Parsing Facades with Shape Grammars and Reinforcement Learning

Olivier Teboul, *Member, IEEE,* Iasonas Kokkinos, *Member, IEEE,* Loic Simon, *Member, IEEE,* Panagiotis Koutsourakis, *Member, IEEE,* and Nikos Paragios, *Fellow, IEEE,*

Abstract—In this work we use Shape Grammars for Facade Parsing, which amounts to segmenting 2D building facades into balconies, walls, windows and doors in an architecturally meaningful manner. The main thrust of our work is the introduction of Reinforcement Learning (RL) techniques to deal with the computational complexity of the problem. RL provides us with techniques such as Q-learning and state aggregation which we exploit to efficiently solve facade parsing.
We initially phrase the 1D parsing problem in terms of a Markov Decision Process, paving the way for the application of RL-based tools. We then develop novel techniques for the 2D shape parsing problem that take into account the specificities of the facade parsing problem. Specifically, we use state aggregation to enforce the symmetry of facade floors and demonstrate how to use RL to exploit bottom-up, image-based guidance during optimization.
We provide systematic results on the Paris building dataset and obtain state-of-the-art results in a fraction of the time required by previous methods. We validate our method under diverse imaging conditions and make our software and results available online.

Index Terms—Image Parsing, Shape Grammar, Reinforcement Learning, Semantic Segmentation, Markov Decision Processes.

1 Introduction

The proliferation of large urban image datasets, the advent of commercial applications such as Google/Bing maps, and the revival of the scene understanding problem have brought the interpretation of building images at the forefront of computer vision research. The problem has both practical and theoretical interest: on the practical side commercial applications need to automatically interpret massive image datasets in terms of semantically meaningful structures. On the theoretical side solving the problem involves the fitting of models that accommodate structure variation and continuous variables, which is largely unexplored.

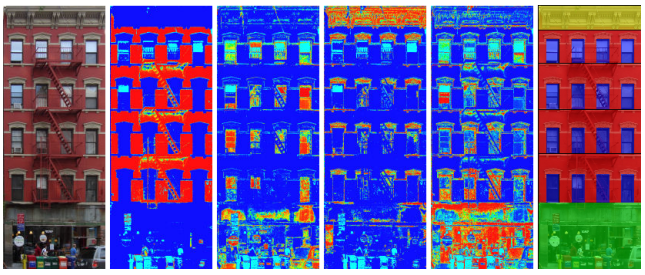
In this work we focus on the task of partitioning a rectified image of a building into a set of predefined semantic classes such as wall, window, balcony and roof. We address the optimization problem illustrated in Fig. 1: we have a pixel-wise merit function indicating the match of the observations with the candidate classes, and we want to find an *architecturally meaningful* partitioning of the image that will optimally bundle together the responses of these merit functions.

Enforcing architectural consistency can be achieved by incorporating knowledge about buildings through proper models. The main challenge lies in dealing with an unknown number of floors or windows, which amounts to accommodating structure variation. We therefore turn to grammatical models which use recursion, and employ *Shape Grammars* whose generative power has been convincingly demonstrated in the computer graphics community [1], [2].

Our contributions are as follows: first we develop a variant of Shape Grammars, termed Binary Split Grammars, which allows us to maintain the expressive power of Shape Grammars, while making them amenable to optimization algorithms for parsing. Second, we develop a Reinforcement Learning-based approach to the 2D facade parsing problem that supports versatile and flexible algorithmic ideas such as state aggregation, data-driven exploration and the incorporation of user defined constraints. A crucial advantage of this approach over Dynamic Programming-based alternatives, such as the 2D extension of the CYK algorithm [3] is speed. We design three appropriate merit functions and show that the performance of our approach transfers successfully over different problem settings. Last, we share online an extension of the dataset of [4] along with our parsing software to help reproduce our experiments and compare with other techniques.

We start by introducing a first example of the problem in 1D in Sec. 2; through this example we introduce basic notions used in the rest of the paper. In Sec.s **??** we present background material on Binary Split Grammars and Markov Decision Processes respectively. The parsing algorithm, some customized

- *O. Teboul is with the MAS laboratory, Ecole Centrale Paris, Grande Voie des Vignes, 92290, Chatenay-Malabry, and Google, Inc. E-mail: olivier.teboul@ecp.fr*
- *I. Kokkinos, L. Simon, P. Koutsourakis and N. Paragios are with Ecole Centrale Paris.*



original wall windows roof shop parsing

Fig. 1. Overview of the method: our goal is to segment the original image using the pixel-wise merit functions for each class (wall, window, roof and shop) and a shape grammar that enforces architectural consistency. The merit function values range from 0 (blue) to 1 (red), and measure data fidelity to the corresponding class. The colors of the segmentation indicate the different classes.

techniques developed in particular for facade parsing as well as specific merit functions are given in Sec.s **??** respectively. We provide experimental results in Sec. 8 and compare to related works in Sec. 9.

