Shape Grammar Parsing via Reinforcement Learning

Olivier Teboul[1,2] Iasonas Kokkinos[1,3] Loïc Simon[1] Panagiotis Koutsourakis[1] Nikos Paragios[1,3]
[1] Laboratoire MAS, Ecole Centrale Paris, [2] Microsoft France, [3] INRIA Saclay, GALEN Group

Abstract

We address shape grammar parsing for facade segmentation using Reinforcement Learning (RL). Shape parsing entails simultaneously optimizing the geometry and the topology (e.g. number of floors) of the facade, so as to optimize the fit of the predicted shape with the responses of pixel-level 'terminal detectors'. We formulate this problem in terms of a Hierarchical Markov Decision Process, by employing a recursive binary split grammar. This allows us to use RL to efficiently find the optimal parse of a given facade in terms of our shape grammar. Building on the RL paradigm, we exploit state aggregation to speedup computation, and introduce image-driven exploration in RL to accelerate convergence. We achieve state-of-the-art results on facade parsing, with a significant speed-up compared to existing methods, and substantial robustness to initial conditions. We demonstrate that the method can also be applied to interactive segmentation, and to a broad variety of architectural styles [1].

1. Introduction

Facade parsing has received increasing attention over the past few years, [1, 2, 3, 4, 5, 6], and can be exploited for urban image databases such as Google Street View or Microsoft Bing Maps; semantically labeling such images could facilitate for instance realistic 3D content creation, or reconstruction. The highly-structured nature of building facades makes their treatment amenable to model-based approaches and in particular to grammatical representations, as they can naturally capture the hierarchical structure of facades. For instance, as opposed to flat CRF techniques, e.g. [7], hierarchical models can enforce global constraints, e.g. straight boundaries, or the nesting of structures.

Apart from solving the goal of facade segmentation in itself, grammar-based approaches hold promise for addressing some of the main challenges of high-level vision: *scalability* can be addressed by reusing parts among different objects [8], while *versatility* requires dealing with structure variation [9]. Grammars naturally address these challenges in Natural Language Processing (NLP) [10], and one can argue that this may also be the case for vision.

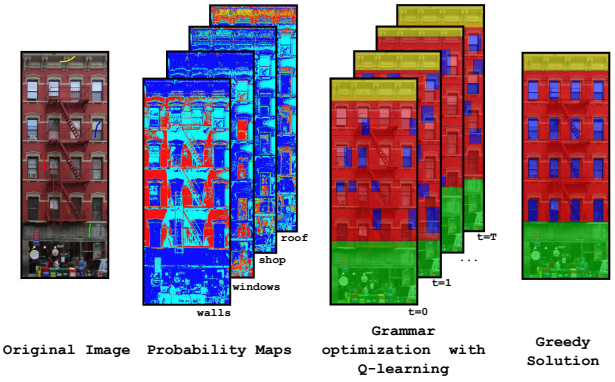


Figure 1. Overview of the method: a discriminative model provides probability maps for all terminal symbols of the shape grammar. An agent then repeatedly segments the image using the grammar by following a shape parsing policy that is learned on-line with RL. The policy at convergence yields the final segmentation.

In computer vision, after the early works of [11, 12], grammars were reintroduced in the last decade for segmentation and detection tasks [1, 8, 13, 14], as well as medical image analysis [15] and 3d reconstruction [16]. One of the main bottlenecks in adapting NLP-based techniques to vision is that when breaking a visual structure into parts the position of each part is a continuous 2D variable. Therefore, adapting discrete Dynamic Programming-type algorithms for parsing (e.g. CYK [10]) can be problematic, as discretization may lead to a huge number of states. To address this, sparse image representations were used in [8, 13], Data-Driven MCMC techniques in [14], while [17] used a gradual refinement of an initally limited set of label map templates. Substantial progress has also been made in speeding up DP-based inference for continuous variables [18, 19, 20], considering however a fixed model topology.

Instead, we pursue an approach based on Reinforcement Learning (RL) [21] to address the parsing problem. RL comprises a set of techniques to approximately, but efficiently solve problems that are phrased as Markov Decision Processes (MDPs), including Dynamic Programming (DP) and Monte-Carlo as extreme cases. We argue that RL

[1]The work has been supported by Microsoft Research through its PhD Scholarship program

provides us with novel tools, that are more appropriate for the shape parsing problem, while also being more flexible than DP. In specific, we introduce RL techniques such as Hierarchical RL, and state aggregation to harness the computational complexity of the shape parsing problem. We also show that RL allows us to seamlessly exploit image-based information during optimization, without necessitating a sparse representation of the image. We thus exploit image-based guidance (contrary to DP), while at the same time being immune to the front-end failures of sparse image representations (unlike e.g. [13, 8]).

We develop our ideas around shape grammars (SG's) [22, 23, 3], which were used in [4] for building interpretation using primarily image-driven methods and in [6] using Monte-Carlo which can be seen as a special case of RL. We demonstrate how RL can result in 50fold speedups over [6], while giving competitive labelling accuracies.

