Learning Grammars for Architecture-Specific Facade Parsing

Raghudeep Gadde, Renaud Marlet, Nikos Paragios

Learning grammars for architecture-specific facade parsing

Raghudeep Gadde , Renaud Marlet, Nikos Paragios

RESEARCH REPORT

N° 8600

September 2015

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 2015 — 41 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 [45]. 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, i.e., the segmentation of facade images into semantic classes, use either conventional segmentation methods [12, 17, 40] or rely on grammar-driven recognition methods [41, 53, 62]. 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 focus on structural segmentation, i.e., with global regularities and hard constraints as opposed to just local pixel labeling. Our final goal is thus not to produce a state-of-the-art pixelwise classification but to provide a state-of-the-art, high-level, structured view of pictured objects. More precisely, 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 piecewise smoothness. Popular methods are based on active contours [29, 49], clustering techniques such as mean-shift [16] and SLIC [1], and graph cuts [4, 30]. However, although they obtain very good pixelwise 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. While it may be enough, e.g., to get a rough estimate of the percentage of glass area for thermal performance evaluation, it is totally inappropriate to generate building models (BIM), with both geometric and semantic information, as used in the construction and renovation industry. Moreover, as they label only what is visible, ordinary segmentation methods 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. On the contrary, grammar-based methods can infer invisible or hardly visible objects thanks to architecture-level regularity. Conventional segmentation methods may also be sensitive to variations of illumination such as cast shadows, night lighting and glass reflection, although the sensitivity can be partly reduced with larger training sets. Here again, grammar-based priors arguably provide better segmentation in case of "illumination noise" thanks to more global constraints. Actually, grammar-based image parsing methods should not be thought of as alternative segmentation methods but as approaches that take a good pixel classification (a.k.a. unaries) as input and that further impose strong architectural constraints as high-level regularizers. The two kinds of approaches are thus complementary: a better low-level classification or segmentation naturally leads to a better parsing and better overall accuracy (assuming the observed facade follows the architecture style modeled in the grammar).

More accurate segmentations have been obtained adding weak architectural constraints, that are either hard-coded [40] or learned [17], yielding improved pixel classifications, but still breaking fundamental architectural rules such as window alignments or balcony-window relationships. Extra structural constraints have been hard-coded into several dynamic programming problems that can be solved efficiently and accurately, again improving the state of the art [12]. However, some structural rules are still not expressed in this approach, such as the vertical alignment of windows, which is a common constraint. It also is difficult to adapt to new structures and new architectural styles because the regularity is defined by hand, problem by problem.

On the contrary, segmentation methods based on shape grammars [2, 33, 41, 53, 54, 58, 59, 62] make the constraints explicit and thus facilitate the parameterization and adaptation to new architecture styles. They 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 [35], or be completely integrated into constraints not requiring inefficient rule sampling [36]. 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 on the grammar design, that is generally written and tuned manually. It is thus natural to try to learn these grammars automatically.

Although grammatical inference is common in natural language processing (NLP), it is rare in computer vision. Recently, a couple of methods have been proposed to automatically learn shape grammars from ground-truth image annotations [41, 69]. 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 [41] does not operate directly on the image but on an irregular lattice space similar to the one used by Riemenschneider et al. [53] for parsing. For each example in the training set, a specific split grammar is constructed based on the lattice representation, alternating horizontal and vertical split rules. Putting together all rules of all examples 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 however this approach does not scale well as the authors have to reduce the size of the training set to keep the induction time practicable.

Weissenberg et al. [69] present 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 sub-components) 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.

Tu et al. [66] propose a powerful and very general framework for the unsupervised learning of stochastic And-Or grammars which, like ours, is also based on some kind of factorization of similar subtrees. But it is not clear how this approach could be applied to the segmentation of facade images. In this framework, when applied to images, terminals are visual words that are to be connected via spatial relations and structured into a compact hierarchy of nonterminals. This hierarchy is inferred from the distribution of terminals in the training set, maximizing the posterior probability of the corresponding grammar. To apply this generic method, a specific work is required to select appropriate visual words and define relevant spatial relations that can carry across the factorization process. Besides, the learning process starts from a flat representation of all visual words in each image of the training set, along with their relations, whose number can grow quadratically with the number of visual words, and there is no indication in the general framework on a strategy for dropping or merging relations when performing generalization. In fact, examples in [66] are only illustrated on objects with a small and fixed number of components that have well-defined relative positions (well centered animal faces with two ears, two eyes and one nose, among four species of mammals), which is quite different from the case of facades with an unknown number of floors and an unknown number of window columns, and where objects can cover a wide portion of the image area (whole extent of wall, roof, sky, running balconies).

Si and Zhu [57] have a similar approach to learn And-Or grammars. Rather than relying on specific and explicit relations between terminals, it is based on the direct encoding of object presence and position in an occupancy grid. However, the size of this encoding grows with the grid resolution (quadratically in the length of objects), which may raise scalability issues. As a matter of fact, it seems that experiments have been reported up to a 19x19 grid only, which is too coarse for the level of accuracy we target (about 70 to 90 % of pixel accuracy for images of size at least 0.2 Mpixels). Besides, in the case of facade images, similar windows that are just shifted a few squares horizontally or vertically would have a very different representation, leading either to an explosion of alternative cases if they are kept separate (large Or-nodes, i.e., overfitting), or to an excessive generalization if they are merged (large And-nodes containing small Or-nodes, i.e., independent probabilities for neighboring squares). On the contrary, split grammars separate presence (given by rules) and position (given by rule parameters), which greatly reduces the space of configurations to explore and allows an independent factorization of rules and parameters.

It seems that these approaches, based on And-Or grammars and visual words, are more suited for classification and detection tasks (as illustrated by presently reported experiments) than for accurate segmentation. To our knowledge, no experiment with these grammar learning methods has been reported on facade segmentation tasks, at least not on the standard datasets used to evaluate and compare competing methods.

Another interesting aspect of these two approaches, at least theoretically, is the use of stochastic grammars. We actually made experiments of facade parsing with the addition of probabilities to split grammar rules. As it resulted in a minor accuracy improvement, we choose not to burden our grammar learning method with probabilities, for such a small margin. It is seems that fixed rule probabilities are less relevant as guides to explore the space of configurations (rule combinations) than the bottom-up cues specific to a given image [48].

Grammar induction has been studied both in the formal language literature [19] (with applications, e.g., to pattern recognition and RNA structure modeling) and in the NLP community [20]. The formal language literature mainly considers learning from strings based on positive examples, possibly complemented by negative data [26], whereas the NLP community focuses on learning distribution information from hand-annotated parse trees representing positive examples. As for the parsing images, where pixels are (at least) 4-connected, the 2D nature of the problem makes inappropriate most approaches based on learning from strings, as their working principle heavily relies on the 1D associativity of the binary concatenation operator [11, 46, 56]. Learning sets for image parsing typically also consist of positive examples only. As a result, the most relevant literature concerning shape grammar learning lies in the NLP community. (Other approaches such as statistical relational learning and inductive logic programming that have some connections to grammar learning, but currently no obvious links to shape grammars.)

Learning from trees is also a way to escape some of the two-dimensional parsing issues. Parsing 2D data [42, 65] indeed has a much higher complexity than 1D parsing. The orders of magnitude also differ widely: an average English sentence, with about 21 words, whose part of speech (POS) can be determined with an accuracy of 97.3%, has a general accuracy of 56% [39]; a small image with only 300,000 pixels, whose pixel accuracy is at best 92% [27], has an overall accuracy less than $10^{-10,000}$. Considering the noise in input data, image parsing actually is closer to speech processing than to plain text parsing. This situation probably explains why a number of proposed algorithms for image parsing consist of a partial, randomized exploration of an extremely large space, corresponding to derivation trees generated in a top-down manner [58, 59, 62].

Now if the choices for splitting a region vertically or horizontally are already made in the trees of the training set, the grammar induction problem then becomes related to the problem of learning a tree automaton from tree-structured data [7]. Indeed, previous approaches for shape grammar learning involve a first stage of tree hypothesis generation to produce ground-truth parse trees from the ground-truth segmentation, based on heuristics [41, 69]; it is similar to the case of unsupervised data-oriented parsing [6], that considers a subset of all possible binary trees that can be constructed over training strings. In our approach, we propose to generate these ground-truth parse trees differently, using a small generic handwritten grammar, which provides more similar trees in which patterns can be found, as well as interpretable parses (in terms of the generic grammar).

Two simple but useless solutions to grammatical inference are either to construct a flat grammar generating only the examples in the training set (one rule per training sample) or to construct a grammar that considers all strings or structures as parse-able. To prevent these trivial solutions and find a right balance between these two extreme cases, the grammar to infer is typically required to have a certain level of generality, thus allowing to also parse unseen sentences or structures, but not too much not to over-generalize. This can be achieved by introducing mechanisms of rule inference that can generalize patterns in the training set, together with a compactness criterion such as a minimum message length (MML) or minimum description length (MDL) principle [25].

In NLP, parsing can be ambiguous due uncertainties when determining the part of speech of words, and also because of possible spelling errors and unknown words. For this reason, statistical information is also learned from training data for the parser to produce most likely sentence analyses. The nature of this information is however strongly related to the nature of the targeted parser and grammar, e.g., whether it is statistical data for a probabilistic context-free grammar (PCFG) [28], a latent-variable PCFG (L-PCFG) [13], or a data-driven dependency parser [47]. The same situation occurs for shape grammar learning. In our case, as we target Teboul et al.'s parser [62], which does not exploit any data distribution knowledge when sampling production rules, probabilistic information makes little sense. This is consistent with the fact that, for practical shape grammars, the parser at any point only has a few structural choices, i.e., a small number of applicable rules if the split position parameters are ignored. Besides, even if many split positions are possible for the same “meta-rule” according to the grammar, position sampling actually depends on bottom-up cues extracted from the parsed image. What matters most is thus the occurrence or not of certain structural patterns and rule parameters in training data, not their frequency.

The work in NLP that is most closely related to our approach is grammar refinement, which operate on annotated trees to learn distribution information but also to generate specialized rules to represent patterns that could not be captured given strong independence assumptions of grammar rules. This may be achieved with symbol splitting [52], latent variable addition [43], or grammar paradigms richer than plain context-free grammars (CFGs), such as tree substitution grammars (TSGs) [5]. TSGs allows for arbitrarily large tree fragments as rules in the grammar and thereby better represent complex structures. The TSG induction scheme proposed by Cohn et al. [15] relies on a Bayesian non-parametric prior for regularizing the tree fragments to explore as rule candidates, giving a bias towards small grammars with small production rules. This method is different from our approach, where we find repeating subtrees in the data and then perform clustering of these complex subtrees. In our implementation, as our target parser only accepts plain binary split grammars (BSGs) [62], we actually represent complex rules using a flat deterministic decomposition which is similar to what occurs when symbol splitting is performed [52]. The inference of latent variables to construct combined instances of specialized rules seems to be a promising alternative to the rule clustering algorithm that we propose, especially spectral methods [13] that appear to scale well (more or less linearly) when the number of hidden states increases [14], compared to EM-based methods [43], for a similar if not better accuracy. The order of magnitude of the reported number of hidden states (up to 32) [14] is comparable to the number of rule instances we generate. The level of recursion in natural language sentences is however much lower than in the kind of shapes and grammars we consider.