2 1D Parsing Example

We start by considering a 1D semantic segmentation example for an image $\mathcal{I}$ of width W and height H, as shown in Fig. 2. In the context of our work this image can represent an artificial floor made of `windows` (white) and `walls` (black).

Our task is to set in alternation the width of the window and the next wall by choosing from a set of admissible widths. The pattern followed by these decisions can be described recursively in terms of the 1D *shape grammar* given in Table 1. The set of possible segmentations that are in accord with this grammar G constitutes the language $L(G)$ of the grammar; we elaborate on $L(G)$ in Sec. 3. In the **shape grammar** paradigm, our problem is to parse the image with the grammar, namely to find the set of grammar rules which combined give an optimal parse of the image.

Optimality is defined through pixel-wise **merit functions** $m(x, y, c)$ that measure how good label c (e.g. wall/window) is for a pixel in position (x, y). In our 1D example the merit at x, y is 1 if c is the right label (`window` on white pixels, `wall` on black pixels) and 0 otherwise. Our task is to find:

$$S^* = \arg \max_{S \in L(G)} \sum_{x,y \in \mathcal{I}} m(x, y, S(x, y)), \qquad (1)$$

where $S(x, y)$ is the label given to pixel x, y by S and $S \in L(G)$ enforces that S is generated by the grammar, i.e. is admissible.

We can rephrase this optimization as a **decision process**: considering that at any time step t we have segmented the image from $x = 0$ to $x = x_t$, our decision amounts to determining the width of the next element. Due to the problem's 1D structure, finding the optimal combination of sequence of decisions (wall and window widths) can be efficiently solved by Dynamic Programming.

Fig. 2. A synthetic signal that has to be segmented into of wall and window (black and white, respectively) areas.

`fl`	$\rightarrow$	`flWa`
`fl`	$\rightarrow$	`flWin`
`flWa`	$\rightarrow$	`wall(x) flWin,` $x \in [\![10, 100]\!]$
`flWin`	$\rightarrow$	`window(x) flWa,` $x \in [\![10, 80]\!]$

TABLE 1
A simple floor grammar. Each value of x represents a different rule, therefore this grammar is made of 164 rules.

To address the 2D problem of facade parsing we will elaborate on three concepts introduced above in a simpler form for the 1D case. First, we use binary split grammars to model the more complex layouts of facades (Sec. 3). Second, we formulate the problem in the Markov Decision Process framework and solve it using Reinforcement Learning (Sec.s **??** respectively). Third, merit functions are learned to deal with the variability of appearances of facade elements (Sec. 7).

3 Binary Split Grammars

Shape Grammars (SGs) were introduced in the 1970's [5], [6] to model complex but structured geometries, and were successfully applied to Procedural Modeling in [1], [2], [7]; more recently SGs have been extended to deal with a broader range of architectural styles [8] and to incorporate physical constraints [9].

Even though SGs have been typically used for synthesis/graphics, our objective is to use them for the inverse problem, namely the parsing of facade images. While the goal of generic image parsing may involve a broad range of geometrical constituents [10], dictionaries made of axis-aligned rectangular shapes [11], [12], [4] may suffice for facades.

We exploit this observation and propose in this section a particular case of SGs called Binary Split Grammar (BSG); the main merit of this particular grammar is that it lends itself to efficient optimization, as we detail in Sec. 5.

3.1 Shapes and Shape Grammars

The basic concept of a shape grammar is a *labeled rectangle*, namely a 5-tuple (c, x, y, w, h), where c is a label or symbol and $(x, y, w, h) \in \mathbb{N}^4$ defines the position and dimensions of an axis-aligned rectangle; for notational convenience we may denote a labeled rectangle as $c(x, y, w, h)$. A shape S is a set of labeled

rectangles: $S = \{s_1, \ldots, s_n\}$; we will consider these rectangles disjoint.