The paper is organized as follows: In Sec. 2 we detail how we use shape grammars for facade synthesis. In Sec. 3 we briefly introduce the main concepts in Reinforcement Learning and describe how solving the shape parsing problem can be formulated in RL terms. The efficient optimization of facade grammars using RL is discussed in Sec. 4, while we provide experimental results in Sec. 5.

2. Shape Synthesis using Shape Grammars

We start by describing 2D shape grammars, then focus on a special case, split grammars, and then give an example of their use as generative models for facades.

2D Shape grammars: A 2D shape grammar (SG) [22] describes a shape configuration in terms of a dictionary of basic shapes. Each basic shape consists of a template, a symbol for its *type* (e.g. 'd' for door, 'w' for wall), and its bounding box parameters; we describe by (c, x, y, w, h, θ) a shape of type c at position (x, y) with width/height dimensions (w, h) and orientation θ.

Complementing the dictionary is a set of replacement rules. Each rule replaces a basic shape with other basic shapes; for instance, one such rule may split a floor shape into a set of alternating window and wall shapes. All rules are of the context-free form: $P \rightarrow C_1, \ldots, C_k$, where P is the precedent and C_i are the antecedents.

A *terminal* is a shape that does not appear on the left-hand side of any rule, i.e. cannot be processed further. A special shape, the *axiom* appears at the beginning of any derivation. A shape grammar starts from the axiom and iteratively breaks it into simpler basic shapes until all shapes become terminals. A tree that is rooted at the axiom, has terminals at all leaves, and has as intermediate nodes basic shapes generated from their ancestors by replacement rules is called a 'derivation tree'.

The major difference with context-free grammars used for NLP is that the replacement rules involve structures with continuous attributes. For each term involved in a replacement rule, $d = 5$ continuous variables are involved. Considering that each variable is quantized into m values, there are $O((m^d)^k)$ possible versions of a replacement rule with k antecedents. Moreover, k is unknown: for instance we do not know how many floors there are in a facade.

To break this $O((m^d)^k)$ complexity into pieces we first introduce split grammars, to deal with d , and then rewrite shape grammars in binary form to deal with k.

Split Grammars: A split grammar [23] is a shape grammar with rules that split shapes along one dimension at a time. The dimensions of the children along the split direction are determined by the rule, while their dimensions along all other axes are inherited from their parent. A split rule is thus defined by an axis and $k-1$ split parameters where k is the number of children. This reduces the number of rules from $O((m^d)^k)$ to $O(dm^k)$, since instead of splitting over all the d dimensions, we split over a single one. This makes the grammar more rigid, but is natural for buildings as they are mostly organized in orthogonal frames.

Binary Split Grammars: Any context-free grammar can be reduced to Chomsky Normal Form (CNF) [24], where each rule has at most two antecedents. To apply this to shape grammars, consider breaking up a facade into floors and walls: instead of a single split rule that breaks the facade in a single shot into multiple floors and walls, we use several simple rules that break up the facade into a floor/wall and the facade's remainder. Such a rule would be:

$$A(fa, X, Y, W, H) \underset{H:h,fl}{\longrightarrow} B(fl, X, Y, W, h)C(fa, X, Y+h, W, H-h),$$

meaning that we take a shape A of type fa(cade), and split it along the H(eight) dimension into a fl(oor) of height h and a remainder of type fa(cade). This results in a shape B of type floor, of height h, and the remainder C of the floor; the union of the supports of B and C gives the support of A.

On the one hand, writing the grammar in binary form increases from 1 to k the rules required to decompose a k-partite structure into its constituents. As the complexity of each rule goes from $O(m^k)$ to $O(m)$, the overall complexity stays $O(m^k)$. But now we can deal with an unknown value of k; during parsing, computation can be shared to parse a shape with different values of k. Moreover, the binary form allows us to phrase our task as an MDP, as described in Sec. 3.

Facade Modelling with Shape Grammars: We demonstrate in Fig. 2 a binary split grammar for facades. We denote in color the grammar terminals, i.e. walls and win-

Parent	Children		Split
Axiom(W,H)	Facade(0,0,W,H)		None
Facade(0,Y,W,H)	Floor(0,Y,W,h)	Fa-Wall(0,Y+h,W,H-h)	Y:h
Fa-Wall(0,Y,W,H)	Wall(0,Y,W,h)	Facade(0,Y+h,W,H-h)	Y:h
Floor(X,Y,W,H)	Wall(X,Y,w,H)	Fl-Win(X+w,Y,W-w,H)	X:w
Fl-Win(X,Y,W,H)	Window(X,Y,w,H)	Floor(X,Y,W-w,H)	X:w

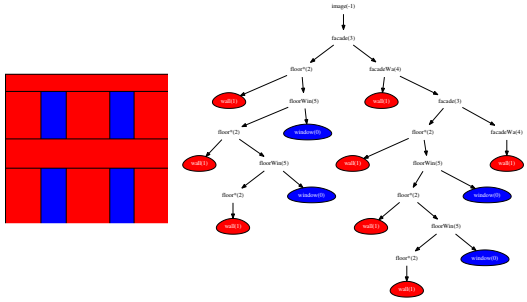


Figure 2. A toy 2D split grammar (top) and a shape generated by it (bottom). Walls and windows are terminals; filling their domain with the appropriate color gives the facade on the right.

dows. We omit θ as we consider that facade images have been ortho-frontally rectified, while the splits are either along the X or Y axis; so θ will always be zero. Finally, we incorporate the shape's 'type' in its name.

