalizes some of the patterns that have been frozen earlier at rule compression stage.

Given parse trees $T_1, \ldots, T_n$ covering all the learning set, we want to identify similar subtrees and group them. The similarity of subtrees here is looser than for rule compression: we still impose structural equality, i.e., equal meta-rules, but we give more freedom to parameters, allowing somewhat different split positions. More importantly, we allow two kinds of rule pattern matching: either a complete subtree U_i of T_i is fully included in a tree T_j (bottom-up matching), or two trees T_i, T_j share a common partial subtree at the root of both T_i and T_j (top-down matching). In both cases, matching is followed by a merging step that shares the pattern across the dataset and generalizes it where each occurrence of the pattern starts to differ.

In our current framework, we first cluster and merge all repeated subtrees identified at the rule compression stage, i.e. recurring subtrees in individual parse trees separately. For this, we use the bottom-up matching scheme. Then, we cluster and merge all parse trees, at root level, with the top-down matching scheme.

7.1 Clustering rule patterns

Rather than use a greedy approach to enumerate groups of similar subtrees, we prefer to define the pattern search as a clustering problem, which is more principled. The idea is that each given tree or subtree is considered as an object to be grouped with other similar trees or subtrees into clusters. Each cluster then corresponds to a pattern. We require the cluster center to be one of the input tree or subtree. We actually define a distance between objects that favors the fact that the cluster center holds the most general part of the pattern. Other objects in the cluster define variations around this core.

This is a standard unsupervised learning problem and existing clustering algorithms can be used. Note however that centroid-based algorithms such as k-means cannot be used here as we require one of the samples to be the cluster center. Recent clustering techniques such as affinity propagation [10] or LP-based clustering [14] have the additional advantages of being insensitive to initialization and of inferring the optimal number of clusters k, around cluster centers $(C_j)_{1\leq j\leq k}$. In our experiments, we employ the LP-based clustering algorithm [14] to minimize the following objective function, which is the sum of the distance of each object to its cluster center:

$$\min_{\substack{k \\ (C_j)_{1\leq j\leq k}}} \sum_{i=1}^{n} \min_{1\leq j\leq k} d(T_i, C_j) + \alpha \sum_{j=1}^{k} \psi(C_j) \tag{18}$$

where

- $d(T, T')$ is the distance between trees T, T' (defined below), satisfying $d(T, T) = 0$,

- $\psi(T) = 1/depth(T)$ is a regularization penalty of choosing T as a cluster center, to avoid the trivial clusterization as a set of singletons, and which favors high trees as cluster centers,
- α is a parameter adjusted to balance the number of clusters, as explained in Section 7.3.

Two different distance functions are used for the two different kinds of merging. The distance d_1 is used for bottom-up clustering and merging. It applies to subtrees identified as repeating in the rule compression stage, measuring the similarity of a subtree completely included in another one. The distance d_2 is used for top-down clustering and merging. It applies to rooted parse trees, measuring how similar the common rooted parts are. They are defined as follows:

$$d_1(T,T') = \begin{cases} \rho(U,T') & \text{if } \exists U\, subtree(U,T) \text{ s.t. } U \equiv T' \\ \rho(T,U') & \text{if } \exists U\, subtree(U',T') \text{ s.t. } U' \equiv T \\ \omega & \text{otherwise} \end{cases} \tag{19}$$

$$d_2(T,T') = \rho(U,U') \text{ where } (U,U') = T \sqcup T' \tag{20}$$

where

- $U \equiv U'$ indicates a structural equality between U and U', not taking into account rule parameters nor non-terminal renaming (cf. Section 6.1),
- $\rho(U,U')$ measures the similarity between structurally equivalent trees $U \equiv U'$ (as defined below),
- $subtree(U,T)$ expresses the occurrence of U as a complete subtree of T,
- $T \sqcup T'$ refers to the largest common part (a.k.a. least general generalization or anti-unification) of T and T', taken from the root, considered as a pair (U,U') of structurally equivalent partial subtrees of T and T', i.e., such that $U \equiv U'$,
- ω is a large value preventing the two trees to be part of the same cluster.

Function $\rho(U,U')$ is defined for structurally equivalent trees $U \equiv U'$, which implies $size(U) = size(U')$. It measures the similarity between the rule parameters $(p_u)_{1\leq u\leq size(U)}$ of U and $(p'_u)_{1\leq u\leq size(U')}$ of U':

$$\rho(U,U') = \frac{1 + \sum_{1\leq u\leq size(U)} |p_u - p'_u|}{size(U)} \tag{21}$$

The value of ρ increases when parameters differ more or when the size of the common part reduces: this favors, in a same cluster, trees that have a large common part and whose parameters differ little. With this definition, d_1 and d_2 are symmetric, and $d_1(T,T) = d_2(T,T) = 0$.

7.2 Merging rule patterns

The merging of rules after clustering is performed as follows. For each cluster $\Gamma = \{T_1, \ldots, T_n\}$, we first consider each instance in each $(T_i)_{1\leq i\leq n}$ of the largest common part $(U_i)_{1\leq i\leq n} = \bigsqcup_{1\leq i\leq n} T_i$ of all elements in the cluster.