A grammar rule modifies a shape by replacing a labeled rectangle $s_i \in S$ by a set of labeled rectangles $(s_i^1, \ldots, s_i^k)$. In our work we consider only binary split rules ($k = 2$) that split a labeled rectangle in two along either the horizontal or vertical directions. We denote a rule to break symbol A along axis 'ho' (for horizontal) into symbols B and C as:

$$A(x, y, w, h) \rightarrow_{ho:\alpha} \{B(x, y, \alpha, h), C(x+\alpha, y, w-\alpha, h)\}. \quad (2)$$

The dimensions of B and C are uniquely determined given A, the split direction ho, and size α, where $\alpha \geq w$; if $\alpha = w$, C is the empty symbol. For brevity we introduce the shorthand notation:

$$A \rightarrow B(\alpha)C \quad (3)$$

which indicates that shape A is split horizontally ($\uparrow$ means vertically) into a shape of width α and the remainder. For instance, suppose that we have a shape $S = \{A(0, 0, 10, 1)\}$ and a rule $r : A \rightarrow B(3)C$. After applying r on S, we obtain $S = \{B(0, 0, 3, 1), C(3, 0, 7, 1)\}$.

A BSG G is a 4-tuple $(\mathcal{N}, \mathcal{T}, \mathcal{R}, \omega)$, where $\mathcal{N}$ is a set of non-terminals, $\mathcal{T}$ is a set of terminals, ω is a special non-terminal called the axiom and $\mathcal{R}$ a finite set of binary split rules. A labeled rectangle $c(x, y, w, h)$ is terminal if it cannot be further expanded by a rule.

To generate a shape S according to a BSG G we start from the axiom $\{\omega\}$. At each step of the generation a non-terminal element $s_i \in S$ is selected and a rule $r \in \mathcal{R}$ applicable to s_i is chosen. After applying r the labeled rectangle s_i is removed from S and replaced by its offsprings. This process is called a *derivation process* and stops when S only contains terminal elements. We call such a shape a *segmentation*. If the axiom ω corresponds to the image domain, a shape made of terminal elements is an image partition that associates every rectangular region with a label.

We can equivalently represent S in terms of a *parse tree* rooted at ω. During the derivation, the offsprings of s_i are added as its children to the tree. At the end of the process the leaves of the parse tree are terminal elements while its internal nodes represent non-terminal labeled rectangles.

The *language* $L(G)$ is the set of all the possible derivations of the grammar G; in our case this amounts to all possible image segmentations.

3.2 Cyclic symbols and rules

Given a grammar G we say that a symbol c is cyclic if there exists a parse tree in $L(G)$ that has a subtree rooted on c and contains at least another symbol c. To simplify the optimization in Sec. 5, we require that for any rule $A \rightarrow B(x)C$ where both B and C are non-terminals, only C is *cyclic*.

Rules are mutually cyclic if the LHS of one rule appears in the RHS of the other. When dealing with repeated alternated structures (such as wall and windows in the previous example) the use of mutually cyclic rules is a convenient way to deal with an arbitrary number of element repetitions. For instance, Table 1 enables us to express any of the segmentations involved in the 1D example in Fig. 2 thanks to $flWa$ and $flWin$ being mutually cyclic.

4 Markov Decision Processes

In the section we provide the necessary background for Markov Decision Processes, as these are central to understanding the Reinforcement Learning tools that we employ subsequently.

4.1 Definitions

Decision Processes: An MDP is described by a 4-tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, P the transitions probabilities between states and R the expected rewards consecutive to the actions taken at every state.

At time t, the environment's situation is summarized by a state variable, $s_t \in \mathcal{S}$. Based on s_t, the agent takes the action $a_t \in \mathcal{A}(s_t)$. This action modifies the environment, which is reflected in the change of the state from s_t to s_{t+1} as well as a reward r_{t+1}.

The transition from a state s to a state s' consequent to action a is in general non-deterministic, and is governed by a probability distribution $P^a_{ss'}$:

$$P^a_{ss'} = p(s_{t+1} = s' | s_t = s, a_t = a). \quad (4)$$

In our setting the state transitions are deterministic, since these amount to applying deterministic grammar rules; we are thus working with a contrived version of Eq. 4, where p amounts to a Dirac function placing all its mass around the deterministically estimated value of s_{t+1}.