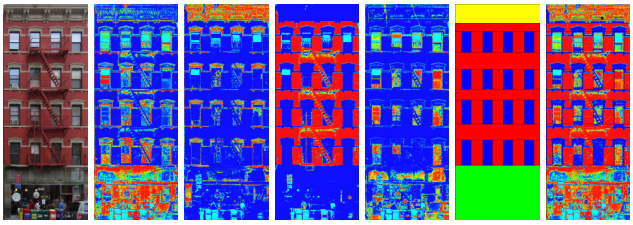
According to the grammar the facade is first split vertically into an alternating sequence of floors and walls (rules 2 & 3); each floor is then split into a sequence of walls and windows (rules 4 & 5). Note that the grammar forces the horizontal alternation of 'wall' terminals and 'floor' shapes and the vertical alternation of 'wall' terminals and 'window' terminals. Moreover the splits alternate directions at each layer; facades are split vertically and floors horizontally.

There are some notable properties of the grammar: first, the recursive form of the grammar allows us to consider any number of elements; this is determined by the size of the facade, and the parameters of the splits. This is well suited to deal with structure variation. Second, the same terminal ('wall') is used at two different layers of the hierarchy, both to separate floors within a facade and to separate windows within a floor. This is an instance of 'reusing' parts in a grammar [9, 8] made here specific to shape grammars.

At the bottom of Fig. 2 we show a shape derived by using some of the rules given at the top of the figure; windows are shown as blue and walls as red. We can thus use our grammar as a generative model for a variety of building facades by using different parameters in its rules. We now turn to the inverse problem of parsing a given shape with our grammar, which amounts to finding the sequence of rules to apply on an axiom in order to optimally fit an input image.

3. Shape Parsing via Reinforcement Learning

We now turn from synthesis with shape grammars to the inverse problem of image interpretation. For a facade this amounts to finding a sequence of rules that optimally break a given image into windows, doors, walls, etc. This is similar to semantic segmentation/parsing [7, 17] in that we assign a class label to each pixel. The difference is that in our case the labeling is the result of a shape grammar derivation; we thus refer to our labeling problem as *shape parsing*.



(a) Input (b) m(shop) (c) m(roof) (d) m(wall) (e) m(win) (f) c (g) m(c)

Figure 3. Illustration of the objective function: (b)-(e) are the pixel-wise merit functions for each terminal class (shop/roof/wall/window), indicated by both color and text. (f) is a parse of the image, indicating the labels in terms of terminal color. (g) combination of the merits according to the parse in (f).

This task has two sides: first, we consider that we have an image-driven, *bottom-up* merit function $m(x, y, c) \in [0, 1]$ that indicates whether pixel x, y is of class c; we consider different variants of m in Sec. 5. For instance, as shown in Fig. 3(b)-(e) we can use discriminatively trained classifiers to score the posterior probability of different terminal classes at the individual pixel level.

Second, the grammar injects *top-down* information in the solution, for instance all terminals are axis-aligned rectangles, and all windows on a floor have the same height, as shown in Fig. 3(f). This constrains the set of possible labellings in a more problem-specific manner than what could be attained by a flat, MRF-based semantic segmentation approach.

Our goal is to come up with a grammar-based segmentation of the shape that optimally fits to the bottom-up information of m, i.e. maximizes the quantity $\sum_{x,y} m(x, y, c(x, y))$, where $c(x, y)$ is the terminal-level labelling induced by the grammar segmentation. Defining the merit of a terminal $A(c_A, x, y, w, h)$ of type c_A as:

$$M(A(c_A, x, y, w, h)) = \sum_{x'=x}^{x+w} \sum_{y'=y}^{y+h} m(x', y', c_A) \qquad (1)$$

we can recursively express the merit of a non-terminal shape as the sum of its descendants' merits. Pushing the recursion to its end, we see that our task is to pick a set of grammar rules that maximizes the merit of the 'axiom' shape, by providing a combination of terminals with maximal cumulative merit.

This is challenging for two reasons: first, a structure is not split directly into terminals; instead, for example, a 'floor' part is split into a terminal 'window' and a non-terminal, 'remaining' part. We can directly evaluate the

window's merit (as it is terminal) but for the remainder we need to wait until it is further split into terminals. This problem amounts to maximizing *cumulative* reward (also called *return*), as opposed to *immediate* reward.

Second, certain non-terminals are split exclusively into non-terminals, e.g. a facade is split into floors; these need to be further split into walls and windows before deciding about the merit of the facade's split. This reflects the grammar's hierarchy. This problem amounts to performing optimization in a hierarchical setting.

These two problem aspects, namely optimizing cumulative rewards and dealing with hierarchically defined reward functions are common in Reinforcement Learning, where an agent learns to optimize its behavior for a succession of tasks, which may involve multiple subtasks, before leading to concrete rewards. We will now make this connection more concrete by introducing some necessary concepts from Reinforcement Learning.