Another aspect is the difference of size of the training corpora. In NLP, the training sets used for grammar induction, such as the WSJ section of the Penn treebank, typically contain more than 30,000 trees (i.e., sentences). Although language constructs are arguably more complex than shape relationships, and thus require more training data, this is at least two orders of magnitude larger than the training sets used here for shape grammar learning, where the number of ground-truth segmentations for learning in our experiments is 40 to 300. This calls for different compromises.

The problem of grammatical inference is also studied in the data compression literature. The goal here is to find the smallest grammar that can generate a given string [3, 8, 9, 37]. However, as the information in this case is of symbolic nature (as opposed, e.g., to signals), compression is generally defined to be loss-less. The grammar is thus designed to generate one and only one string. While some of these techniques can be accommodated to generate a given set of strings, they are not suited for generalization: the grammar is designed to reject any unknown strings, even if it is “similar” to a learned string. These techniques are thus not adapted to our problem, as we need to moderately generalize from the learning set.

1.2 Overview

Our method for automatically learning grammars from images with ground-truth annotations operates on split grammars. As the above two methods [41, 69], 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.

A graphical overview of our approach is pictured in Figure 1. We first consider a small,

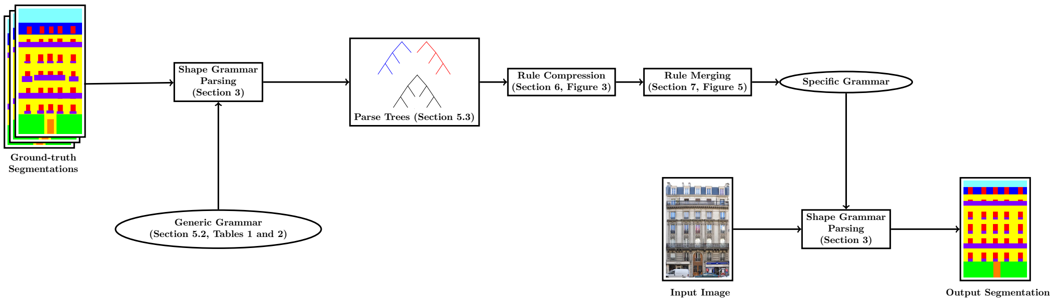

Ground-truth Segmentations
Shape Grammar Parsing (Section 3)
Parse Trees (Section 5.3)
Rule Compression (Section 6, Figure 3)
Rule Merging (Section 7, Figure 5)
Specific Grammar
Generic Grammar (Section 5.2, Tables 1 and 2)
Input Image
Shape Grammar Parsing (Section 3)
Output Segmentation

Figure 1: Overall pipeline of the framework.

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 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 these 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 [69], our learned grammars can be used for efficient parsing. Our learned grammars reach the performance of handcrafted grammars in terms of accuracy of resulting segmentation with better parsing time. On the Haussmannian dataset [62], it also outperforms the grammar generated by Martinovic and Van Gool's method [41] (for a different parser). Besides, our approach addresses the scalability issue of their method.

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 simple, handwritten, generic grammars. Compared to current approaches, it provides less arbitrary and more systematic structures, from which patterns can better emerge, and that can be understood by a human.

- Contrary to other methods [69], 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 for learning grammars that has been 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 Art-deco dataset that we made 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 (that do not enforce all the hard architectural constraints that our generated grammars guarantee).

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 ground-truth parse trees and explains why these 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 following 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. [70] 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 a labeled image rectangle as the basic shape. This labeled rectangle is recursively split horizontally or vertically into labeled sub-rectangles according to the grammar rules.

2.1 The grammatical formalism

More formally, a labeled rectangle of an image 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 (2D-BSG, or BSG for short) 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 *simple rule* in $\mathcal{R}$ has one of the two following forms:

$$A \quad \to \quad B \tag{1}$$

$$A \quad \xrightarrow{a}_p \quad BC \tag{2}$$

The left-hand side of the arrow is a single 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 sub-rectangles of type B and C. The effect of the above rules on a rectangle scope (x, y, w, h) is as follows:

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

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

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

If $a = \mathsf{h}$, the rectangle is split horizontally (with a vertical split line), which creates two adjacent sub-rectangles of the same height. If $a = \mathsf{v}$, the split is vertical (with a horizontal split line), which creates two sub-rectangles 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 sub-rectangles (not with null height or width).

Terminals represent atomic elements, i.e., rectangles that contain only 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:

$$A_0 \overset{a}{\to}_{(p_1,\dots,p_{n-1})} M_1 \dots M_n \Leftrightarrow \begin{cases} A_0 \overset{a}{\to}_{p_1} M_1 X_1 \\ X_1 \overset{a}{\to}_{p_2} M_2 X_2 \\ \dots \\ X_{n-2} \overset{a}{\to}_{p_{n-1}} M_{n-1} M_n \end{cases} \tag{6}$$

$$A_0 \overset{a}{\to} \dots (M) \dots \Leftrightarrow \begin{cases} A_0 \overset{a}{\to} \dots X \dots \\ X \overset{a}{\to} M \end{cases} \tag{7}$$

$$A_0 \overset{a}{\to} M_1 \,|\, \dots \,|\, M_n \Leftrightarrow \begin{cases} A_0 \overset{a}{\to} M_1 \\ \dots \\ A_0 \overset{a}{\to} M_n \end{cases} \tag{8}$$

$$A_0 \overset{a}{\to} \dots M{+} \dots \Leftrightarrow \begin{cases} A_0 \overset{a}{\to} \dots X \dots \\ X \overset{a}{\to} M \\ X \overset{a}{\to} MX \end{cases} \tag{9}$$

$$A_0 \overset{a}{\to} \dots M? \dots \Leftrightarrow \begin{cases} A_0 \overset{a}{\to} \dots \dots \\ A_0 \overset{a}{\to} \dots M \dots \end{cases} \tag{10}$$

$$A_0 \overset{a}{\to}_{\{p_1,\dots,p_m\}} M \Leftrightarrow \begin{cases} A_0 \overset{a}{\to}_{p_1} M \\ \dots \\ A_0 \overset{a}{\to}_{p_m} M \end{cases} \tag{11}$$

where X, X_1, etc., are extra auxiliary non-terminals and where M, M_1, etc., is a concatenation of expressions built on non-terminal and terminal symbols. Note that these are only abbreviations, not a change of paradigm. In particular, a *parameterized rule* $A \overset{a}{\to}_P BC$ is just a factorization of the *meta-rule* $A \overset{a}{\to} BC$ for all the parameters $p \in P$ of *instantiated rules* $A \overset{a}{\to}_p BC$. Tables 1 and 2 provide examples of grammars in this formalism.

Split grammars can also be seen as tree substitution grammars (TSGs) by making explicit the split as a tree operator, with given split axis and position. Simple rewriting rule of the form $A \to B$ stay the same. Other kinds of rule are understood as follows:

$$A \overset{a}{\to}_p BC \qquad \Leftrightarrow \qquad A \to \mathsf{split}_{a,p}(B, C) \tag{12}$$

This allows the definition of *complex rules*, whose right-hand side is a tree with operators $\mathsf{split}_{a,p}$ as non-leaf nodes, and terminals or non-terminals as leafs, e.g.,

$$\mathsf{split}_{a_1,p_1}(\mathsf{split}_{a_2,p_2}(A_1,A_2),\mathsf{split}_{a_3,p_3}(A_3,\mathsf{split}_{a_4,p_4}(A_4,A_5)))$$

. In the following, we will construct complex rules by combining simple rules, e.g.,

$$\left.\begin{array}{l} A_0 \overset{a_1}{\to}_{p_1} A_1 B_1 \\ A_1 \overset{a_2}{\to}_{p_2} A_2 B_2 \end{array}\right\} \quad \Rightarrow \quad A_0 \to \mathsf{split}_{a_1,p_1}(\mathsf{split}_{a_2,p_2}(A_2, B_2), B_1) \tag{13}$$

Note that in a right-hand side tree, the axes $a_1, \ldots, a_n$ of the split nodes are not required to be equal.

This grammar formalism (BSG, or TSG with split nodes) defines context-free shapes: a non-terminal is transformed according to the grammar independently of its context. As such, this formalism cannot capture grid regularities, e.g., to model the alignment of windows both horizontally and vertically. For this reason, Teboul additionally defines a repetition tag [61], that can be put on any non-terminal of the grammar. This tag indicates that all derivations of this non-terminal (see Sect. 2.2) shall be identical w.r.t. its split direction. This variant of the usual binary split grammars allows the expression of grid-like constraints. For instance, if the non-terminal for floors is tagged, all floors will have identical window splits, which will ensure that all windows are vertically aligned. This tag extends the BSG formalism to non context-free grammars (probably to something akin to Type-1 grammars in the Chomsky hierarchy, or possibly a subset, but we have no claim in that respect).

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 the 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 sub-rectangles 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 underlined 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 $A \rightarrow B$ has one child whose rule has B as left-hand side, and a non-leaf node bearing rule $A \overset{a}{\rightarrow}_p B\,C$ has two children whose rule have B and C as left-hand sides. Such a derivation tree is also called a parse tree. It can be seen as a 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 whose leaf nodes contains rules that have only terminals in their right-hand side. This implies the subtree cannot be further derived, as there is no non-terminal whom to attach a corresponding rule as son.

For complex rules, we have to distinguish derivation trees from derived trees. A *derivation tree* in this case represents as above the successive application of grammar rules from an initial non-terminal: nodes of the derivation tree are grammar rules. A *derived tree* is the combination of trees occurring on the right-hand side of the rules, to form a single tree: here, non-leaf nodes of a derived tree are operators $\mathsf{split}_{a,p}$, and leaf-nodes are terminals or non-terminals.

Whereas derivation trees and corresponding derived trees are isomorphic in the case of simple rules (putting aside the case of simple rewritings $A \rightarrow B$), they are not in the case of complex rules, as a single derived tree may originate from different derivation trees. Note also that a derivation tree can be seen as a complex rule that has as left-side the non-terminal of the root rule and as right-hand side the corresponding derived tree. (In the following, we 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 a parser to produce a good result 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 that leads to an 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 [54], (ii) evolutionary computation algorithms [59] and (iii) Markov decision processes, in particular reinforcement learning [62]. In this work, we use the latter for experiments because of the better performance it seems to provide compared to the other methods. Furthermore, the effectiveness of the learned grammar can then be evaluated compared to the handcrafted grammar used in [62] under identical settings (see Section 8 for more details). It actually turns out that our learned grammar is more specific to the architecture style than this handwritten grammar (see also Section 4) and thus provides a better accuracy.