The reward r_{t+1} received for selecting action a in state s and arriving in state s' can again be non-deterministic; we denote its expectation as $R^a_{ss'}$:

$$R^a_{ss'} = \mathbb{E}\left[r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\right], \quad (5)$$

where the expectation uses the conditional distribution on rewards $p(r_{t+1} = r \mid s_t = s, a_t = a, s_{t+1} = s')$ given the combination of s, s', a. As we will see in Sec. 5.3 the treatment of stochastic rewards along the lines of Eq. 5 is essential to the formulation of our parsing algorithm.

As suggested by the forms of Eq.s **??**, the transition probabilities among states and the associated expected rewards have the Markov property, namely depend only on the current state and the selected action and not on the past states and actions.

Return: The goal of the agent is to maximize its long-term reward, also called the *return*, defined in terms of

the amount of rewards accumulated until the end of the process. We call *episode* a sequence of states from an initial one to the end, called the horizon T. The return at time step t is defined as the sum of observed rewards gathered from time $t+1$ to the horizon.

$$\Gamma_t = \sum_{k=t+1}^{T} r_k. \tag{6}$$

We note that we do not necessarily know in advance the value of the horizon T but in our case we practically only consider only finite MDPs, i.e. $T < \infty$.

Policy: A *policy* $\pi(s,a)$ determines the preference that the agent has for taking action a when being in state s; a probabilistic policy can be constructed as:

$$\pi(s,a) = p(a|s), \tag{7}$$

where $p(a|s)$ indicates the probability that the agent will choose action a at state s. We first note that this is an agent-centric quantity, and can in general be independent of the environment statistics. Operationally, we can say that at state s the agent chooses an action a by sampling $p(a|s)$; this then triggers the agent-environment interaction, dictated by $p(s'|a,s)$ and $p(r|s',a,s)$ involved in Eq. 4 and Eq. 5 respectively, which describe how the environment reacts to the agent's actions and rewards them.

A policy dictates a behavior and an associated distribution on returns; and the goal of learning is to change the policy so that the expected return increases. This brings us to the concept of a *value function* that is used to measure the quality of a policy.

Value Function: The *value function* or Q-function $Q^\pi(s,a)$ represents the gain that the agent can expect to accumulate in the long-run, if starting at state s it takes action a and follows the policy π thereafter:

$$Q^\pi(s,a) = \mathbb{E}\left[\Gamma_t \;\middle|\; s_t = s, a_t = a\right]. \tag{8}$$

The expectation is over the joint distribution on states, actions and rewards implied by (a) the probabilistic policy π, which determines the probability with which actions will be chosen at different actions, and (b) the distributions $p(s'|a,s)$, $p(r|s',a,s)$ which indicate the environment's rewards for the agent's actions.

Bellman's equation exploits the problem's Markov nature to express the value function recursively as:

$$Q^\pi(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \sum_{a' \in \mathcal{A}(s')} \pi(s',a') Q^\pi(s',a') \right], \tag{9}$$

where we observe how the different factors of randomness decouple; in particular for the considered (s,a) pair we marginalize over the subsequent states s' and the anticipated rewards r_{t+1} corresponding to the (s,s',a) combinations to recover the expected reward $\sum_{s'} P^a_{ss'} R^a_{ss'}$; and then we consider that starting from s' the agent will choose the subsequent actions according to $\pi(s',a')$, which we therefore use to marginalize $Q^\pi(s',a')$ over a'.

By gathering the equations in Eq. 9 for all state-action pairs we obtain a system of linear equations that can in principle be inverted to get Q^π; but in practice the state space is large and direct inversion may be impossible. **Optimality equations**: We can compare two policies in terms of their value functions: a policy π is better than a policy π' for a given MDP, if:

$$\forall s, \quad \max_a Q^\pi(s,a) \geq \max_a Q^{\pi'}(s,a). \tag{10}$$

We call a policy *optimal* and denote it as π^*, if:

$$Q^*(s,a) \doteq Q^{\pi^*}(s,a) = \max_\pi Q^\pi(s,a). \tag{11}$$

Q^* satisfies the Bellman optimality equation:

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \max_{a'} Q^*(s',a') \right]. \tag{12}$$

The optimal policy is related to Q^*: to maximize cumulative reward, at every state s, the agent must choose action $a^* = \arg\max_a Q^*(s,a)$. An optimal policy is therefore deterministic and derived from Q^*.