3.1. Reinforcement Learning Formulation

We will establish a connection between our problem and Reinforcement Learning by introducing some basic terminology of RL, and directing the reader to [21] for an excellent introduction to the topic. We start by describing the parsing problem in terms of an agent taking actions and obtaining rewards. We then identify the Markov Decision Process describing the interaction of the agent with its environment, and describe how RL can identify an optimal agent policy. We then simplify the problem of determining the optimal policy by exploiting the problem's hierarchy.

Agent, state & actions: Our 'agent' is a parsing engine; its state $s = (\mathcal{T}, \mathcal{N})$ is a derivation tree $\mathcal{T}$ together with a pointer to the non-terminal node $\mathcal{N}$ that is currently processed. The agent can use as action a at s any grammar rule applicable to $\mathcal{N}$. If the rule produces a terminal, the agent's *immediate reward* equals the terminal's merit. Otherwise it is zero. After applying an action a the agent moves to a new state $s' = (\mathcal{T}', \mathcal{N}')$; $\mathcal{T}'$ is obtained from $\mathcal{T}$ by appending to node $\mathcal{N}$ the children generated by applying action a. If one or more non-terminals are produced by the rule, $\mathcal{N}'$ moves to the leftmost non-terminal. Otherwise, $\mathcal{N}'$ becomes the first unprocessed non-terminal encountered while backtracking in the tree. Our agent's initial state has as $\mathcal{T}$ the grammar's axiom. The agent stops when there are no non-terminals left.

Our agent essentially performs a depth-first derivation of an image interpretation tree; its goal is to maximize the reward accumulated by its actions. This amounts to finding the derivation tree with maximal cumulative merit at its terminals, i.e. solving the parsing problem outlined in the previous Section.

MDP formulation: We can formulate our agent's interaction with the environment in terms of a Markov Decision Process (MDP): the next state s' depends entirely on the current state s and the chosen action a. The immediate reward is a function $R(s, a)$ of the current state and action. The agent's actions are determined by a *policy* function $\pi(s, a) = p(a|s)$, giving the probability of picking action a at state s. The expected reward $V^\pi(s)$ of an agent starting at state s and then following policy π can be computed with Bellman recursion:

$$V^\pi(s) = \sum_a \pi(s, a) \left[R(s, a) + V^\pi(s')\right] \tag{2}$$

In words, we first compute the *immediate* reward $R(s, a)$ obtained by performing action a at state s. Action a leads us (deterministically in our case) to a new state s'; by following policy π after s', the agent will receive an expected reward $V^\pi(s')$. The agent's cumulative reward for playing a at s is thus the sum of its immediate, $R(s, a)$, and expected-to-come $V^\pi(s')$ rewards. $V^\pi(s)$ is obtained from the expectation under policy $\pi(a, s)$ of this cumulative reward $R(s, a) + V^\pi(s')$.

We can alternate between the *value function* $V^\pi(s)$ and the *action-value function* $Q^\pi(s, a)$ as follows:

$$Q^\pi(s, a) = R(s, a) + V^\pi(s(a)) \tag{3}$$

$$V^\pi(s) = \sum_a \pi(s, a) Q^\pi(s, a) \tag{4}$$

$Q^\pi(s, a)$ is intuitively similar to $V^\pi(s)$, with the difference that we identify a, instead of taking a summation over it.

The agent's goal is to maximize the (expected) cumulative reward obtained from its actions. This amounts to finding the policy π with maximal expected cumulative reward.

Optimal Policies via RL: Reinforcement learning includes a range of techniques allowing us to find a policy $\pi^*(a, s)$ with maximal expected cumulative reward (see e.g. [21] for details). For instance Q-learning alternates between using the current estimate of $Q(s, a)$ to determine π and then following π to refine $Q(s, a)$. For the first part (from Q to π), at each state s, with probability $1 - \epsilon$, the agent chooses $a^* = \text{argmax}_a Q(s, a)$. Otherwise, it picks an action at random, thereby exploring the state space. The so-called ϵ-greedy exploration policy is a way to tackle the *exploration-explotation* trade-off:

$$\pi(s, a) = (1 - \epsilon)\delta(a, a^*) + \epsilon U(a), \tag{5}$$

where U is a uniform distribution and δ is the Kronecker function.

For the second part (update of Q) the agent picks the most promising action a' at the state s' resulting from its previously chosen action, a. $Q(s, a)$ is then updated as:

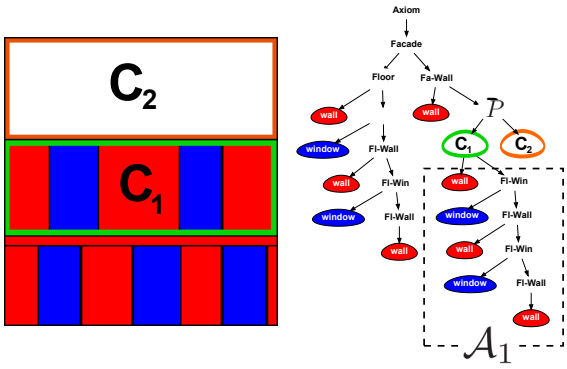


Figure 4. State simplification for a shape parse tree being processed in a DFS order: The agent in P needs to first fully derive C_1 before moving into the derivation of C_2. For the derivation of C_1 our agent can find the optimal sequence of state-action pairs, $\mathcal{A}_\infty$ by relying on a *local policy* that ignores the part of the derivation tree lying above P. This local policy can rely on only a part of the information contained in the state of the agent at the current step.