Second, we generate a complex rule corresponding to the largest common part. To make sure this rule pattern is "frozen" and specific to the cluster, we rename all non-leaf non-terminals in the largest common part, as in Section 6.1, excluding the start symbol if present. We also group the parameters of instantiated rules into single parameterized rules. More formally, for each meta-rule $A \xrightarrow{a} BC$ in the largest common part, which has instances $A \xrightarrow{a}_{p_i} BC$ in each U_i

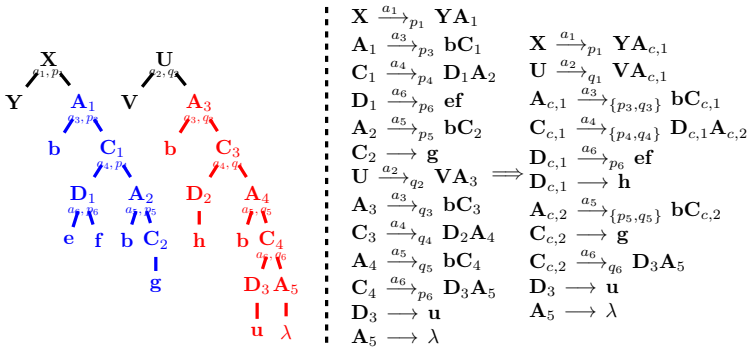


Figure 5: Example of rule merging.

and which is renamed $A_\lambda \xrightarrow{a} B_\mu C_\nu$, we generate a new rule $A_\lambda \xrightarrow{a}_P B_\mu C_\nu$ where $P = \{p_i\}_{1\leq i\leq n}$. For meta-rules of the form $A \xrightarrow{a} B$ in the largest common part, we simply generate a new rule of the form $A_\lambda \xrightarrow{a} B_\mu$ according to the non-terminal renaming.

Last, we need to make sure that the non-terminal B_γ at the root of a newly renamed pattern can be derived from the rules that were deriving B before renaming. This only concerns bottom-up merging; for top-down merging the non-terminal at the root remains the start symbol. For formally, for each rule $A \xrightarrow{a}_P B_i\, C$ such that B_i is the root of the largest common part U_i of Γ in T_i, we generate a new rule $A \xrightarrow{a}_P B_\gamma\, C$. The same applies to rules of the form $A \xrightarrow{a}_P C\, B_i$ and $A \xrightarrow{a}_P B_i$.

An example of such a rule merging (bottom-up case) is shown on Figure 5.

7.3 Adjusting clustering parameters

The clustering result depends heavily on the value of α. A very high value of α results in very few cluster centers with large cluster radius, while a small α value could result in each data-point being a cluster center. In order to determine the optimal value of α, we consider three well-known indices, namely the Dunn's index [8], the Davies-Bouldin index [7] and the Silhouette index [24]. These indices are based on already clustered data. They combines measures of cluster compactness (distances between cluster members) and cluster separation (distances between clusters vs within clusters). Given a distance d, they are defined as follows given k clusters $(\Gamma_i)_{1\leq i\leq k}$ with respective centers $(C_i)_{1\leq i\leq k}$.

Dunn Index [8]: This metric is defined as the ratio between the minimal inter-cluster distance and the maximal intra-cluster distance:

$$D = \frac{\min\limits_{1\leq i<j\leq k} d(C_i, C_j)}{\max\limits_{1\leq i\leq k} \max\limits_{X,Y\in\Gamma_i} d(X,Y)} \tag{22}$$

A higher Dunn index indicates better clustering.

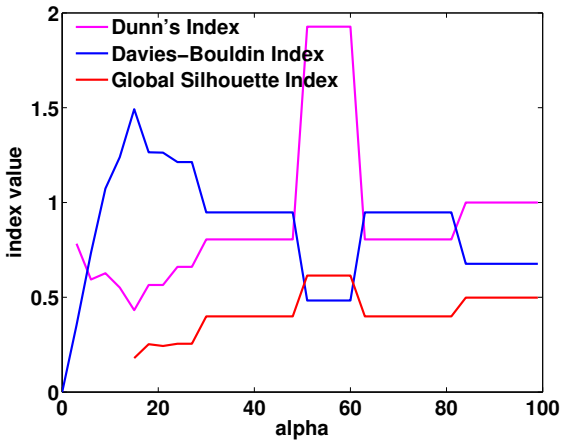


Figure 6: Clustering index plots for 1 fold (ECP2011 dataset).

Davies-Bouldin Index [7]: As Dunn index, this metric measures cluster compactness vs cluster separation. It is defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k}\left\{\max_{\substack{1\leq j\leq k\\ j\neq i}}\left\{\frac{\bar{d}_i + \bar{d}_j}{d(C_i, C_j)}\right\}\right\} \tag{23}$$

$$\bar{d}_i = \frac{1}{|\Gamma_i|}\sum_{X\in\Gamma_i} d(X, C_i) \tag{24}$$

where is $\bar{d}_i$ is the average distance of members of Γ_i to the cluster center C_i. A lower DB value indicates a better separation of the clusters and a greater proximity among members of a cluster.