4.2 Solving MDPs

The equations above provide us with optimality conditions, but not with means to obtain an optimal policy. The most common approach for this is Dynamic Programming (DP); thanks to the problem's Markov structure, an optimal solution can be obtained by gradually sweeping through states and considering all possible actions for every state. But if the number of state-action combinations becomes large (as is the case in our problem), such techniques may become inefficient. Another option for policy learning is to use Monte-Carlo sampling; there the agent is allowed at each step to take decisions with some randomness, and after the episode finishes, namely after a sequence of actions, the decisions taken are used to assess, and improve a given policy. However in Monte-Carlo sampling the whole process needs to be 'run' before its score is assessed, thus the particular Markov structure of the problem is not being exploited. Reinforcement Learning lies somewhere in between, in the sense that policy improvement takes place in a gradual, 'state-per-state' manner, as in Dynamic Programming, while at every step a single action is considered, as in Monte Carlo. We refer to [13] for an excellent presentation of the connections between these methods.

In particular Reinforcement Learning continuously updates the Q function and the agent's policy in an interleaved manner. In case Q^π is a good estimate of Q^* at every state s the 'greedy' action

$$a^* = \arg\max_a Q^\pi(s,a). \tag{13}$$

should be preferred. However if Q^π is a bad estimate then the policy of the agent should leave some room

Algorithm 1 Q-Learning

$\forall s, a \quad Q(s,a) = 0$
loop
  $s \leftarrow$ first state of the episode
  repeat
    $a^* \leftarrow \arg\max_a Q(s,a)$
    $a \leftarrow$ sample from $\pi_{a^*}(s,a)$ (Eq. 14)
    Take action a, observe s', r
    $Q(s,a) \leftarrow \alpha\left[r + \max_{a'} Q(s',a')\right] + (1-\alpha)Q(s,a)$
    $s \leftarrow s'$
  until s = end state of the episode
end loop

for *exploration*. The problem of balancing the good degree of exploration is called the *exploration - exploitation trade-off*. This balance should also evolve over time, so that the agent explores more at the beginning when it has a poor knowledge of its environment and less at later iterations.

This can be accomplished by using an 'ϵ-greedy policy' which chooses at every state s with probability $1-\epsilon$ the 'greedy' action and with probability ϵ randomly selects another action:

$$\pi_{a^*}(s,a) = \left(1 - \frac{\epsilon(n-1)}{n}\right)\delta(a^*,a) + \frac{\epsilon}{n}\left[1 - \delta(a^*,a)\right], \tag{14}$$

where δ is the Kronecker function, and n is the number of actions available at state s. If the policy tends towards a greedy policy, by making the exploration rate ϵ decrease to 0, then the policy is said to be *greedy in the limit of infinite exploration* (GLIE), which is a common assumption in the analysis of convergence of iterative algorithms ([13], [14]).

Having pinned down how the agent's policy is determined from Q, we now turn to the task of improving it. The Q-Learning algorithm [15], illustrated in Algorithm 1, can learn optimal policies through repeated sampling. At time t of a given episode the agent is in state s, samples an action a that brings it to state s' while the agent collects a reward r. Q-learning updates the estimate of $Q^*(s,a)$ after selecting an action a by using equation (12). The update uses only the new state s' and replaces the expected reward $R^a_{ss'}$ by the observed reward r. The update equation of Q-learning is:

$$Q_{k+1}(s,a) = \alpha_k\left[r + \max_{a'} Q_k(s',a')\right] + (1-\alpha_k)Q_k(s,a), \tag{15}$$

where the learning rate α_k gradually decreases with the number of iterations, ensuring that the algorithm gradually converges [15].

4.3 Hierarchical MDPs

So far we have been considering the case where the application of every rule generates an immediate reward; this is in general a limitation, since in many cases taking an action cannot be directly evaluated. As a concrete example that relates to our task, consider decomposing a facade into a sequence of floors and walls; when picking the height of a floor we cannot directly estimate the merit of this action, as this also requires knowing the actions taken to break a floor into window/wall symbols, and potentially also the windows into glass and balcony terminals.

What we are thus facing is a *hierarchy of tasks*: the reward for a task is defined in terms of the sum of the rewards accumulated during its execution, which can in general require a recursive computation: taking an action amounts to calling a routine, which in turn may call other routines. The merit of the particular action is only available once all the relevant routines have returned. The combination of task hierarchies with MDPs has been addressed in the context of Reinforcement Learning, [16], [17], [18] and is also related to Semi-Markov DPs (SMDPs), in the sense that the reward for an action does not arrive directly after taking the action. In our case we ignore the time lag between the execution of a high-level action and the receipt of the associated reward, since this does not affect the final solution's quality; therefore several of the technicalities related to SMDPs become irrelevant. The Q-learning updates are identical with the ones presented previously, with the difference that the 'immediate' rewards now refer to rewards that are recursively computed through the task hierarchy.