$$\Delta Q(s,a) = \alpha[R(s,a) + \max_{a'} Q(s',a') - Q(s,a)] \quad (6)$$

where α is a learning rate. This can be interpreted as bringing closer a previous estimate of $Q(s,a)$ with the more up-to-date estimate, $R(s,a) + \max_{a'} Q(s',a')$. Repeatedly applying Eq. 6 for several runs while slowly decreasing ϵ and α results in learning an optimal policy. Once π^* is found, choosing at each step $a^*(s) = \text{argmax}_a \pi^*(s,a)$ will yield the optimal solution for an MDP.

Hierarchical Decomposition: Given an optimal policy we could start from the 'axiom' state and then optimally parse the image by greedily choosing the best action at each state. But in practice we cannot learn this policy using RL with the state representation described so far, since the number of states is huge -it includes all possible parse trees for an image. However, as we will now describe, we can simplify the expression of $Q(s,a)$ based on ideas from Hierarchical Reinforcement Learning [25].

As noted in Sec. 3, our agent performs a depth-first derivation. Consider, as shown in Fig. 4, that we have just expanded a node P into two non-terminals C_1, C_2, and the node to be currently processed is $\mathcal{N} = C_1$. Our agent will first fully derive C_1, and then C_2. Consider that it starts expanding C_1 at time 1 and finishes it at time j. It then goes on expanding C_2, and backtracks when finished with C_2. To simplify $Q(s,a)$ we break the sequence $\mathcal{A} = \{(s_1,a_1)\dots(s_k,a_k)\}$ of state-action pairs starting from $s_1 = (\mathcal{T},\mathcal{N})$ into two subsequences $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_1 = \{(s_1,a_1),\dots,(s_j,a_j)\}$ corresponds to the state-action pairs involved in expanding the offspring of $\mathcal{N}$. $\mathcal{A}_2 = \{(s_{j+1},a_{j+1}),\dots,(s_k,a_k)\}$ deals with C_2 and the rest of the derivation tree, until the agent ends.

Consider a local policy that maximizes the expected cumulative reward during $\mathcal{A}_1$, instead of the whole sequence $\mathcal{A}$. Rewards in $\mathcal{A}_1$ are only affected by $\mathcal{N}$, the currently expanded node instead of the whole tree $\mathcal{T}$; so we only need $\mathcal{N}$'s attributes (4 continuous variables) to learn this policy, plus the shape's type (window/wall/floor, etc). Moreover, what happens in $\mathcal{A}_1$ cannot affect the rewards $\sum_{i=(j+1)}^{k} R(s_i,a_i)$ gathered during $\mathcal{A}_2$. Therefore this local policy will also be globally optimal.

Based on this approach, finding the optimal subtree below $\mathcal{N}$ can be performed with a 'macro' action [25]; once it is finished, the agent labels $\mathcal{N}$ 'completed', keeps the reward $\sum_{(s',a')\in\mathcal{A}_1} R(s,a)$ and moves to s_{j+1}. Within this macro, a local Q function can thus summarize s in terms of the attributes of $\mathcal{N}$. This Q function corresponds to the expected reward accumulated during the macro.

Regarding the action argument, a, of $Q(s,a)$, note that at each step the agent decides where to split a shape along a fixed axis. We can thus represent the local Q function within a macro in terms of $\mathcal{N}$'s attributes for s and a scalar for a; this concludes our simplification for $Q(s,a)$ so far.

Note that the children resulting from a split may be non-terminals themselves. But we can use nested macros with local policies per child. The reward accumulated during the derivation of a child provides the reward $R(s,a)$ resulting from the split of the parent shape by a. These rewards can be used to learn the local $Q(s,a)$ functions with Q-learning.

4. Accelerated Parsing using Reinforcement Learning Techniques

The hierarchical decomposition presented above dramatically simplifies s,but we still need to perform RL for a 5-D action-value function. We now describe two techniques that allow us to speedup computation in ways that would be impossible with plain DP. We first describe state aggregation, a RL technique to deal with a large number of states. Then, we present an image-driven technique that directs our agent's exploration towards promising areas.

State Aggregation: State aggregation [26] reduces the number of state-action pairs over which $Q(s,a)$ is estimated by coalescing sets of states into single states. To motivate our approach, consider a 'floor' shape. Intuitively, the height of a floor should not affect how we split it into windows and walls, since these are placed on the same horizontal locations for all floors. The policy for splitting a floor could thus be expressed by two functions $\pi_{win/wall}(x,a)$ quantifying how good it is to put a window/wall of width a at location x; so we go from 5D policy

functions $\pi_{win/wall}(s,a), s=(x,y,w,h)$ to 2D functions, $\pi_{win/wall}(x,a)$. Similarly, the policy to split a facade into floors and walls should depend only on the height y.