Global Silhouette Index [24]: Contrary to the previous two indices, this metric takes into account the distance among all members in a cluster, not just with the cluster center. It is defined as:

$$GS = \frac{1}{k}\sum_{i=1}^{k}\left\{\frac{1}{|\Gamma_i|}\sum_{X\in\Gamma_i}\frac{b_i(X) - a_i(X)}{\max(a_i(X), b_i(X))}\right\} \tag{25}$$

$$a_i(X) = \frac{1}{|\Gamma_i| - 1}\sum_{Y\in\Gamma_i, Y\neq X} d(X, Y) \tag{26}$$

$$b_i(X) = \min_{1\leq j\leq k, j\neq i}\frac{1}{|\Gamma_j|}\sum_{Y\in\Gamma_j} d(X, Y) \tag{27}$$

where $a_i(X)$ is the average distance between X and the other elements in the same cluster Γ_i, and $b_i(X)$ is the lowest average distance of X to other clusters. A higher index indicates a better clustering.

Choice of parameter α. The above three indices are used in the experiment section to define α. The best value of α, to produce well-partitioned clusters, corresponds the maximum of Dunn and Davies-Bouldin indices and to the minimum of the Global Silhouette index. Although

	[30]	[29]	[19]	Ours[1]	[18]	[4]	Ours[2]
Door	79	47	50	52	60	79	62
Shop	20	88	81	86	86	94	94
Balcony	35	58	49	55	71	91	84
Window	29	62	66	64	69	85	72
Wall	58	82	80	83	93	90	89
Sky	73	95	91	92	97	97	98
Roof	51	66	71	67	73	90	79
Average	49.3	71.1	69.7	71.3	78.4	89.4	82.5
Overall	48.55	74.71	74.82	76.18	85.06	90.82	86.96
IoU	-	-	-	57.6	-	-	71.8

Table 3: Segmentation results on the ECP2011 dataset. Considered methods are as follows: [29,30] use hand-crafted grammars; [19] segments pixels without strong structural consistencies; [18] and [4] are state of the art. "Ours[1]" are results obtained using our learned grammar ($\mathcal{G}_{\mathbf{L}}$) from the simple generic grammar $\mathcal{G}_{\mathsf{sgen}}$. The first four columns use a merit function based on randomized forest. "Ours[2]" represent results obtained from the generic grammar and DARWIN pixel merits.

they differ in their formulation, these indices mostly agree on the kind of data we are clustering, as can be seen in Figure 6.

8 Results

In this section, we provide both quantitative and qualitative results. We experimented our method on three existing standard datasets of rectified annotated facade images: ECP2011 [29], Graz2012 [22] and CMP2013 [31]. In addition, we experimented with a new dataset with Art-deco architecture style that we have collected specifically to illustrate the applicability of our approach to a variety of structural constraints. Unless otherwise mentioned, we use the generic grammar of Table 2 to generate ground-truth parse trees, as well as DARWIN [11] for the pixel classifier. We first study the accuracy of parsing using the learned grammars: we report classwise accuracy, average class accuracy, overall pixel accuracy and average IoU score. We also evaluate the grammars in terms of scalability, size and inference performance. In all our experiments, unless otherwise specified, we use a 5-fold cross validation setup similar to [18,19], with 80% of the images for learning and 20% for testing.

8.1 ECP2011 dataset [29]

The ECP2011 dataset [29] consists of 104 annotated images of Haussmannian buildings in Paris. For this set of images, we actually use the new, more accurate ground-truth annotations released by Martinovic [18]. We consider two experimental settings. In the first one, we use the pixel merits based on a randomized forest (RF) [30]. This makes our results comparable to published results obtained in the same setting [19,29,30]. In the second one, we use better pixel merits from DARWIN [11], with default settings. It allows us to compare with the state of the art of facade segmentation [4,18]. The feature vector used in DARWIN includes RGB color information, HoG descriptor, LBP texture descriptor and normalized pixel location.

In the testing phase, we run the RL parsing algorithm for a maximum of 1000 iterations using the learned grammar. Table 3 shows a detailed comparison of our method with existing methods.

	[29]	$\mathcal{G}_{\text{sgen}}$	$\mathcal{G}_{\text{gen}}$
episodes	1740	306	580
derivation length	108	28	37

Table 4: Performance comparison of hand-crafted grammar w.r.t. learned grammar on ECP2011: median number of episodes for convergence and average derivation length.

The first four columns provide figures for methods using the RF merits. Our method outperforms the other grammar learning method that we know of and that is useable for parsing [18]. Besides, it has similar or better performance as the hand-rafted grammar, with the same parser. The last three columns of the table compares with state-of-the-art methods that rely on better pixel merits. Our method is slightly below than the state-of-the-art method of Cohen et al. [4]. There might be several explanations. Maybe their pixel classifier is better than DARWIN's. Maybe the search space remains too large and the parser requires more runs and longer runs to obtain a better accuracy. Maybe we encode stronger constraints that facilitate better scores when relaxed. In any case, our approach is much more flexible. Whereas Cohen et al. hard-code structural constraints that are specific to a given class of object, we automatically learn such constraints.