5 Parsing Algorithm

We now formulate parsing as an MDP in the 1D case and a Hierarchical MDP in the 2D one.

5.1 MDP Formulation

We formulate the parsing problem as follows: we are provided with a BSG $G = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \omega)$ of language $L(G)$ and an image $\mathcal{I}$ with pixel-wise merit $m(x,y,c)$ that indicates how appropriate it is to associate a pixel $(x,y) \in \mathcal{I}$ with label $c \in \mathcal{T}$. Our goal is to find the segmentation $S \in L(G)$ that maximizes the global merit:

$$S^* = \arg\max_{S \in L(G)} \sum_{x,y \in \mathcal{I}} m(x,y,S(x,y)). \tag{16}$$

Despite the functional's simplicity, its optimization is hard since the constraints involve the language of the input grammar. To see this, first note that the cardinal number of $L(G)$ does not allow to perform an exhaustive search. Moreover the language $L(G)$ is described implicitly by the grammar; to make sure the language constraint is met, we need to resort to synthesizing the segmentation through a derivation process. Finally we also note that $L(G)$ contains segmentations with various layouts and therefore cannot be described by a fixed number of parameters. Parsing

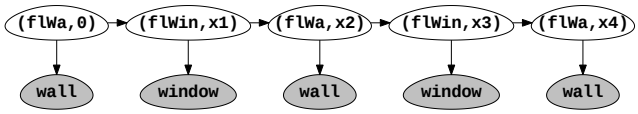


Fig. 3. Constructing a parse tree can be seen as a decision process. At each step the agent is placed on a white node and takes a decision which creates two nodes: the shaded node below is terminal and provides a reward, while the agent moves to the following, non-terminal, white node. Note that we have a large number of possible states, corresponding to the possible values of the second, 'x', argument of the state.

an image with a grammar is therefore tantamount to optimizing over an unknown number of parameters.

Our method is based on the following observation: the grammar derivation is a decision process and parsing an image can be considered as finding the sequence of decisions that best explains the input image. In this work, we consider that an agent is parsing an image by interacting with the (partial) parse tree of the segmentation. At each time step, the agent is at a given node of the tree that represents the state of the environment and chooses a rule to apply on this node; it is the action taken by the agent. A node is a labeled rectangle and applying a rule splits it into two other labeled rectangles. The agent proceeds with the parse in a Depth-First-Search (DFS) order, until all rectangles are decomposed into terminals (end state of the decision process). Whenever a terminal node is generated the agent receives a reward based on the pixel-wise merit of the created region. Since the overall merit is incrementally accumulated during the derivation process its maximization can be cast in terms of a Markov Decision Process; intuitively, the Markov Property can be associated with the grammar's context-free nature.

We note that a connection between parsing and reinforcement learning has been explored in an NLP context in [19]; in our work we establish a connection between parsing with 2D shape grammars and reinforcement learning and address the problem-specific technical hurdles involved, as described below.

5.2 1D Parsing

Getting back to the 1D example presented in Sec. 5.2, we now pin down the states, actions and rewards of the corresponding decision process.

5.2.1 MDP definition

At each time step the agent splits a non-terminal labeled rectangle (c, x, y, w, h) into two and decides where to split it by choosing a rule.

States encapsulate the information that the agent uses when taking a decision. A full-blown state would include the 5-tuple (c, x, y, w, h). For the 1D case y and h can be omitted as they are constant, while w is related with x through the relation $w = W - x$, where W is the size of the image and x, w are the position and width of the state respectively. We thus describe the non-terminal node (c, x, y, w, h) with a the lower-dimensional state:

$$s = (c, x).$$

In the grammar of Table 1 the possible states correspond to labeled rectangles starting at any pixel position x and ending at position W, where W is the size of the image. Since there are two non-terminal labels ('flWin' and 'flWall') and $W = 300$, our MDP involves 600 states.

Actions correspond to the rules comprising the grammar (Table 1). Each value of the split parameter x corresponds to a different rule; the number of possible values of x is discretized to a fixed range of values. In the particular example of Sec. 2, windows range from 10 to 80 pixels wide and walls from 10 to 100.