A problem with this emerges because the policy $\pi(s,a)$ is obtained from $Q(s,a)$, as explained in Sec. 3. In turn $Q(s,a)$ is learned by following the policy and observing the rewards $R(s,a)$ obtained over different runs. But $R(s,a)$ is by definition a function of (x,y,w,h), and not only of the variable over which the split is performed. For instance, consider a policy for splitting a floor into walls and windows. Expressing $Q(s,a)$ in terms of only x,a ignores the y coordinate at which the floor lies, or its height h. If the agent has misplaced the floor or under-estimated its height, the reward $R(s,a)$ obtained by putting a window of length a at x will be affected. The rewards obtained from different runs of the same algorithm for the same combinations of x,a will thus be different.

However, RL algorithms can accommodate stochastic rewards; if the reward $R(s,a)$ for a state-action pair changes over different runs, Q-learning will converge to the correct solution under the *expected reward*, $\bar{R}(s,a)$. We can thus learn a Q function per shape that only involves the split variable, say, x by treating the changes in the rewards to the different values of y,w,h as random. Actually, the policy currently used by the agent induces a distribution $P(y,w,h)$ over the other variables y,w,h. The expected reward is:

$$\bar{R}(s,a) = \sum_{y,w,h} R(x,y,w,h,a)P(y,w,h) \qquad (7)$$

Consider for instance a floor macro. The local policy $Q_{Win}(x,a)$ expresses the reward we expect to gather until the completion of a floor when starting from x, and placing a window of length a. The expectation considers all placements of the floor within the facade, while the probability of each placement is determined by the facade policy.

Even though a formal demonstration of the convergence of the method is still missing, we can at least intuitively justify that this scheme converges to a correct policy as follows: consider that the high-level policy for facade splitting starts placing floors more often at proper locations. This will result in similar rewards over runs of the 'floor' macro, and eventually lead also the local policy for floor splitting closer to the optimal one. Similarly, if the local policy becomes correct, misplaced floors will get low rewards, contrary to properly placed ones, which will be properly split. This leads in turn to an improvement of the high-level policy. There is thus a mutual, EM-like, reinforcement of the correct high- and low- level policies. As in all EM-based schemes local minima can occur, but in practice all solutions we obtain look reasonably good.

The robustness of the Q-learning algorithm to initial conditions can be shown empirically. For that matter, we run 200 times our method on the same image with identical parameters. For each experiment, we run Q-learning for 5000 episodes. We record the returns after each episode and average them all among the 200 experiments. Fig. 5 shows the increasing learning curve of the agent, as well as the decreasing evolution of the standard deviation over the experiments.

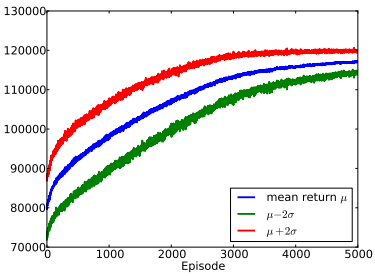


Figure 5. Evolution of the cumulative agent return as a function of episode, averaged over 200 experiments. The mean is increasing and the standard deviation decreases in larger episodes.

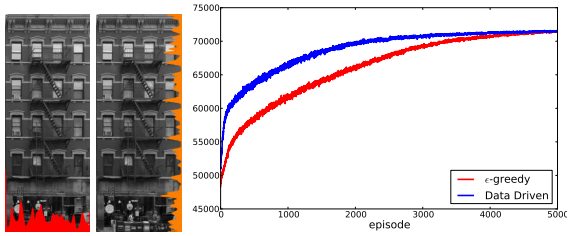


Figure 6. Image-based guidance accelerates Q-learning: the gradient of the image is used to guide the stochastic exploration of actions. This results in a great speedup of the Q-learning algorithm, as we can see in the average learning curves (average returns at each episode over 200 experiments.)

Image-Driven Exploration: As mentioned in Sec. 3 at each step of RL with probability ϵ our agent will choose an action at random. This stochastic exploration allows it to eventually optimize its policy globally, by getting it 'unstuck' from a single policy.

Our modification is that instead of having a uniform distribution for these actions, we use image-based cues to suggest good actions -this can be seen as an adaptation of the DDMCMC scheme of [14] to Q-learning. For instance when our agent is splitting a floor and is located at x, taking an action a will lead it to place a boundary at location $x+a$. We have good reason to believe that this boundary should be close to locations with strong horizontal gradients. To avoid locally noisy measurements due to occlusion, noise etc., we average the horizontal gradient magnitude function along the y axis, and form a $1D$ function, $h(x) = E_y$ to indicate proper actions, as proposed by [27]. The proposal distribution for horizontal split actions is then a softmax function: $P(a;x) = e^{h(x+a)} / \sum_{a'} e^{h(x+a')}$ while the policy

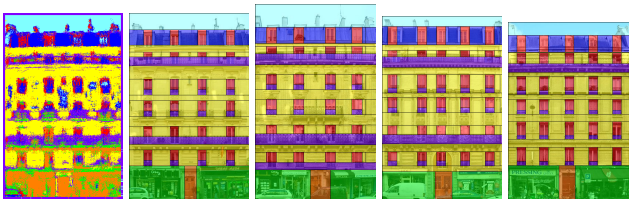


Figure 7. Output of our RL-based parsing method on the Ecole Centrale Paris dataset. We reach state-of-the art results with a more flexible grammar than the one proposed in [6], while cutting down the computation time by more than an order of magnitude.

becomes $\pi(s,a) = (1-\epsilon)d_{a^*}(a) + \epsilon P(a;x)$. A similar proposal distribution is used for the facade's splits into floors and walls, but using the vertical gradients this time.