Table 4 provides a comparison of the computational performance of the handcrafted grammar [29] with respect to the learned grammar. The manually-written grammar consists of 19 parametric rules, representing 281 instantiated rules. The average optimal number of clusters while learning from $\mathcal{G}_{\text{gen}}$ was 29 on this dataset. Comparing with the handcrafted grammar, the learned grammar is more efficient at least by a factor of five in terms of number of episodes required for convergence. Furthermore, thanks to such a compact grammar, the average length of the derivation sequence (counting meta-rules are one) is reduced by a factor of three.

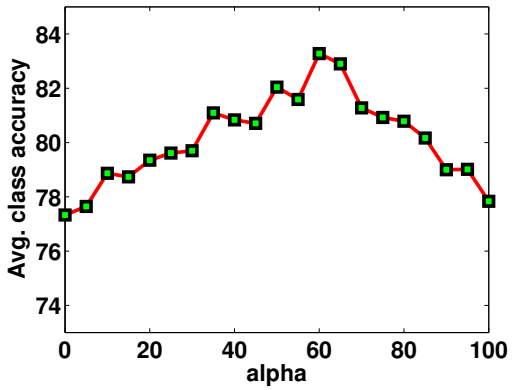


Figure 7: Impact of α on average class accuracy on one fold of the ECP2011 dataset.

To show the role of α, we plot the number of cluster centers against the average number of episodes for convergence and the average class accuracy for a fold (see Figure 7). Intuitively, a high value of α implies fewer number of clusters with large cluster size. This induces a considerable generalization in the learned grammar which can easily result in a suboptimal parse. And for a low value of α, there will be a large number of clusters, shrinking the generalization capability of the learned grammar. An appropriate value of α is one for which the generalization

capacity of the learned grammar is large.

8.2 Paris Art-deco Facades Dataset

We have constructed a new dataset with 80 rectified facade images of *Art-deco* style, from Paris, to illustrate the general applicability of our grammar learning method. Similar to ECP2011, images are annotated into seven classes, *door*, *shop*, *balcony*, *window*, *wall*, *sky* and *roof*. We provide ground truth annotations for these images, which have a rectangular regularity. The dataset is publicly available[1]. This dataset is more challenging compared to ECP2011 due to occlusions and complex structural layout of a facade. The Art-deco style, in general, follows a regular mix of small and large windows. The average number of episodes for convergence was observed to be 670 with the learned Art-deco grammar, while the average derivation length was 30. The number of optimal clusters were found to be 18 for this dataset.

Door	59	34	3	4	0	0	0	+17
Shop	6	88	2	0	4	0	0	+18
Balcony	0	3	63	11	16	1	6	+14
Window	0	0	12	66	14	1	7	+24
Wall	2	5	3	4	84	0	3	+8
Sky	0	0	0	2	0	92	6	+1
Roof	0	0	7	16	15	4	58	+2

Table 5: Confusion matrix for the learned grammar on the Art-deco dataset. Numbers in green show the improvement over pixelwise classification.

8.3 Graz2012 Dataset [22]

The Graz2012 dataset [22] consists of 50 images. A majority of them represent the Gruenderzeit architecture style which is common in Germany and Austria. Classwise accuracies are shown in Table 6. The average number of episodes for convergence was observed to be 180 with the learned grammar, while the average derivation length was 22. The number of optimal clusters was found to be 21 for this dataset. The average number of rules in the clustered grammar is 29.

	Door	Window	Wall	Sky	**Average**	**Overall**	**IoU**
[22]	41	60	84	91	69.0	78.00	58.0
Ours	44	77	91	92	76.1	85.63	68.4

Table 6: Segmentation result on the Graz2012 dataset using the proposed framework. "Ours" represent results obtained using the learned grammar ($\mathcal{G}_{\mathbf{L}}$) with pixel merits from DARWIN.

8.4 CMP2013 Dataset [31]

The CMP dataset [31] contains a mixture of worldwide styles including a majority of Prague buildings. It consists of 378 images of diverse facades with ground-truth annotations initially provided for twelve classes *facade*, *molding*, *cornice*, *pillar*, *window*, *door*, *sill*, *blind*, *balcony*, *shop*, *deco* and *background*. In our experiments, we use only five classes: {*shop, door, balcony, window, wall*}. The *shop*, *door*, *balcony* classes are taken directly while the the new *window* class