3.1 Principles of Reinforcement Learning

In reinforcement learning (RL) [60], 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), described by a finite set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, transition probabilities P, and expected rewards R consecutive to actions. At time t, the agent in state s_t, takes action $a_t \in \mathcal{A}(s_t)$ leading the agent to a new state s_{t+1} with an immediate reward of r_{t+1}. The transition from state s to s' due to an agent action is subject to the probability $P^a_{ss'}$ and the reward received is an expectation $R^a_{ss'}$ on the distribution $P^a_{ss'}$. Formally, we have:

$$P^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a) \tag{14}$$

$$R^a_{ss'} = E[r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'] \tag{15}$$

The goal of the reinforcement learning agent is to maximize its long term reward which is :

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{16}$$

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. Such a behavior is governed by the agent's policy $\pi(s, a)$, the probability of choosing action a while in state s. This leads to the following state-value function $V^\pi(s)$ and action-value function $Q^\pi(s, a)$:

$$V^\pi(s) = \sum_a \pi(s, a) Q^\pi(s, a) \tag{17}$$

$$Q^\pi(s, a) = \sum_{s'} P^a_{ss'} \big(R^a_{ss'} + \gamma V^\pi(s')\big) \tag{18}$$

For the most optimal policy π^*, the above two equations lead to the following non-linear Bellman optimality equations:

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'} \big(R^a_{ss'} + \gamma V^*(s')\big) \tag{19}$$

$$Q^*(s, a) = \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma \max_{a'} Q^*(s', a')] \tag{20}$$

These optimal value functions can be approximated using several algorithms, for example the Q-learning algorithm. For a further details please refer to [60].

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 pixelwise classifier [62]. 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 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 non-terminal. We refer the reader to [61, 63] for more details.

4 What grammars to learn?

Given a training set consisting of annotated images, we want to learn a grammar that is able to "parse well" similar unannotated images. Three aspects of such a "good parsing" can be considered, that depend both on the parser and on the grammar: accuracy of the resulting segmentation, parsing speed (and more generally resource consumption), and relative repeatability of the results if the same facade is parsed several times. 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 optimum, yielding sub-optimal results. Besides, most parsers also include randomized procedures and are thus non-deterministic. The convergence property of a RL parser can be studied, e.g., by observing the reward and its standard deviation over time [48, 61]. In the following, we focus on accuracy and speed, varying the grammar for a fixed parser; repeatability seems to be a less relevant issue given the experimental data.

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 [62] 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.

Conversely, a grammar may have different analyses for a given shape. Such a grammar is called ambiguous. As an ambiguous grammar for a given language uselessly increases the search space, we would like to learn unambiguous grammars, i.e., grammars for which any segmentation has at most one corresponding parse tree. However, the fact that a grammar is ambiguous is undecidable for context-free grammars. Thus, in practice, the property that a grammar is unambiguous can only be enforced by construction.

Note that these properties are not all intrinsic to grammars. They may 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 similar to another top-down parser based on a randomized search over the structure and parameter space.

As our goal is to prevent experts from having to manually write and tune grammars, our

Simple generic grammar $\mathcal{G}_{\mathsf{sgen}}$		
Axiom	$\xrightarrow{\mathsf{v}}$	GroundFloor Floors RoofFloor sky
GroundFloor	$\xrightarrow{\mathsf{h}}$	shop door shop
Floors	$\xrightarrow{\mathsf{v}}$	wall (Floor wall)+
Floor	$\xrightarrow{\mathsf{h}}$	wall (BalcWins wall)+
Floor	$\xrightarrow{\mathsf{v}}$	balcony WinFloor
WinFloor	$\xrightarrow{\mathsf{h}}$	wall (windows wall)+
BalcWin	$\xrightarrow{\mathsf{v}}$	balcony window
RoofFloor	$\xrightarrow{\mathsf{v}}$	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 [62].

learned grammars should ideally have a similar or better performance than handwritten 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 both the training and the test 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 [62] 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 from $\mathcal{T}$ 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 [41, 69]. As mentioned in the introduction, Martinovic and Van Gool [41] 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. [69] prefer to define an energy that gives a score to split line candidates. They use a greedy strategy to recursively split the image using optimal split positions.

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

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

minimum description length (MDL) [41] 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 [69]. 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.

Table 1 shows a simple generic grammar that has the same structural expressive power as Teboul et al.'s Haussmannian grammar [61]. Table 2 shows another example of a generic grammar 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. (They are unambiguous because, considering the ground-truth segmentation as the input and starting from the axiom, there is always only one single rule that can be applied to consume a part of the input, with the same label(s) as in the rule, and thus only one rule to grow the corresponding derivation tree. Moreover, this consumption must be maximum, i.e., with terminals of greatest extent, otherwise no further rule can be applied and the derivation is not complete.) Note also that only the meta-rules are shown here. 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.

Note that such a generic grammar only makes sense for the learning task, to generate mean-

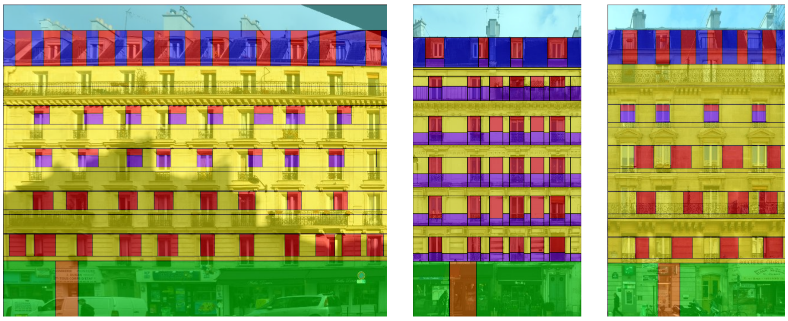


Figure 2: Segmentation maps after 5000 iterations of RL parsing [62] with the generic grammar of Table 1.

ingful 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, i.e, 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.

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 [62]. Actually, 5000 episodes are not enough for convergence. This is in contrast with the handwritten compact grammar for Haussmannian facades, where convergence is typically observed within 2000 episodes of RL parsing [62]. 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 $I_{\mathtt{gt}}$. Additionally, we replace the usual merit function based on a pixel classifier by the label image itself:

$$m(l,x,y) = \begin{cases} 1 & \text{if } I_{\mathtt{gt}}(x,y) = l \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

With this definition, the merit of a parse tree is equal to the number of corresponding pixels that are assigned the same label as 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 label 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 [41]. Although we could experiment with only one parser (Teboul et al.'s parser [62], that is publicly available), we believe our approach is not tied to a single parser but has general applicability.

Besides simplicity, one advantage of this approach is also that the generated ground-truth parse trees can be easily understood, as they reuse the same "concepts" and terms as the generic grammar. This translates as well to the specialized grammars that we infer. For instance, a specific kind of floor in the learned specialized grammar can still be recognized as a floor, and even be given a name derived from the corresponding non-terminal in the generic grammar. (See Section 8.5 for a qualitative analysis of Art-deco facades, made easier with this property.) This is in contrast with the other approaches [41, 69], that have to generate arbitrary names. More importantly, one could argue that the grammar we learn is strongly equivalent [44] to grammars that would be written by hand for the targeted architecture style: the whole structure of the corresponding parse trees should be equivalent up to some kind of isomorphism, not just their leafs, i.e., the underlying segmentation. This is in contrast with grammars made from trees that are generated as heuristic groupings of segments in ground-truth images [41, 69]. In this case, patterns could for instance be discovered for columns of windows rather than for rows; there would then be nothing like a floor in the corresponding parse trees.

5.4 Direct use of parse-tree rules

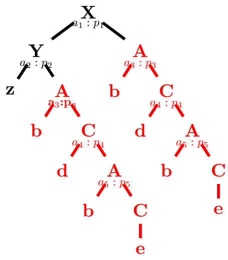
A grammar specific to the images of a ground-truth training set can be simply produced by just gathering all the rules (including their split parameters) present in the corresponding parse trees. Such a grammar is denoted by $\mathcal{G}_{\text{gt}}$.

While 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. Further, it is 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



$$\begin{array}{l} \mathbf{X} \xrightarrow{a_1}_{p_1} \mathbf{YA} \\ \mathbf{Y} \xrightarrow{a_2}_{p_2} \mathbf{zA} \\ \mathbf{A} \xrightarrow{a_3}_{p_3} \mathbf{bC} \\ \mathbf{C} \xrightarrow{a_4}_{p_4} \mathbf{dA} \\ \mathbf{A} \xrightarrow{a_5}_{p_5} \mathbf{bC} \\ \mathbf{C} \longrightarrow \mathbf{e} \end{array} \Longrightarrow \begin{array}{l} \mathbf{X} \xrightarrow{a_1}_{p_1} \mathbf{YA}_1 \\ \mathbf{Y} \xrightarrow{a_2}_{p_2} \mathbf{zA}_1 \\ \mathbf{A}_1 \xrightarrow{a_3}_{p_3} \mathbf{bC}_1 \\ \mathbf{C}_1 \xrightarrow{a_4}_{p_4} \mathbf{dA}_2 \\ \mathbf{A}_2 \xrightarrow{a_5}_{p_5} \mathbf{bC}_2 \\ \mathbf{C}_2 \longrightarrow \mathbf{e} \end{array}$$

Figure 3: Example of rule compression.

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 consists first in identifying repetitions in each parse tree individually, and consider them as instances of the same pattern, specific to the considered facade. This lossless compression 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. These operations are described in the following two sections.

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 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:

$$\operatorname*{arg\,max}_{\substack{U \\ \mathit{subtree}(U,T) \\ \mathit{nbocc}(U,T)\geq 2}} \quad \mathit{nbocc}(U,T)\, \mathit{size}(U) \tag{22}$$

where:

- $\mathit{subtree}(U,T)$ says that U is a complete subtree of T,
- $\mathit{nbocc}(U,T)$ is the number of occurrences of U in T,

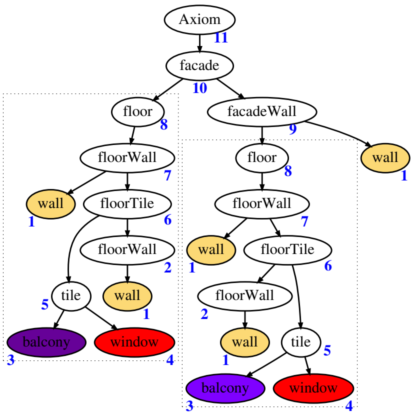


Figure 4: Certificates of a parse tree with 2 repeating subtrees.

- *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 certain threshold (see Section 8).