The proposal distribution favors actions that place the boundaries close to strong gradients. Since it is non-zero for all a's, we can still formally guarantee that our agent will visit all states infinitely often; but it will check promising ones more frequently, thereby exploiting the image guidance. Empirically, as we can see on Fig. 6 we observe that the image-driven strategy results a significant speed-up, in particular in the first phase, where with similar energy levels are reached in one or two orders of magnitude less steps than with plain ϵ-greedy search.

5. Experimental Validation

For quantitative validation we use the experimental setup of [6]. We train a random forest based on the dataset provided by the benchmark. The class posteriors returned by this multi-class classifier provide us with the pixel-level m functions described in Sec. 3. In the leftmost image of Fig. 7 we show a MAP labeling of an image from the test set; obtained by setting the most probable class for each pixel according to the RF-based merit$l(x,y) = \text{argmax}_c m(x,y,c)$. The other images are semantic segmentations obtained by the proposed algorithm.

Both our new algorithm and our old one [6] aim at partitioning the domain of a building's facade in a manner that respects this bottom-up information. Our older inference algorithm used Monte Carlo simulation of the grammar and took approximately 10 minutes per image. In our new work, the RL parsing algorithm requires typically 30 seconds per image to converge on an Intel Xeon CPU W3530 2.8GHz. For reference, a Dynamic Programming-based solution to the same problem took almost 2 days for a single image on the same machine. Note that even though in our exposition so far we have been using only two layers for simplicity, our grammar naturally accommodates more layers, which we use to obtain our results.

Quantitatively, we achieve comparable results with the state-of-the-art shown in [6] (see table 1). Overall, the confusion matrix is slightly better, expect for balconies due to the higher complexity of our new grammar (we allow a balcony to appear at any floor). Not only do we achieve state-of-the-art quantitative results, but we also achieve a speed-up by an order of 50 in terms of running time, and 300 in terms of generated buildings (around 3000 in our method, against 10^6 in [6]). To stress the benefits of our method, we tested it on different types of architectures, using a very flexible grammar with 4 terminal classes. In Fig. 8, we demonstrate segmentation results based another merit function inspired from Grab-cut [28]. The user manually selects on the image some examples of terminal elements and we fit Gaussian Mixture Model (GMM) made of 3 Gaussian kernels for each class. The GMM posteriors constitute the merit function.

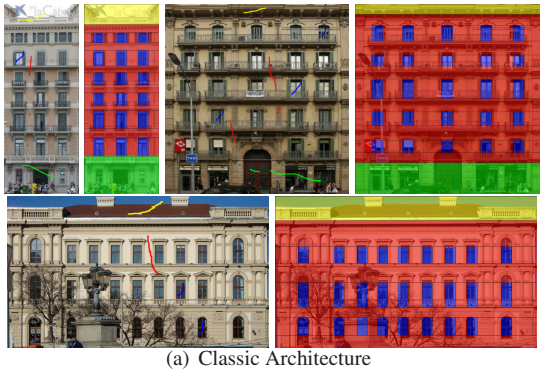
$$\begin{pmatrix} \mathbf{97} & 2 & 1 & 0 & 0 & 0 & 0 \\ 14 & \mathbf{84} & 2 & 0 & 0 & 0 & 0 \\ 2 & 1 & \mathbf{84} & 0 & 1 & 5 & 7 \\ 0 & 0 & 0 & \mathbf{94} & 4 & 2 & 0 \\ 0 & 0 & 4 & 0 & \mathbf{86} & 10 & 0 \\ 0 & 0 & 11 & 0 & 5 & \mathbf{81} & 3 \\ 1 & 0 & 26 & 0 & 0 & 10 & \mathbf{63} \end{pmatrix} \begin{matrix} +2 \\ +13 \\ +1 \\ +0 \\ +6 \\ +0 \\ -9 \end{matrix} \begin{matrix} shop \\ door \\ wall \\ sky \\ roof \\ window \\ balcony \end{matrix}$$

Table 1. Confusion matrix obtained on the benchmark of [6] with the new RL based parsing algorithm. Element i,j represents the percentage of pixels labeled i by the method and j in the ground truth. We compare the diagonal with [6], giving the difference between both methods, in green when the RL method outperforms the old one, and in red otherwise.

Further, thanks to the use of shape grammars, the proposed method is robust to occlusions, noise and illumination conditions (see Fig. 9). Beside, it is flexible and can be adapted to modern architectures, as shown in Fig. 8(b)(right).