[1] `https://github.com/raghudeep/ParisArtDecoFacadesDataset/`

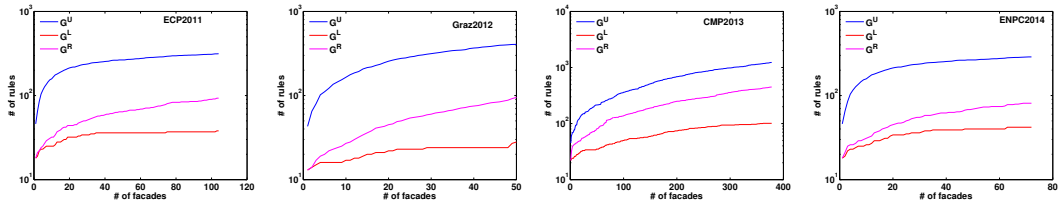


Figure 8: Number of rules in the learned grammar (Y-axis) w.r.t. the size of training set (X-axis). Notice the log scale of Y-axis.

is a combination of window and blind from the original labels. Rest all except the *background* class, correspond to the new *wall* region. Classwise accuracy is shown in Table 7. The average number of episodes for convergence was observed to be 1200 with the learned grammar, while the average derivation length was 32. The number of rules in the learned grammar was about 78.

	Door	Shop	Balcony	Window	Wall	Average	Overall	IoU
[31]	54	59	46	59	84	60.4	78.32	-
Ours	49	66	32	58	89	58.8	82.54	42.4

Table 7: Segmentation results on CMP2013 dataset. **Ours** represent results obtained using the learned grammar $(\mathcal{G}_{\mathbf{L}})$ with pixel merits from DARWIN.

8.5 Qualitative Analysis

We qualitatively evaluate our learned grammars in terms of scalability, size of the learned grammar and performance of inference. To provide an insight on the scalability of our grammar learning method, we plot the number of inferred rules against the size of the training set (see Figure 8). For the ECP2011 dataset, the number of rules in the learned grammar is almost saturated after 25 samples, validating the claim of [33]. And for the Paris Art-deco dataset, the most common rules correspond to (i) two large widely separated windows, on the first and fifth columns, (ii) large window in the middle (third) column, (iii) running balcony on the top floor. For the other two datasets, the number of rules grows with the training samples, indicating the diversity of the dataset. Computations for the rule compressing steps in our implementation took 5 ms per facade on average on the ECP2011 dataset. The clustering step took 15 ms on single core of Intel i7-4770 machine. Please note that rule compression can applied in parallel on all facades of the training set.

Furthermore, we show the performance of different grammars from intermediate stages of the proposed method on the ECP2011 dataset (see Figure 9). For this purpose, we use the following grammars: $(\mathcal{G}_{\mathtt{gt}})$ represents the uncompressed grammar obtained directly from parse trees; $(\mathcal{G}_{\mathrm{R}})$ represents the grammar obtained after the repeating subtree reduction step. To analyze the performance of these different grammars, the RL parsing algorithm is run for a 6000 episodes. While using the generic grammar, the parsing algorithm could not converge in less than 5000 episodes for all the images on all datasets. The parsing algorithm was struck in local optimum while using the uncompressed grammar $(\mathcal{G}_{\mathtt{gt}})$ and the sub-tree reduced grammar $(\mathcal{G}_{\mathrm{R}})$ leading to sub-optimal parse's. Figure 9 shows the impact of the learned grammar in terms of speed.

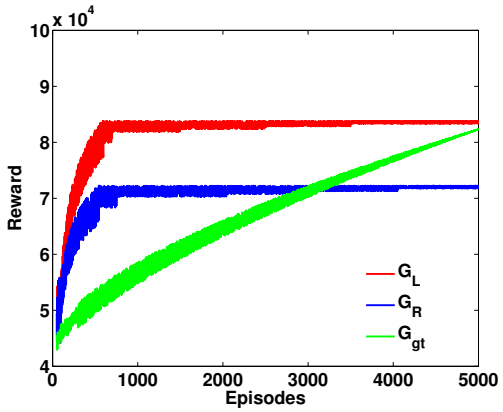


Figure 9: Average learning curves with different grammars on the ECP2011 dataset (average reward at each episode)

8.6 Cross dataset analysis

To investigate commonalities and dissimilarities in grammar rules between different styles of architecture, we operate our learned Haussmannian grammar on Art-deco facades, and the other way around. Figure 10 show such segmentation results on two images. The most common rule between these two styles corresponds to a running balcony on the top floor of a facade. And the most distinctive rules are (i) periodical large windows in the Art-deco style and uniformly-sized windows in the Haussmannian style, (ii) the number of floors: seven in Art-deco and five in Haussmannian. Not only this experiment provides an insight in understanding common rule patterns across different styles, but it also strengthens the need for style-specific grammars. Table 8 shows the performance of grammar learned using Art-deco and haussmannian styles on haussmannian and Art-deco facades.

	$\mathcal{G}_{AA}$	$\mathcal{G}_{AH}$	$\mathcal{G}_{HA}$	$\mathcal{G}_{HH}$
Door	59	56	57	62
Shop	88	86	83	94
Balcony	63	51	54	84
Window	66	56	48	72
Wall	84	71	76	89
Sky	92	82	92	98
Roof	58	68	51	79
Average	72.9	67.1	65.9	82.5
Overall	78.81	71.9	70.8	86.96
IoU	59.4	55.8	57.6	71.8

Table 8: Cross dataset analysis using Art-deco (A) and Haussmannian (H) facades. $\mathcal{G}_{AH}$ represent grammar learned using Art-deco style and applied on Haussmannian facades.Others follow similarly.