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. [69], 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 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 [62] (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 [10], making

Algorithm 1 : Subtree isomorphism

```
1: $H \leftarrow \emptyset$   // Hash table mapping signatures to certificates
2: $c_{\mathsf{new}} \leftarrow 0$   // Counter to make new fresh certificates
3: for all $u$ node of $T$, in bottom-up order do
4:     let $(c_1, \ldots, c_n)$ be the certificates of sons of $u$, if any
5:     $l \leftarrow \text{label}(u)$            // Get label of subtree at u
6:     $s \leftarrow (l, c_1, \ldots, c_n)$      // Make signature of subtree at u
7:     $c \leftarrow$ getCertificate(s) // Make/get cert. for subtree at u
8: end for
Ensure: identical subtrees ⇔ identical certificates
```

the search mostly linear in the size of the tree. A family of popular approaches turns the issue into a substring matching problem [38, 71]. We prefer to rely on a proposition of Valiente [68] — actually a variant of a folklore method recalled by Flajolet et al. [22] —, 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 and 2). When considering a new node, a signature is made from the label of the node and the certificates of its n 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 positions, 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 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 actually 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, with corresponding parameters, this information can give access to the mean position $\frac{m_i}{N_i}$ of all positions similar to p_i. For this, a minor change is made to Algorithm 1: we replace line 7 by $c \leftarrow \text{getCertificate}(s, n, p)$, where n is the number of sons and p is the split parameter in case $n = 2$. Procedure getCertificate(s, n, p) is defined in Algorithm 3.

To find a complete subtree U in T that maximizes term (22), 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 ($\mathcal{G}_{\text{st}}$) can be obtained and used for inference. Compared to the generic split grammar (with all possible parameters) or to the parse-tree grammar

Algorithm 2 : getCertificate(s)

```
if $s \notin Dom(H)$ then    // if signature is unknown yet
    $c_{\mathsf{new}} \leftarrow c_{\mathsf{new}} + 1$    // make new fresh certificate
    $H[s] \leftarrow c_{\mathsf{new}}$    // associate it with signature
end if
return $H[s]$    // return certificate associated to signature
```

where $Dom(H)$ is the domain of hash table H.

Algorithm 3 : getCertificate(s, n, p)

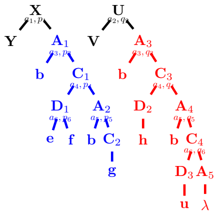
```
if $n \neq 2$ then    // If rule is not a split rule
    return getCertificate(s)    // Return normal certificate
end if    // If rule is a split rule at p
if $s \in Dom(H)$ then    // If signature is already known
    $(p_i, c_i, m_i, N_i)_{1 \leq i \leq k} \leftarrow H[s]$    // Access remembered info.
    for all $1 \leq i \leq k$ do    // For all previously stored $p_i$
        if $|p - p_i| \leq \theta$ then    // if $p \approx p_i$
            $m_i \leftarrow m_i + p$    // Sum positions similar to $p_i$
            $N_i \leftarrow N_i + 1$    // Count positions similar to $p_i$
            return $c_i$    // Yield same certificate as $p_i$
        end if
    end for
end if    // If s unknown or p too different from the $p_i$'s
$c_{\mathsf{new}} \leftarrow c_{\mathsf{new}} + 1$    // Make fresh certificate
$H[s] \leftarrow H[s] \cup \{(p, c_{\mathsf{new}}, p, 1)\}$    // Remember new p for s
return $c_{\mathsf{new}}$    // Return new certificate for s
```

(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. (See Section 8.5 and Table 4 for figures on compression factor and convergence rate.) 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



$$\begin{array}{l}
\mathbf{X} \xrightarrow{a_1}_{p_1} \mathbf{YA}_1 \\
\mathbf{A}_1 \xrightarrow{a_3}_{p_3} \mathbf{bC}_1 \\
\mathbf{C}_1 \xrightarrow{a_4}_{p_4} \mathbf{D}_1\mathbf{A}_2 \\
\mathbf{D}_1 \xrightarrow{a_6}_{p_6} \mathbf{ef} \\
\mathbf{A}_2 \xrightarrow{a_5}_{p_5} \mathbf{bC}_2 \\
\mathbf{C}_2 \longrightarrow \mathbf{g} \\
\mathbf{U} \xrightarrow{a_2}_{q_2} \mathbf{VA}_3 \\
\mathbf{A}_3 \xrightarrow{a_3}_{q_3} \mathbf{bC}_3 \\
\mathbf{C}_3 \xrightarrow{a_4}_{q_4} \mathbf{D}_2\mathbf{A}_4 \\
\mathbf{A}_4 \xrightarrow{a_5}_{q_5} \mathbf{bC}_4 \\
\mathbf{C}_4 \xrightarrow{a_6}_{p_6} \mathbf{D}_3\mathbf{A}_5 \\
\mathbf{D}_3 \longrightarrow \mathbf{u} \\
\mathbf{A}_5 \longrightarrow \lambda
\end{array}
\Longrightarrow
\begin{array}{l}
\mathbf{X} \xrightarrow{a_1}_{p_1} \mathbf{YA}_{c,1} \\
\mathbf{U} \xrightarrow{a_2}_{q_1} \mathbf{VA}_{c,1} \\
\mathbf{A}_{c,1} \xrightarrow{a_3}_{\{p_3,q_3\}} \mathbf{bC}_{c,1} \\
\mathbf{C}_{c,1} \xrightarrow{a_4}_{\{p_4,q_4\}} \mathbf{D}_{c,1}\mathbf{A}_{c,2} \\
\mathbf{D}_{c,1} \xrightarrow{a_6}_{p_6} \mathbf{ef} \\
\mathbf{D}_{c,1} \longrightarrow \mathbf{h} \\
\mathbf{A}_{c,2} \xrightarrow{a_5}_{\{p_5,q_5\}} \mathbf{bC}_{c,2} \\
\mathbf{C}_{c,2} \longrightarrow \mathbf{g} \\
\mathbf{C}_{c,2} \xrightarrow{a_6}_{q_6} \mathbf{D}_3\mathbf{A}_5 \\
\mathbf{D}_3 \longrightarrow \mathbf{u} \\
\mathbf{A}_5 \longrightarrow \lambda
\end{array}$$

Figure 5: Example of rule merging.

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 [23] or LP-based clustering [31] 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 [31] to minimize the following objective function, which is the sum of the distance of each object to its cluster center:

$$\min_{\substack{k \\ (C_j)_{1\leq j\leq k}}} \sum_{i=1}^{n} \min_{1\leq j\leq k} d(T_i, C_j) + \alpha \sum_{j=1}^{k} \psi(C_j) \tag{23}$$

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:

$$d_1(T,T') = \begin{cases} \rho(U,T') & \text{if } \exists U \, \mathit{subtree}(U,T) \text{ s.t. } U \equiv T' \\ \rho(T,U') & \text{if } \exists U \, \mathit{subtree}(U',T') \text{ s.t. } U' \equiv T \\ \omega & \text{otherwise} \end{cases} \tag{24}$$

$$d_2(T,T') = \rho(U,U') \text{ where } (U,U') = T \sqcup T' \tag{25}$$

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),
- $\mathit{subtree}(U,T)$ 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 $\mathit{size}(U) = \mathit{size}(U')$. It measures the similarity between the rule parameters $(p_u)_{1\leq u\leq \mathit{size}(U)}$ of U and $(p'_u)_{1\leq u\leq \mathit{size}(U')}$ of U':

$$\rho(U,U') = \frac{1 + \sum_{1\leq u\leq \mathit{size}(U)} |p_u - p'_u|}{\mathit{size}(U)} \tag{26}$$

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 $A \overset{a}{\to} BC$ in the largest common part, which has instances $A \overset{a}{\to}_{p_i} BC$ in each U_i and which is renamed $A_\lambda \overset{a}{\to} B_\mu C_\nu$, we generate a new rule $A_\lambda \overset{a}{\to}_P B_\mu C_\nu$ where $P = \{p_i\}_{1\leq i\leq n}$. As each simple rule r_j accumulates its own set of specific parameters $P_j = \{p_{j,i}\}_{1\leq i\leq n_j}$, the complex

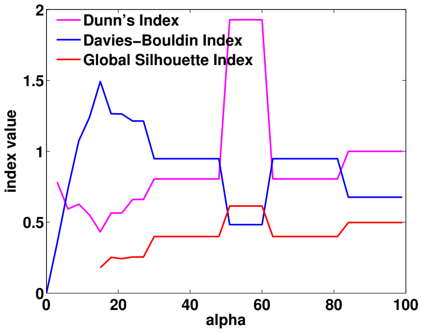


Figure 6: Clustering index plots on the validation set for one fold (ECP2011 dataset).

rule that combines them consequently gets a set of parameter vectors that corresponds to the product of the single-rule parameter sets, i.e., $\prod_{1\leq j\leq k} P_j$. For meta-rules of the form $A \to B$ in the largest common part, we simply generate a new rule of the form $A_\lambda \to B_\mu$ according to the renaming of non-terminals defined by λ.

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. 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 $A \to B_i$.

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

(We think that, if the generic grammar is unambiguous, then the specialized grammars that we generate are unambiguous too. However, we do not have a formal proof of it. In any case, parsing with our generated grammars experimentally has good convergence and accuracy properties, as can be seen from Section 8. Even if some specialized grammars contained a form of ambiguity, it does not prevent us from obtaining good results.)

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 [21], the Davies-Bouldin index [18] and the Silhouette index [55]. 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 [21]: This metric is defined as the ratio between the minimal inter-cluster distance and the maximal intra-cluster distance:

$$D = \frac{\min\limits_{1\leq i<j\leq k} d(C_i, C_j)}{\max\limits_{1\leq i\leq k} \max\limits_{X,Y\in\Gamma_i} d(X,Y)} \tag{27}$$

A higher Dunn index indicates better clustering.

Davies-Bouldin Index [18]: As Dunn index, this metric measures cluster compactness vs cluster separation. It is defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k}\left\{\max_{\substack{1\le j\le k\\ j\neq i}}\left\{\frac{\bar{d}_i+\bar{d}_j}{d(C_i,C_j)}\right\}\right\} \tag{28}$$

$$\bar{d}_i = \frac{1}{|\Gamma_i|}\sum_{X\in\Gamma_i} d(X,C_i) \tag{29}$$

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 [55]: 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:

$$GS = \frac{1}{k}\sum_{i=1}^{k}\left\{\frac{1}{|\Gamma_i|}\sum_{X\in\Gamma_i}\frac{b_i(X)-a_i(X)}{\max(a_i(X),b_i(X))}\right\} \tag{30}$$

$$a_i(X) = \frac{1}{|\Gamma_i|-1}\sum_{Y\in\Gamma_i, Y\neq X} d(X,Y) \tag{31}$$

$$b_i(X) = \min_{1\le j\le k, j\neq i}\frac{1}{|\Gamma_j|}\sum_{Y\in\Gamma_j} d(X,Y) \tag{32}$$

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 Global Silhouette indices and to the minimum of the Davies-Bouldin index. Although they differ in their formulation, these indices mostly agree on the kind of data we are clustering, as can be seen in Figure 6. Rather than select a single index, and as their computation cost is negligible, we choose the value of α such that the ratio $\frac{D\times DB}{GS}$ is maximum, which could add some robustness in case one of the indices would disagree with the others. Other authors [50, 51] have used a similar treatment.