6. Conclusion and Future Work

We have introduced Reinforcement Learning (RL) as an efficient method for parsing shape grammars. We demonstrated that RL provides us with tools appropriated for the continuous setting of vision, such as Q-learning and state aggregation, thereby allowing us to control the computational complexity of the problem, while also exploiting image-based guidance, in a seamless manner. We achieved state-of-the-art performance on a challenging benchmark, and showed the potential of the method to deal with a wide variety of buildings. We do believe that this first application of RL to facade parsing shows interesting perspectives in image parsing. In future work we intend to integrate other RL capabilities such as function approximation, and further explore the application of RL to broader classes of image grammars.

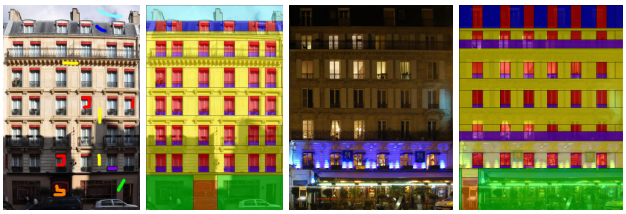


(a) Classic Architecture

(b) Skyscrapers

Figure 8. Grab-Cut on some examples of classic architectures from Barcelona and Budapest, and skyscrapers from California and Paris. Observe the colored brush strokes on the original images, that were used to train the GMM used as merit functions.

(a) Robustness to occlusions



(b) Robustness to illumination

Figure 9. Robustness to artificial and natural occlusions and to challenging illuminations (cast shadows and night lighting).

References

[1] F. Alegre and F. Dellaert. A probabilistic approach to the semantic interpretation of building facades. In *Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*, 2004. 2273

[2] Feng Han and Song-Chun Zhu. Bottom-up/top-down image parsing by attribute graph grammar. In *ICCV*, 2005. 2273

[3] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. In *SIGGRAPH*, 2006. 2273, 2274

[4] P. Müller, G Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of facades. In *SIGGRAPH*, 2007. 2273, 2274

[5] O. Barinova, V. Lempitsky, E. Tretiak, and P. Kohli. Geometric image parsing in man-made environments. In *ECCV*, 2010. 2273

[6] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape prior. In *CVPR*, 2010. 2273, 2274, 2279

[7] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost. In *ECCV*, 2006. 2273, 2275

[8] S. Fidler, M. Boben, and A. Leonardis. Evaluating multi-class learning strategies. In *NIPS*, 2009. 2273, 2274, 2275

[9] S. C. Zhu and D. Mumford. Quest for a Stochastic Grammar of Images. *FTCGV*, 2007. 2273, 2275

[10] E. Charniak. *Statistical Language Learning*. MIT Press, 1993. 2273

[11] K. S. Fu. *Syntactic Pattern Recognition*. Prentice-Hall, 1974. 2273

[12] A Rosenfeld. *Picture Languages: Formal Models for Picture Recognition*. Academic Press, 1979. 2273

[13] Y. Chen, L. Zhu, Lin C, A. Yuille, and H. Zhang. Rapid Inference on an AND/OR graph. In *NIPS*, 2007. 2273, 2274

[14] Z. Tu, X. Chen, A. Yuille, and S.C. Zhu. Image Parsing: Unifying Segmentation, Detection, and Recognition. *IJCV*, 63(2):113–140, 2005. 2273, 2278

[15] J. Schlecht, K. Barnard, E. Spriggs, and B. Pryor. Inferring grammar-based structure models from 3d microscopy data. In *CVPR*, 2007. 2273

[16] A. Toshev, P. Mordohai, and B. Taskar. Detection and parsing architecture at city scale from range data. In *CVPR 10*, 2010. 2273

[17] L. Zhu, Y. Chen, Y. Lin, C. Lin, and A. Yuille. Recursive Templates for 2D Parsing. In *NIPS*, 2008. 2273, 2275

[18] Benjamin Sapp, Alexander Toshev, and Ben Taskar. Cascaded models for articulated pose estimation. In *ECCV*, 2010. 2273

[19] Vittorio Ferrari, Manuel J. Marin-Jimenez, and Andrew Zisserman. Progressive search space reduction for human pose estimation. In *CVPR*, 2008. 2273

[20] Pedro F. Felzenszwalb, Ross B. Girshick, and David A. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010. 2273

[21] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. 2273, 2276

[22] G. Stiny. *Pictorial and Formal Aspects of Shape and Shape Grammars*. Birkhäuser, 1975. 2274

[23] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. Instant architecture. In *SIGGRAPH*, 2003. 2274

[24] J. Hopcroft and J. Ullman. *Introduction to automata theory*. Addison-Wesley, 1979. 2274

[25] T. G. Dietterich. Hierarchical reinforcement learning with MAXQ. *JAIR*, 13, 2000. 2277

[26] S. Singh, T. Jaakkola, and M. Jordan. Reinforcement learning with soft state aggregation. In *NIPS*, 1995. 2277

[27] Sung Chun Lee and Ram Nevatia. Extraction and integration of window in a 3d building model from ground view images. In *Computer Vision and Pattern Recognition*, 2005. 2278

[28] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptative gmmrf model. In *ECCV*, pages 428–441, 2004. 2279