Figure 10: Learned haussmannian and art-deeco grammars applied on an art-deco style facade (second and third images from left). The right most image show the difference in the segmentation result

9 Conclusions

In this paper we have proposed a novel method for learning split grammars from annotated images, and we have used it to learn typologies of architectures. The method assumes a simple generic grammar which is used to parse the training set. Reasoning on the associated derivation trees, to first identify common subtrees and then merge similar trees, determines the set of meta-rules corresponding the observed typology of buildings. It leads to a compact (in terms of derivation trees) and simple (in terms of inference process) grammar. State-of-the-art results with respect to typology-specific grammars or grammars learned from data demonstrate the extreme potentials of our method.

Extending this to other typologies of architecture is an ongoing work, such as applying the concept to modern architectures. Such a task will possibly benefit from improved likelihoods of image classes [18]. Improving the process of establishing the set of meta-rules by reasoning simultaneously on the compact derivations of all training examples is a natural extension of our method. Considering more trees at the rule-merging stage should also lead to an improved performance. Furthermore, extending this context to 3D grammars is an extremely promising task, and in particular when taking into account the difficulty of defining such a grammar manually.

Acknowledgments. This work was partly carried out in IMAGINE, a joint research project between Ecole des Ponts ParisTech (ENPC) and the Scientific and Technical Centre for Building (CSTB). It was partly supported by ANR project Semapolis ANR-13-CORD-0003.

References

[1] Fernando Alegre and Frank Dellaert. A probabilistic approach to the semantic interpretation of building facades. In *CIPA International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*, pages 25–27, 2004.

[2] Alexander C Berg, Floraine Grabler, and Jitendra Malik. Parsing images of architectural scenes. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[3] Yun Chi, Richard R Muntz, Siegfried Nijssen, and Joost N Kok. Frequent subtree mining – an overview. *Fundamenta Informaticae*, 66(1):161–198, 2005.

[4] Andrea Cohen, Alexander G Schwing, and Marc Pollefeys. Efficient structured parsing of facades using dynamic programming. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages –. IEEE, 2014.

[5] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[6] Dengxin Dai, Mukta Prasad, Gerhard Schmitt, and Luc Van Gool. Learning domain knowledge for facade labelling. In *Computer Vision–ECCV 2012*, pages 710–723. Springer, 2012.

[7] David L Davies and Donald W Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1(2):224–227, 1979.

[8] Joseph C Dunna. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104, 1974.

[9] Philippe Flajolet, Paolo Sipala, and Jean-Marc Steyaert. Analytic variations on the common subexpression problem. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages 220–234. Springer, 1990.

[10] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[11] Stephen Gould. DARWIN: a framework for machine learning and computer vision research and development. *The Journal of Machine Learning Research*, 13(1):3533–3537, 2012.

[12] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[13] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[14] Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. Clustering via lp-based stabilities. In *Advances in Neural Information Processing Systems 21*, pages 865–872, 2009.

[15] Panagiotis Koutsourakis, Loic Simon, Olivier Teboul, Georgios Tziritas, and Nikos Paragios. Single view reconstruction using shape grammars for urban environments. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1795–1802. IEEE, 2009.

[16] Mateusz Kozinski and Marlet Renaud. Image parsing with graph grammars and markov random fields. In *Winter Conference on Applications of Computer Vision (WACV)*, 2014.

[17] E. Makinen. On the subtree isomorphism problem for ordered trees. *Information Processing Letters*, 32(5):271–273, 1989.

[18] Andelo Martinovic, Markus Mathias, Julien Weissenberg, and Luc Van Gool. A three-layered approach to facade parsing. In *Computer Vision–ECCV 2012*, pages 416–429. Springer, 2012.

[19] Andelo Martinovic and Luc Van Gool. Bayesian grammar learning for inverse procedural modeling. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 201–208. IEEE, 2013.

[20] Pascal Muller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. In *ACM SIGGRAPH 2006 / ACM Transactions on Graphics*, pages 614–623, 2006.

[21] Stanley Osher and Nikos Paragios. *Geometric level set methods in imaging, vision, and graphics*. Springer, 2003.

[22] Hayko Riemenschneider, Ulrich Krispel, Wolfgang Thaller, Michael Donoser, Sven Havemann, Dieter Fellner, and Horst Bischof. Irregular lattices for complex shape grammar facade parsing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1640–1647. IEEE, 2012.

[23] Nora Ripperda and Claus Brenner. Reconstruction of facade structures using a formal grammar and rjmcmc. In *Pattern Recognition*, pages 750–759. Springer, 2006.

[24] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[25] Loic Simon, Olivier Teboul, Panagiotis Koutsourakis, and Nikos Paragios. Random exploration of the procedural space for single-view 3d modeling of buildings. *International journal of computer vision*, 93(2):253–271, 2011.