8 Results

In this section, we provide both quantitative and qualitative results using the proposed framework and compare with state-of-art. We experimented our method on three existing standard datasets of rectified annotated facade images: ECP2011 [62], Graz2012 [53] and CMP2013 [67]. In addition, we evaluated our approach with ENPC2014, a new dataset with yet a different architecture style, that we have collected specifically to illustrate the applicability of our approach to a variety of structural constraints and to study the sensitivity of grammar learning to architecture styles.

Most facades pictured in these datasets represent buildings that contain a notable amount of regularity, both across the dataset and within the facade itself. For instance, they typically have at least three floors and at least three windows per floor, that are laid out according to one or two grid-like patterns, with possible variations in position and size though. This is an appropriate setting for segmenting with a grammar-based prior, and also for learning grammatical patterns from just a few tens of annotated samples. On the contrary, grammatical approaches are less suited for datasets that feature facades with little regularity, e.g., with few windows, highly uneven layouts and strong architectural inconsistencies, such as eTRIMS [32]. For such datasets, grammatical priors have to be relaxed [12, 34]. Naturally, trying to learn grammars from such datasets is inappropriate too, especially if the number of images is small, e.g., 60 annotated images in the eTRIMS dataset.

For all our experiments, we use the RL parser made available by Teboul et al. [62], with default settings, on an Intel Xeon E3-1225 CPU 3.2GHz. Unless otherwise mentioned, we use the generic grammar of Table 2 ($\mathcal{G}^2_{\text{gen}}$) to generate ground-truth parse trees, and we use DARWIN [24] with default settings to generate specific pixel classifiers from annotated images, independently for each dataset. We first study the accuracy of parsing using the learned grammars: we report classwise accuracy, average class accuracy, overall pixel accuracy and average intersection-over-union score (IoU). 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 [12, 40, 41], with 60% of the images for grammatical inference and pixel classifier generation, 20% for choosing the value of α, and the remaining 20% for testing. For each experiment with one of our grammar learning method, we thus actually generate 5 pixel classifiers, 5 specialized grammars, and average the resulting figures. Concerning rule compression, we set the similarity threshold mentioned in Section 6.1 to 10 pixels, except for the CMP2013 dataset for which it is set to 30 pixels because the images have a higher resolution.

To somehow compare with Weissenberg et al. [69], despite the fact that they do not evaluate their generated grammars for parsing, we replicated the part of their framework that deals with grammatical inference, namely transformation to n-ary split nodes [69, Sect. 4.1] and production rule inference by parameter merging [69, Sect. 4.2]. Note however that we did not replicate their method for generating ground-truth parse trees [69, Sect. 3.3]; in the following experiments, we always use as ground-truth parse trees the ones we obtain from the generic grammar approach (cf. Section 5.3). The parsing and size comparison with Weissenberg et al.'s method that we provide thus only concerns the grammar generation from *our* ground-truth parse trees.

For the rest of this section, we use the following notations to represent the induced grammars from different steps of different frameworks:

- $[69]^1$, $[69]^2$ represent the grammars induced by n-ary composition [69, Sect. 4.1] and then parameter merging [69, Sect. 4.2], using our implementation of their method and our ground-truth parse-trees.

- $\mathcal{G}_{\mathbf{gt}}$, $\mathcal{G}_{\mathbf{st}}$, $\mathcal{G}_{\mathbf{cl}}$ represent, respectively, the grammar inferred directly from the ground-truth parse trees (Section 5), after subtree reduction (rule compression, Section 6), and after clustering (rule merging, Section 7).

8.1 ECP2011 Haussmannian dataset [62]

The ECP2011 dataset [62] consists of 104 annotated images of Haussmannian buildings in Paris. For this set of images, we use the new, more accurate ground-truth annotations released by Martinovic [40]. We consider two experimental settings. In the first one, we use for grammar inference the simple generic grammar ($\mathcal{G}^1_{\text{gen}}$, shown in Table 1), and for parsing a pixel classifier based on a random forest (RF) [64]. This makes our results directly comparable to published

	.... RF unaries	....						.... DARWIN unaries.......					State of art	
			Grammar induced from $\mathcal{G}^1_{\mathbf{gen}}$					Grammar induced from $\mathcal{G}^2_{\mathbf{gen}}$					(no grammar)	
	[62]	**[41]**	**[69]**^{1}	**[69]**^{2}	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$	**[69]**^{1}	**[69]**^{2}	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$	**[40]**	**[12]**
Door	47	50	20	26	19	41	52	49	54	48	57	**62**	60	**79**
Shop	88	81	84	85	79	85	86	87	89	88	90	**94**	86	**94**
Balcony	58	49	30	42	24	51	55	58	69	66	78	**84**	71	**91**
Window	62	66	24	48	26	58	64	52	59	56	67	**72**	69	**85**
Wall	82	80	74	78	71	78	83	79	83	76	85	**89**	**93**	90
Sky	95	91	**99**	97	95	92	92	**99**	96	96	96	98	97	97
Roof	66	71	33	34	29	63	67	52	58	54	73	**79**	73	**90**
Average	71.1	69.7	51.9	58.6	49.1	66.9	71.3	67.9	72.6	66.5	78.1	**82.5**	78.4	**89.4**
Overall	74.7	74.8	62.9	69.3	59.9	73.1	76.2	74.2	78.6	71.8	82.6	**86.9**	85.1	**90.8**
IoU	-	-	36.5	42.1	34.3	55.4	57.6	54.8	57.3	52.3	67.7	**71.8**	-	-

Table 3: Segmentation results on the ECP2011 dataset: [62] uses a handcrafted grammar; [41] infers a grammar but without strong constraints such as grid alignments; [40] and [12] are state-of-the-art methods with hard-coded constraints (that are soft or that do not cover all architectural constraints).

		Grammar induced from $\mathcal{G}^1_{\mathbf{gen}}$					Grammar induced from $\mathcal{G}^2_{\mathbf{gen}}$				
	[62]	**[69]**^{1}	**[69]**^{2}	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$	**[69]**^{1}	**[69]**^{2}	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$
Convergence time (s)	22	13.4	7.1	24	7.5	**6.6**	18.1	10.8	32	9.8	8.9
# of episodes	1740	1117	695	1956	489	**306**	1421	876	2518	720	580
Derivation length	108	**26**	28	103	42	27	31	35	122	49	37

Table 4: Performance comparison of handcrafted grammar [62] w.r.t. learned grammar on ECP2011: average parsing time, median number of episodes for convergence and average derivation length.

results obtained in the same setting [41, 62, 64], i.e., with the same expressive power of the grammar (e.g., only single-window or whole-facade running balconies) and with the same pixel merits. In the second setting, we use for grammatical inference the richer generic grammar ($\mathcal{G}^2_{\text{gen}}$, shown in Table 2), which allows more architectural variation, and better pixel merits from DARWIN [24]. The feature vector used in DARWIN includes RGB color information, HoG descriptor, LBP texture descriptor and normalized pixel location.

We provide a detailed comparison of our approach with existing methods in terms of both accuracy (Table 3) and convergence time (Table 4). For the grammars that we generate, we run the RL parsing algorithm for a maximum of 10 seconds per image. For Teboul et al.'s RL parser with a handwritten grammar [62], we report the figures given by Martinovic and Van Gool [41] as the figures first provided by the authors were in a different setting, with a less accurate ground-truth [62].

With weak pixel merits from a RF classifier, our method performs better than the handcrafted grammar from [62] and better than the generated grammars from [41]. Comparing with the grammar induction framework from [69], we achieve better segmentation result with our learned grammar at a faster convergence rate. The manually-written grammar consists of 19 parametric rules, representing 281 instantiated rules. Comparing with the handcrafted grammar, the learned

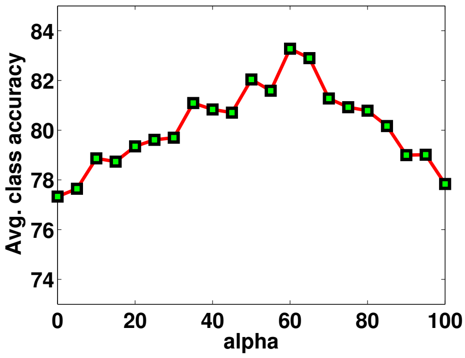


Figure 7: Impact of α on average class accuracy on the test set of the fold from Figure-6 (ECP2011 dataset).

grammar ($\mathcal{G}_{\text{cl}}$) from $\mathcal{G}^1_{\text{gen}}$ is more efficient at least by a factor of five in terms of number of episodes required and by a factor of three in terms of wall clock time for convergence. One of the reasons might be that, thanks to such a compact grammar, the average length of the derivation sequence (counting complex rules as one) is reduced by a factor of three. Although we compare favorably to grammar-based methods, whether the grammar is written by hand or learned automatically, and even to some weakly-constrained segmentation methods [40], our approach does not reach the accuracy of the state-of-the-art hard-coded segmentation method [12]. It might be due to the fact that they use a very good pixel classifier and/or because they do not try to enforce as many hard constraints as we do.

To show the role of α, we plot the value of α against the average class accuracy for one fold on the ECP2011 dataset (see Figure 7). Intuitively, a high value of α implies fewer number of clusters with large cluster size. This induces a major generalization in the learned grammar, which enlarges the search space, potentially leading to 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; the learned grammar would overfit the training data and not be adapted to unseen images, leading to inaccurate parses. An appropriate value of α is thus one for which the generalization capacity of the learned grammar is balanced. This can be seen from Figure 7 by observing the average class accuracies for very high and very low values of α. Note that the best value for α in this case happens to be about the same as the one we compute automatically in a similar setting (see Figure 6). Figure 14 shows some visuals results.

8.2 Graz2012 Dataset [53]

The Graz2012 dataset [53] consists of 50 images. A majority of them represent the Gruenderzeit architecture style which is common in Germany and Austria. As there are only 4 classes in this dataset, namely *door*, *window*, *wall* and *sky*, we had to downgrade the generic grammar to discard the other terminals, i.e., *shop*, *roof* and *sky*.

Classwise accuracies are shown in Table 5, with a comparison to Riemenschneider et al.'s method [53]. Our learned grammar outperforms the other methods. The average number of episodes for convergence was observed to be 180 with the learned grammars, while the average derivation length was 22. The number of optimal clusters was found to be 21 for this dataset, and the average number of rules in the clustered grammar was 29. Figure 12 shows some visual results.

		DARWIN unaries				
	[53]	**[69]**^{1}	**[69]**^{2}	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$
Door	41	29	33	31	39	**43**
Window	60	64	66	62	69	**76**
Wall	84	84	87	82	89	**91**
Sky	91	**95**	94	93	92	92
Average	69	68.2	70.1	66.9	72.3	**75.6**
Overall	78.0	79.3	81.9	77.4	83.9	**86.6**
IoU	58.0	61.6	63.2	59.4	63.1	**68.4**

Table 5: Results on the Graz2012 dataset.