[26] Loic Simon, Olivier Teboul, Panagiotis Koutsourakis, Luc Van Gool, and Nikos Paragios. Parameter-free/pareto-driven procedural 3d reconstruction of buildings from ground-level sequences. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 518–525. IEEE, 2012.

[27] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[28] Olivier Teboul. *Shape Grammar Parsing: Application to Image-based Modeling*. PhD thesis, Ecole Centrale Paris, 2011.

[29] Olivier Teboul, Iasonas Kokkinos, Loic Simon, Panagiotis Koutsourakis, and Nikos Paragios. Shape grammar parsing via reinforcement learning. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2273–2280. IEEE, 2011.

[30] Olivier Teboul, Loic Simon, Panagiotis Koutsourakis, and Nikos Paragios. Segmentation of building facades using procedural shape priors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3105–3112. IEEE, 2010.

[31] Radim Tylecek. The cmp facade database. Technical report, CTU–CMP–2012–24, Czech Technical University, 2012.

[32] Gabriel Valiente. *Algorithms on trees and graphs*. Springer, 2002.

[33] Julien Weissenberg, Hayko Riemenschneider, Mukta Prasad, and Luc Van Gool. Is there a procedural logic to architecture? In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 185–192. IEEE, 2013.

[34] Peter Wonka, Michael Wimmer, Francois Sillion, and William Ribarsky. Instant architecture. *ACM Transactions on Graphics (TOG).*, 22(3):669–677, 2003.

[35] Mohammed J Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2002.

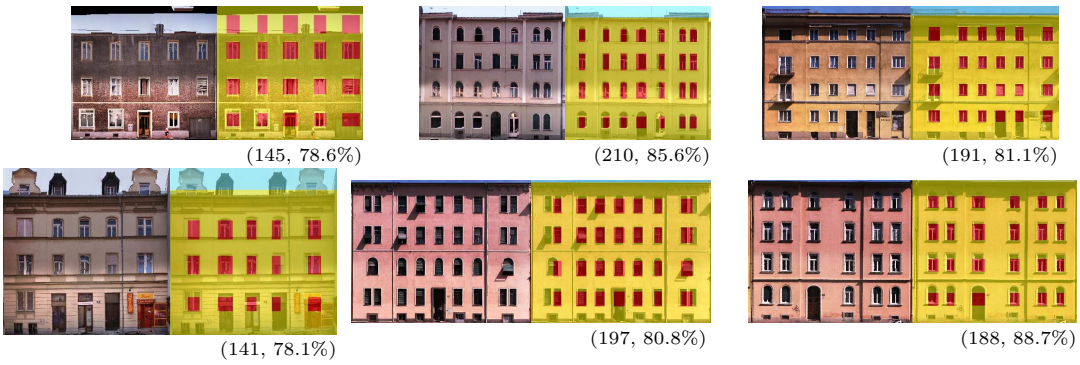


Figure 11: Qualitative results on Graz2012 dataset. Image (left) and segmentation using learned grammar (right) are shown here along with number of episodes for convergence and overall accuracy.

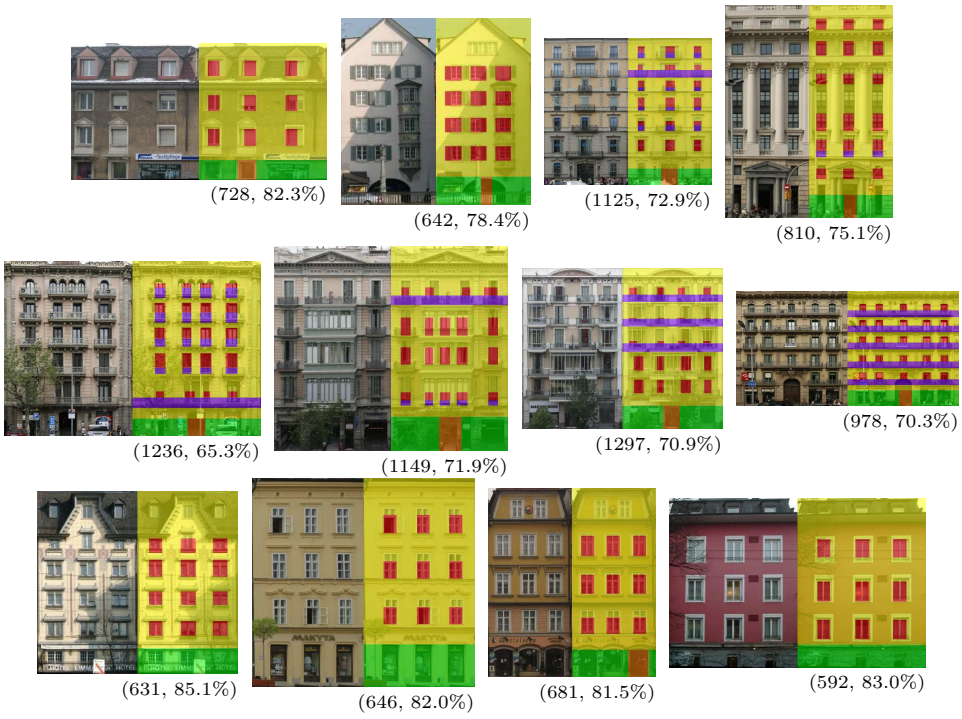


Figure 12: Qualitative results on CMP2013 dataset. Image (left) and segmentation using learned grammar (right) are shown here along with number of episodes for convergence and overall accuracy.