		DARWIN unaries				
	[67]	**[69]**^{1}	**[69]**^{2}	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$
Door	**54**	38	39	46	45	49
Shop	59	61	63	59	63	**66**
Balcony	**46**	26	27	24	25	32
Window	**59**	44	46	49	52	57
Wall	84	81	83	76	86	**89**
Average	**60.4**	50.2	51.6	50.9	54.2	58.8
Overall	78.3	70.4	72.34	67.7	75.6	**82.5**
IoU	-	35.5	37.8	34.5	39.7	**42.4**

Table 6: Results on CMP2013 dataset.

8.3 CMP2013 Dataset [67]

The CMP dataset [67] 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 eleven classes *facade*, *molding*, *cornice*, *pillar*, *window*, *door*, *sill*, *blind*, *balcony*, *shop* and *deco*, plus one class for the *background*, corresponding to cropped areas after image rectification. To enable a comparison of our method across different datasets and different kinds of architecture, we did not try to extend the generic grammar $\mathcal{G}^2_{\text{gen}}$ to cover all the extra classes. We had to adapt it nonetheless because the dataset does not include classes *sky* and *roof*. We thus downgraded the generic grammar to the five classes *shop*, *door*, *balcony*, *window* and *wall*.

To compare the resulting accuracy with the figures reported by Tylecek [67], we also had to merge or ignore some of his classes, based on the reported covariance matrix. While the *shop*, *door* and *balcony* classes are taken directly, the accuracy we give for the *window* class in [67] is actually a combination of the figures for the original labels *window* and *blind*. Similarly, all the other labels are merged into a unique *wall* class, except the *background* class that is ignored by all methods. Classwise accuracy is shown in Table 6. The average number of episodes for convergence was observed to be 1200 with the learned grammars, while the average derivation length was 32. The average number of rules in the learned grammar was 78. Figure 13 shows few visual results.

8.4 ENPC2014 Art-deco Dataset

The Haussmannian style, as illustrated in the ECP2011 dataset, features facades with high regularity, not only regarding window layout but also concerning window sizes, which often have the same width across an entire facade. To demonstrate that architecture-specific grammars are required for a better parsing (see Section 8.6), a dataset with identical semantic classes but different architecture style is needed. For this reason, we have constructed a new dataset, called ENPC2014, with 79 images of Art-deco buildings in Paris. Although they have commonalities with Haussmannian facades, Art-deco facades actually differ, in particular in the typical sizes of windows (which can be wider) and in the number of floors (which can be higher). The balcony layout may also be different. Besides, the dataset includes some layout inconsistencies due to image rectification as some windows and balconies are often protruding in the Art-deco style. It is similar to the case of roof windows, already present in the ECP2011 Haussmannian dataset, which often are not in the same plane as the other facade windows. Similar to ECP2011, images in ENPC2014 are segmented and annotated into seven classes, *door*, *shop*, *balcony*, *window*, *wall*,

	DARWIN unaries				
	[69]^1	**[69]**^{2}	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$
Door	49	53	41	56	**59**
Shop	78	84	78	85	**88**
Balcony	49	57	46	57	**63**
Window	51	59	46	58	**66**
Wall	72	79	78	77	**84**
Sky	**97**	96	95	95	92
Roof	52	54	49	56	**58**
Average	64.1	68.9	61.8	69.1	**72.9**
Overall	68.4	74.3	69.5	73.4	**78.8**
IoU	48.0	57.8	48.2	55.1	**59.4**

Table 7: Segmentation results on the ENPC2014 dataset.

sky and *roof*. The segments in the ground-truth annotations we defined follow a rectangular regularity, but no alignment is artificially enforced. The dataset is publicly available[1].

Concerning our experiments, we use the same generic grammar $\mathcal{G}^2_{\text{gen}}$ as with the other datasets to generate our specialized grammars. Accuracy results are reported in Table 7. The average number of episodes for convergence was observed to be 670 with the learned Art-deco grammars, while the average derivation length was 30. The number of optimal clusters were found to be 18 for this dataset. Figure 15 shows few visual segmentations on this dataset.

8.5 Scalability and qualitative analysis

To provide an insight on the scalability of our grammar learning method, we plot in Figure 8 the number of inferred rules against the size of the training set. For the ECP2011 dataset, the number of rules in the learned grammar is almost saturated after 25 samples, validating the claim of [69]. For the ENPC2014 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. (Such an interpretation in made easier by the fact that our inferred grammars are generated from a generic grammar that already has an understandable semantics.) For the datasets CMP2013 and ENPC2014, the number of rules continues to grow with the training samples, indicating the diversity of the dataset and underlying architecture styles. Note that the numbers of rules provided here by our implementation of Weissenberg at al.'s method are a bit smaller than the values reported in the authors' paper [69]. This could be explained by the fact that we find more complex rule patterns, as we can combine both horizontal and vertical splits, and/or by the fact that our input ground-truth parse trees generated from a generic grammar display more regularity than the parse trees discovered by the heuristics in [69].

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 Xeon E3-1225 machine. As for extracting the ground-truth parse trees, we run the RL parser for a maximum of 15 s per annotated image. However, convergence was observed in 4.8 s on average. Please note that rule compression can be applied in parallel on all facades of the training set. Martinovic et al. do not report running times but it seems their approach does not scale well as, in their experiments, the authors limit the training sets to "30 images to keep the induction time

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

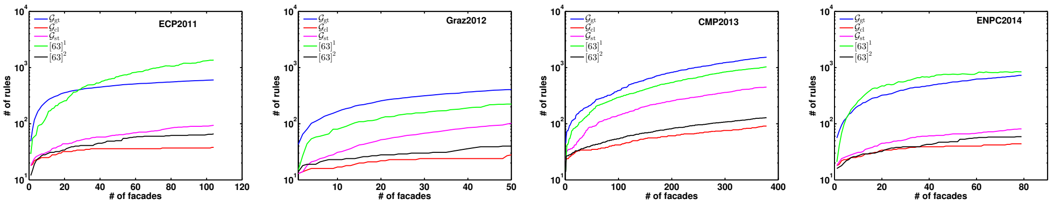


Figure 8: Number of rules in the learned grammar (Y-axis) w.r.t. the size of training set (X-axis). Notice the log scale of Y-axis.

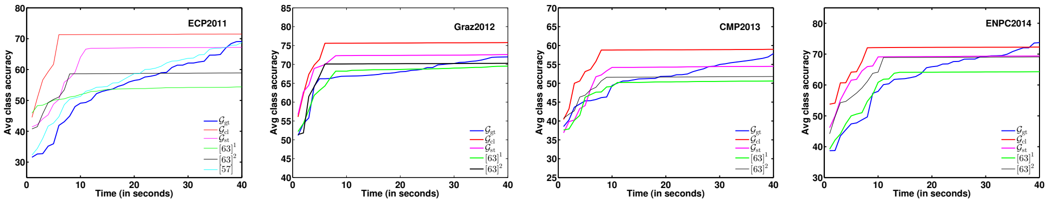


Figure 9: Average class accuracy (y-axis) w.r.t. time (x-axis).

within reasonable bounds" [41]. Weissenberg et al. [69] report that their inference algorithm takes about 32 ms per facade on an Intel Core i7 930. Their inference method is mostly linear and works online. As we are currently relying on the LP-based clustering of Komodakis et al. [31], our implementation is not online, but it could be made so using an online clustering algorithm. In any case, learning time probably is not an issue given the current performance and the typical size of the training sets. Larger orders of magnitude for the number of images with handmade ground-truth annotations would defeat some of the interests of generating a grammar automatically to reduce the human burden on this task.

Figure 9 shows the performance of induced grammars from different stages of our framework and also a like-for-like comparison with different grammars obtained by our implementation of Weissenberg et al.'s framework. For these experiments, the RL parser is run for 40 seconds and the average class accuracy is plotted with respect to time.

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 shows 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.

	$\mathcal{G}_{\mathrm{AA}}$	$\mathcal{G}_{\mathrm{AH}}$	$\mathcal{G}_{\mathrm{HA}}$	$\mathcal{G}_{\mathrm{HH}}$
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	78.8	71.9	70.8	87.0
IoU	59.4	55.8	57.6	71.8

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

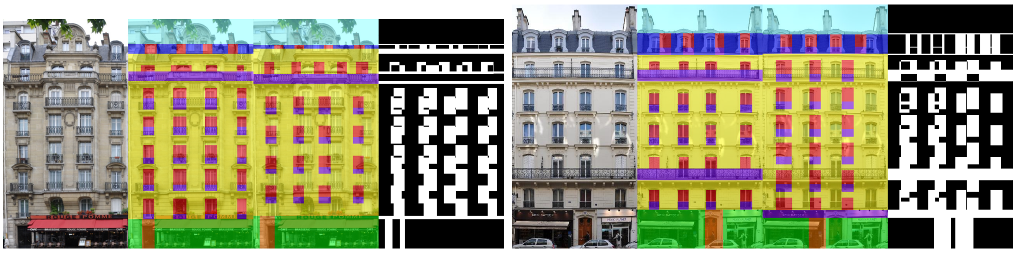


Figure 10: Cross-comparison of learned Art-deco and Haussmannian grammars on Haussmannian and Art-deco facades. Top, from left to right: Art-deco facade, analyzed with Art-deco grammar, analyzed with Haussmannian grammar, and disagreement map (white color for differences). Bottom: Haussmannian facade, analyzed with Haussmannian grammar, analyzed with Art-deco grammar, disagreement map.

8.7 Sensitivity to the accuracy of pixel classifiers

A question that arises is how much the underlying pixel classifier, that the parser uses to evaluate sampled layout configurations, impacts the performance of a given grammar. (Note that the goal here is not to reach the best pixelwise accuracy possible, but still to perform a structural segmentation that follows architectural constraints.) For this, we experimented with 4 different unaries: random forests (RF) [64], DARWIN [24], Auto-Context (AC) [27], and the ground-truth (GT) labeling itself. We consider 6 different grammars (or more precisely, 6 families of grammars in the case of grammar generation, as we follow a 5-fold cross-validation in this case): the handwritten grammar of Teboul et al. [62], grammars generated by the two variants of Weissenberg et al.'s method $[69]^1$ and $[69]^2$, and grammars generated by our 3 variants $\mathcal{G}_{\mathrm{gt}}$, $\mathcal{G}_{\mathrm{st}}$, $\mathcal{G}_{\mathrm{cl}}$. For all experiments, the RL-based parser is run under identical settings, with 2000 iterations. Figure 11 shows the overall pixel accuracy on the ECP2011 dataset for these 4 pixel classifications and 6 grammars. As can be seen, for a given grammar, the better the pixel classifier, the better the resulting pixel accuracy after parsing. Besides, the quality ranking of the grammars is preserved when the accuracy of the pixel classification increases. This shows that the quality of the grammars (or grammar generators) is relatively independent of the underlying pixel classifier

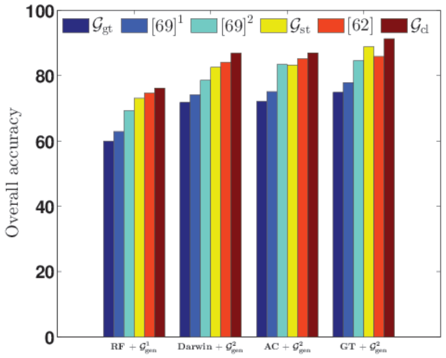


Figure 11: Performance of hand-crafted and learned grammars using different unaries on the ECP2011 dataset.

used by the parser. In these experiments, our approach consistently performs better.