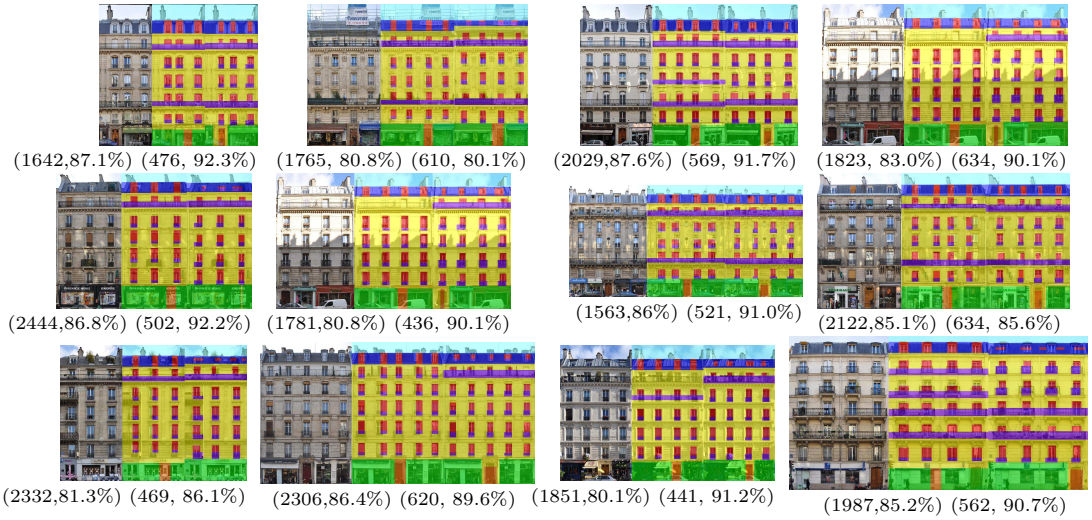


Figure 13: Qualitative results on ECP2011 dataset. Image (left) and segmentation using hand-written grammar (center) and learned grammar (right) are shown here along with number of episodes for convergence and overall accuracy.

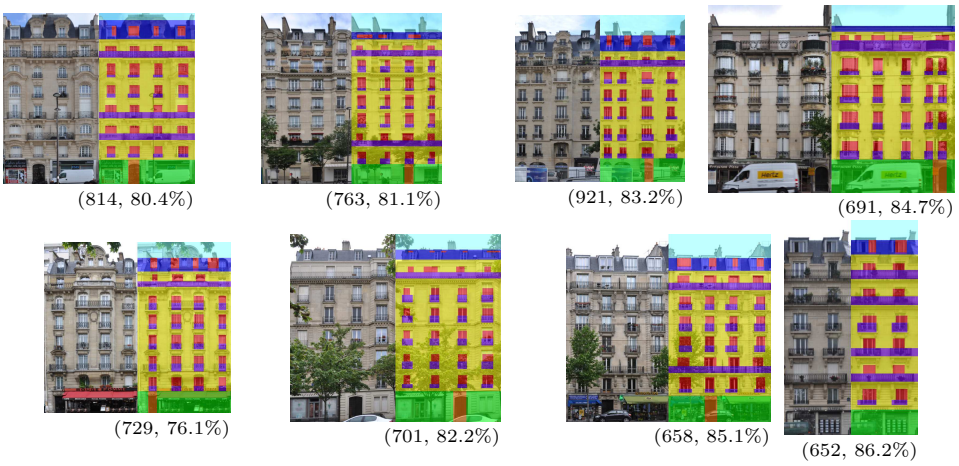
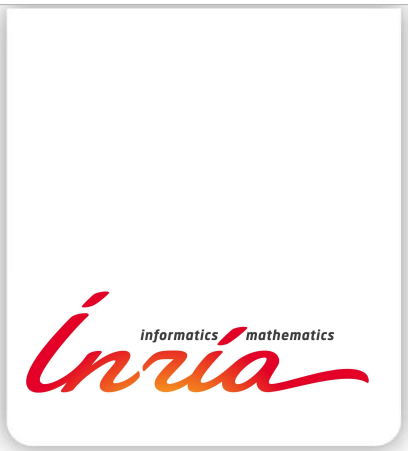


Figure 14: Qualitative results on Paris Art-deco Facades dataset. Image (left) and segmentation using learned grammar (right) are shown here along with number of episodes for convergence and overall accuracy.



RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

1 rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing
Campus de l'École Polytechnique
91120 Palaiseau

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399