9 Conclusion

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 handcrafted grammars or to 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 [40]. 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 approach 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 and European Research Council Grant DIOCLES (ERCSTG-259112).

References

[1] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Susstrunk, S.: SLIC superpixels compared to state-of-the-art superpixel methods. Pattern Analysis and Machine Intelligence, IEEE Transactions on **34**(11), 2274–2282 (2012)

[2] Alegre, F., Dellaert, F.: A probabilistic approach to the semantic interpretation of building facades. In: CIPA International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres, pp. 25–27 (2004)

[3] Benz, F., Kötzing, T.: An effective heuristic for the smallest grammar problem. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp. 487–494. ACM (2013)

[4] Berg, A.C., Grabler, F., Malik, J.: Parsing images of architectural scenes. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pp. 1–8. IEEE (2007)

[5] Bod, R.: An efficient implementation of a new DOP model. In: 10th Conference on European Chapter of the Association for Computational Linguistics (EACL 2003), Volume 1, pp. 19–26 (2003)

[6] Bod, R.: An all-subtrees approach to unsupervised parsing. In: 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006), pp. 865–872. Association for Computational Linguistics (2006)

[7] Carrasco, R.C., Oncina, J., Calera-Rubio, J.: Stochastic inference of regular tree languages. Machine Learning **44**(1-2), 185–197 (2001)

[8] Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Rasala, A., Sahai, A., et al.: Approximating the smallest grammar: Kolmogorov complexity in natural models. In: Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing (STOC), pp. 792–801. ACM (2002)

[9] Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A., Shelat, A.: The smallest grammar problem. Information Theory, IEEE Transactions on **51**(7), 2554–2576 (2005)

[10] Chi, Y., Muntz, R.R., Nijssen, S., Kok, J.N.: Frequent subtree mining – an overview. Fundamenta Informaticae **66**(1), 161–198 (2005)

[11] Clark, A.: Distributional learning of some context-free languages with a minimally adequate teacher. In: Grammatical Inference: Theoretical Results and Applications, pp. 24–37. Springer (2010)

[12] Cohen, A., Schwing, A.G., Pollefeys, M.: Efficient structured parsing of facades using dynamic programming. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pp. –. IEEE (2014)

[13] Cohen, S.B., Stratos, K., Collins, M., Foster, D.P., Ungar, L.: Spectral learning of latent-variable pcfgs: Algorithms and sample complexity. The Journal of Machine Learning Research **15**(1), 2399–2449 (2014)

[14] Cohen, S.B., Stratos, K., Collins, M., Foster, D.P., Ungar, L.H.: Experiments with spectral learning of latent-variable PCFGs. In: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL 2013), pp. 148–157 (2013)

[15] Cohn, T., Blunsom, P., Goldwater, S.: Inducing tree-substitution grammars. The Journal of Machine Learning Research **11**, 3053–3096 (2010)

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

[17] Dai, D., Prasad, M., Schmitt, G., Van Gool, L.: Learning domain knowledge for façade labelling. In: Computer Vision–ECCV 2012, pp. 710–723. Springer (2012)

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

[19] De La Higuera, C.: A bibliographical study of grammatical inference. Pattern recognition **38**(9), 1332–1348 (2005)

[20] D'Ulizia, A., Ferri, F., Grifoni, P.: A survey of grammatical inference methods for natural language learning. Artificial Intelligence Review **36**(1), 1–27 (2011)

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

[22] Flajolet, P., Sipala, P., Steyaert, J.M.: Analytic variations on the common subexpression problem. In: Proceedings of the 17th International Colloquium on Automata, Languages and Programming, pp. 220–234. Springer (1990)

[23] Frey, B.J., Dueck, D.: Clustering by passing messages between data points. science **315**(5814), 972–976 (2007)

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

[25] Grünwald, P.: A minimum description length approach to grammar inference. In: Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing, pp. 203–216. Springer-Verlag (1996)

[26] De la Higuera, C.: Grammatical inference: learning automata and grammars. Cambridge University Press (2010)

[27] Jampani, V., Gadde, R., Gehler, P.V.: Efficient facade segmentation using auto-context. In: Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on, pp. 1038–1045. IEEE (2015)

[28] Johnson, M., Griffiths, T., Goldwater, S.: Bayesian inference for PCFGs via Markov Chain Monte Carlo. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, pp. 139–146 (2007)

[29] Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International journal of computer vision **1**(4), 321–331 (1988)

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

[31] Komodakis, N., Paragios, N., Tziritas, G.: Clustering via lp-based stabilities. In: Advances in Neural Information Processing Systems 21, pp. 865–872 (2009)

[32] Korc, F., Forstner, W.: eTRIMS Image Database for interpreting images of man-made scenes. Tech. Rep. TR-IGG-P-2009-01, Dept. of Photogrammetry, University of Bonn (2009). URL `http://www.ipb.uni-bonn.de/projects/etrims_db/`

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

[34] Koziński, M., Gadde, R., Zagoruyko Sergeyand Marlet, R., Obozinski, G.: A MRF shape prior for facade parsing with occlusions. In: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on (2015)

[35] Koziński, M., Marlet, R.: Image parsing with graph grammars and markov random fields. In: Winter Conference on Applications of Computer Vision (WACV 2014) (2014)

[36] Koziński, M., Obozinski, G., Marlet, R.: Beyond procedural facade parsing: Bidirectional alignment via linear programming. In: 12th Asian Conference on Computer Vision (ACCV 2014) (2014)

[37] Lehman, E., Shelat, A.: Approximation algorithms for grammar-based compression. In: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 205–212. Society for Industrial and Applied Mathematics (2002)

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

[39] Manning, C.D.: Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In: 12th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2011) - Volume Part I, pp. 171–189. Springer-Verlag (2011)

[40] Martinović, A., Mathias, M., Weissenberg, J., Van Gool, L.: A three-layered approach to facade parsing. In: Computer Vision–ECCV 2012, pp. 416–429. Springer (2012)

[41] Martinovic, A., Van Gool, L.: Bayesian grammar learning for inverse procedural modeling. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 201–208. IEEE (2013)

[42] Martinović, A., Van Gool, L.: Earley parsing for 2D stochastic context free grammars. Tech. Rep. KUL/ESAT/PSI/1301, KU Leuven (2013)

[43] Matsuzaki, T., Miyao, Y., Tsujii, J.: Probabilistic CFG with latent annotations. In: 43rd Annual Meeting on Association for Computational Linguistics (ACL 2005), pp. 75–82 (2005)

[44] Miller, P.: Strong generative capacity. CSLI Publications (1999)

[45] Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. In: ACM SIGGRAPH 2006 / ACM Transactions on Graphics, pp. 614–623 (2006)

[46] Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research pp. 67–82 (1997)

[47] Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., Marsi, E.: Malt parser: A language-independent system for data-driven dependency parsing. Natural Language Engineering **13**(2), 95–135 (2007)

[48] Ok, D., Kozinski, M., Marlet, R., Paragios, N.: High-level bottom-up cues for top-down parsing of facade images. In: 2nd Joint 3DIM/3DPVT Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT) (2012)

[49] Osher, S., Paragios, N.: Geometric level set methods in imaging, vision, and graphics. Springer (2003)

[50] Parisot, S., Duffau, H., Chemouny, S., Paragios, N.: Graph based spatial position mapping of low-grade gliomas. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011, pp. 508–515. Springer (2011)

[51] Parisot, S., Duffau, H., Chemouny, S., Paragios, N.: Graph-based detection, segmentation & characterization of brain tumors. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 988–995. IEEE (2012)

[52] Petrov, S., Klein, D.: Improved inference for unlexicalized parsing. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, pp. 404–411. Association for Computational Linguistics (2007)

[53] Riemenschneider, H., Krispel, U., Thaller, W., Donoser, M., Havemann, S., Fellner, D., Bischof, H.: Irregular lattices for complex shape grammar facade parsing. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 1640–1647. IEEE (2012)

[54] Ripperda, N., Brenner, C.: Reconstruction of façade structures using a formal grammar and RJMCMC. In: Pattern Recognition, pp. 750–759. Springer (2006)

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

[56] Sakakibara, Y., Kondo, M.: GA-based learning of context-free grammars using tabular representations. In: ICML, vol. 99, pp. 354–360 (1999)

[57] Si, Z., Zhu, S.C.: Learning and-or templates for object recognition and detection. IEEE Trans. Pattern Anal. Mach. Intell. **35**(9), 2189–2205 (2013). DOI 10.1109/TPAMI.2013.35. URL `http://dx.doi.org/10.1109/TPAMI.2013.35`

[58] Simon, L., Teboul, O., Koutsourakis, P., Paragios, N.: Random exploration of the procedural space for single-view 3D modeling of buildings. International journal of computer vision **93**(2), 253–271 (2011)

[59] Simon, L., Teboul, O., Koutsourakis, P., Van Gool, L., Paragios, N.: Parameter-free/Pareto-driven procedural 3D reconstruction of buildings from ground-level sequences. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 518–525. IEEE (2012)

[60] Sutton, R.S., Barto, A.G.: Introduction to reinforcement learning. MIT Press (1998)

[61] Teboul, O.: Shape grammar parsing: Application to image-based modeling. Ph.D. thesis, Ecole Centrale Paris (2011)

[62] Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., Paragios, N.: Shape grammar parsing via reinforcement learning. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 2273–2280. IEEE (2011)

[63] Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., Paragios, N.: Parsing facades with shape grammars and reinforcement learning. Pattern Analysis and Machine Intelligence, IEEE Transactions on **35**(7), 1744–1756 (2013)

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

[65] Tomita, M.: Parsing 2-dimensional language. In: M. Tomita (ed.) Current Issues in Parsing Technology, *The Springer International Series in Engineering and Computer Science*, vol. 126, pp. 277–289. Springer US (1991)

[66] Tu, K., Pavlovskaia, M., Zhu, S.C.: Unsupervised structure learning of stochastic and-or grammars. In: Advances in Neural Information Processing Systems, pp. 1322–1330 (2013)

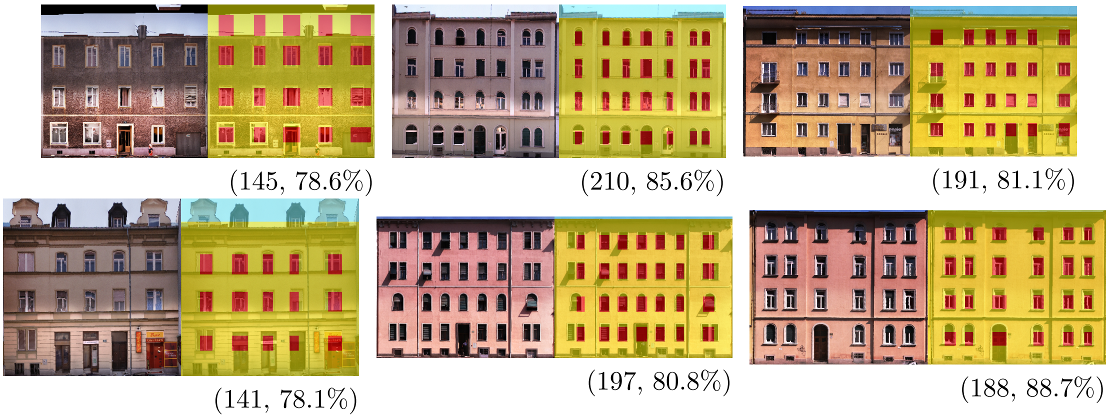
[67] Tylecek, R.: The cmp facade database. Tech. rep., CTU–CMP–2012–24, Czech Technical University (2012)

[68] Valiente, G.: Algorithms on trees and graphs. Springer (2002)

[69] Weissenberg, J., Riemenschneider, H., Prasad, M., Van Gool, L.: Is there a procedural logic to architecture? In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 185–192. IEEE (2013)

[70] Wonka, P., Wimmer, M., Sillion, F., Ribarsky, W.: Instant architecture. ACM Transactions on Graphics (TOG). **22**(3), 669–677 (2003)

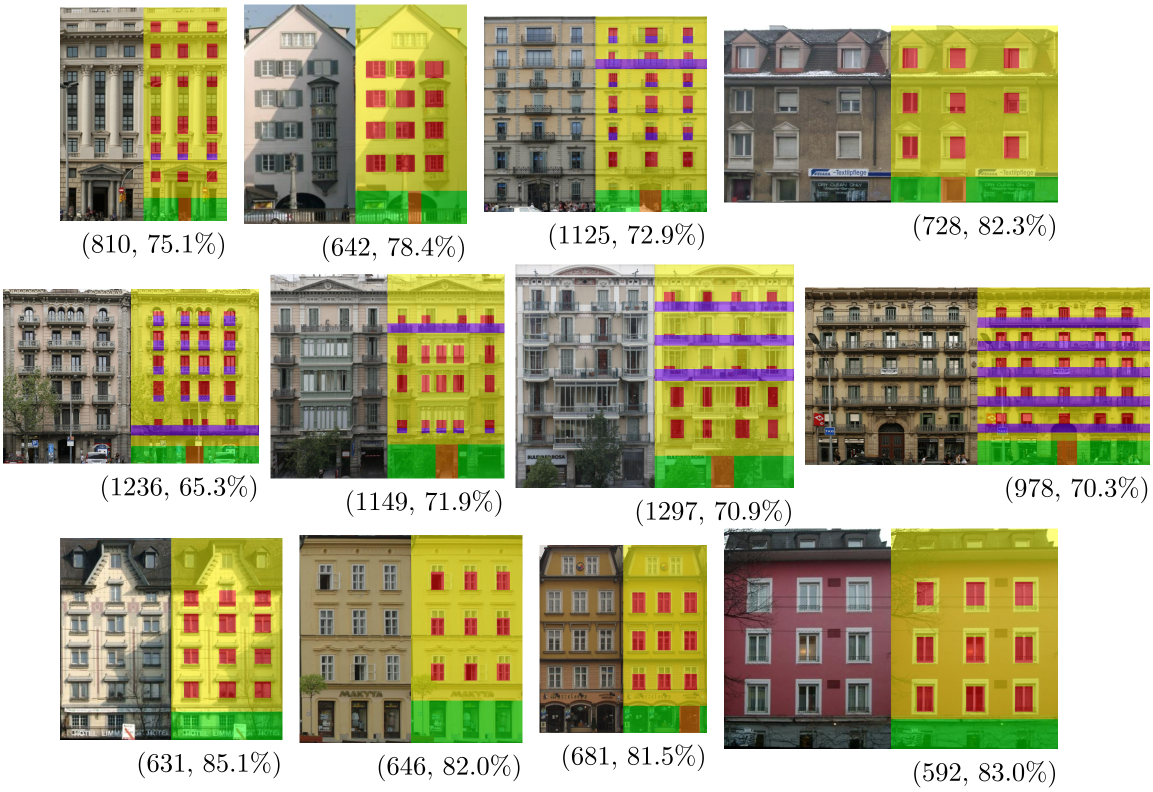
[71] Zaki, M.J.: Efficiently mining frequent trees in a forest. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 71–80. ACM (2002)



(145, 78.6%) (210, 85.6%) (191, 81.1%)

(141, 78.1%) (197, 80.8%) (188, 88.7%)

Figure 12: Qualitative results on Graz2012 dataset. Image (left) and segmentation using learned grammar $\mathcal{G}_{\text{cl}}$ (right) are shown here along with number of episodes for convergence and segmentation accuracy.

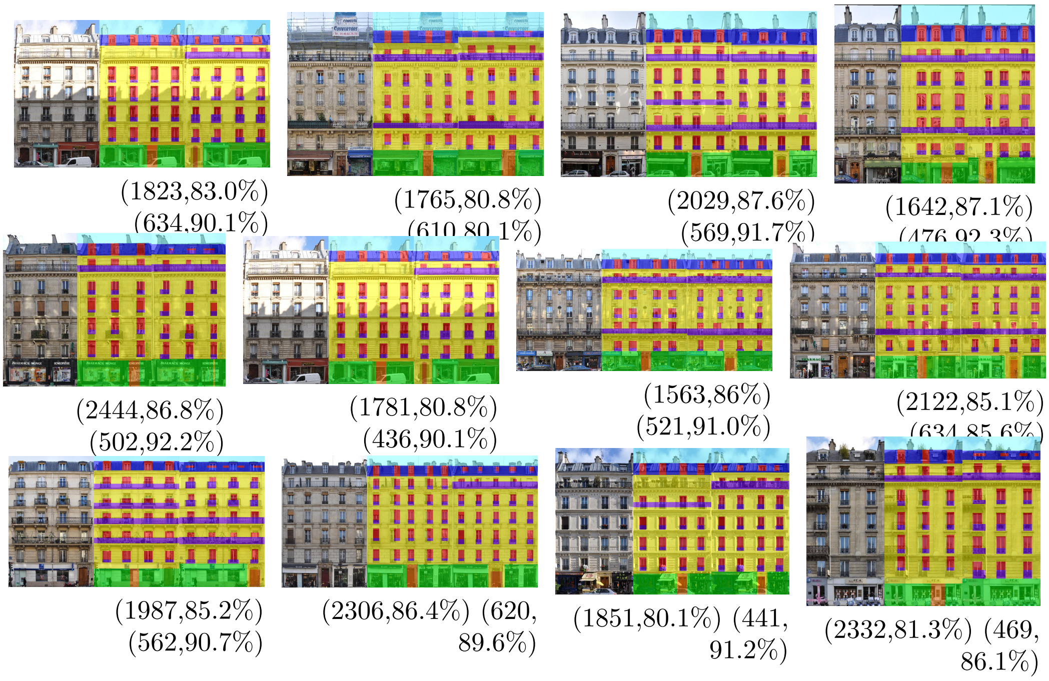


(810, 75.1%) (642, 78.4%) (1125, 72.9%) (728, 82.3%)

(1236, 65.3%) (1149, 71.9%) (1297, 70.9%) (978, 70.3%)

(631, 85.1%) (646, 82.0%) (681, 81.5%) (592, 83.0%)

Figure 13: Qualitative results on CMP2013 dataset. Image (left) and segmentation using learned grammar $\mathcal{G}_{\text{cl}}$ (right) are shown here along with number of episodes for convergence and segmentation accuracy.



(1823,83.0%) (634,90.1%)

(1765,80.8%) (610,80.1%)

(2029,87.6%) (569,91.7%)

(1642,87.1%) (476,92.3%)

(2444,86.8%) (502,92.2%)

(1781,80.8%) (436,90.1%)

(1563,86%) (521,91.0%)

(2122,85.1%) (634,85.6%)

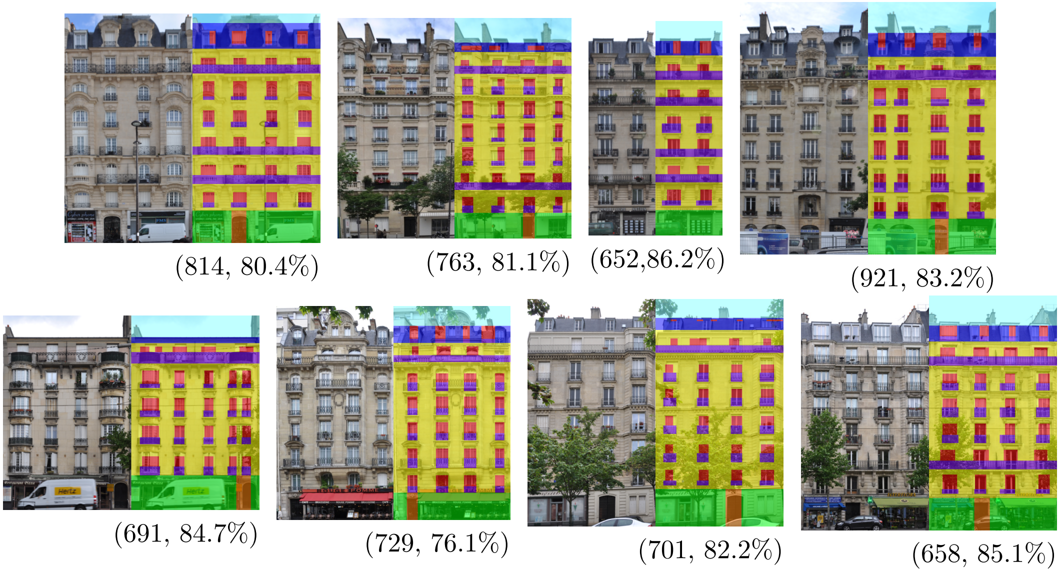
(1987,85.2%) (562,90.7%)

(2306,86.4%) (620, 89.6%)

(1851,80.1%) (441, 91.2%)

(2332,81.3%) (469, 86.1%)

Figure 14: Qualitative results on ECP2011 dataset. Image (left) and segmentation using handwritten grammar (center) and learned grammar $\mathcal{G}_{\text{cl}}$ (right) are shown here along with number of episodes for convergence and segmentation accuracy.



(814, 80.4%)

(763, 81.1%)

(652,86.2%)

(921, 83.2%)

(691, 84.7%)

(729, 76.1%)

(701, 82.2%)

(658, 85.1%)

Figure 15: Qualitative results on ENPC2014 dataset. Image (left) and segmentation using learned grammar $\mathcal{G}_{\text{cl}}$ (right) are shown here along with number of episodes for convergence and segmentation 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