Rewards are obtained by aggregating the pixel-wise merit of regions. In particular for the 1D case each rule creates a terminal and a non-terminal rectangle. The non-terminal rectangle represents what remains to be segmented. Denoting by (c, x, y, w, h) the terminal labeled rectangle created by the rule, we define the region-wise merit as the sum of the pixel-wise merit of the region. This gives the immediate reward r:

$$r = m(c, x, y, w, h) = \sum_{i=x}^{x+w} \sum_{j=1}^{h} m(i, j, c). \qquad (17)$$

The sum of the rewards accumulated along an episode is thus equivalent to the functional defined in equation (16): solving the associated MDP is equivalent to parsing the input image.

Constructing the parse tree can be seen as a sequential process (see Fig. 3): at each white node the agent takes a decision which determines the node to which it moves next, as well as the received reward, computed based on the shaded node below. The action that gives the biggest immediate reward is not necessarily the best action in the long run; it is therefore necessary to establish a policy that will allow the agent to act in a more 'far-sighted' manner.

5.2.2 Solving the MDP

We now compare how Dynamic Programming (DP) and Reinforcement Learning solve the MDP defined above. DP is straightforward for this problem since we have a small number of states and actions, while the state-action transitions and rewards are deterministic. Fig. 4 shows the $Q^*(s, a)$ function for different positions of the symbol 'flWin' and every possible action. Low values are blue and high values are red.

Unlike DP Q-learning is guaranteed to deliver the optimal solution only upon convergence; solutions found in between may be suboptimal. To assess the variability in performance across different runs, we ran a Q-learning agent for $T = 10000$ episodes, using

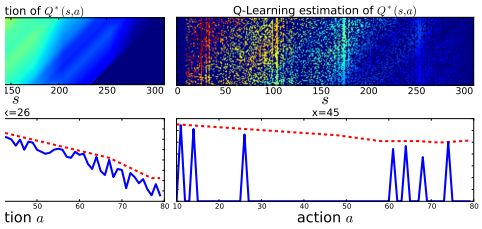


Fig. 4. Comparison of the Q^* function computed by DP and RL on the 1D parsing example of Sec. 2, for the window symbol. The first row shows the values of $Q^*(s,a)$ as estimated by DP (left) and RL (right): the horizontal axis indicates the state s at which the agent currently is, the vertical axis the putative action a taken at s, and the color shows the value of $Q(s,a)$ (blue is low and red is large). The second row shows the values of $Q^*(s,a)$ estimated by DP (red dashed) and RL (solid blue) at two different states, corresponding to the columns of $Q(s,a)$ at $x=26$ (left) and $x=45$ (right).

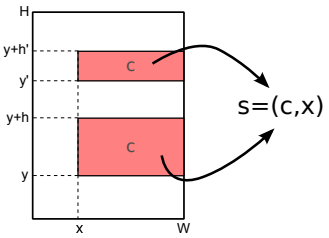


Fig. 5. Two different atomic shapes in red `(c,x,y,w,h)` and `(c,x,y',w',h')` are mapped to the same state `s=(c,x)`, since along the split dimension X they share the same position `x` and symbol `c`.

a ϵ-greedy policy where both ϵ and the learning rate decrease exponentially from 1 to 0.01. The run-time is about the same as the one with DP, in the order of 10 seconds. In practice, the Q-function is neither uniformly, nor exactly evaluated. The agent explores the state-space, but gives more credit to the reward-wise interesting regions. To illustrate this, consider that we are at position $x=26$ (lower left in Fig. 4). The agent is in a region of the state space that pays-off: this is indeed a white region of the image. The exact DP solution, in dashed red, is well approximated by the Q-learning one represented in solid blue. On the contrary, in the middle of a poorly interesting region (lower right) the estimate of Q^* is far from its exact value. There, the state of the agent corresponds to a position which is in the middle of a black region. Putting a window in a region representing walls does not yield a large reward, irrespective of the window size. The fact that position $x=45$ consistently did not result in large rewards resulted in the agent passing more rarely through it at later rounds, and therefore a worse approximation of $Q^*(s,a)$ at that particular s.

This also indicates the main mechanism through which RL can accelerate optimization, namely by devoting fewer computation resources to unpromising states: unlike DP, RL evaluates $Q^*(s,a)$ only around promising parts of the state space. This property of RL becomes increasingly important in the 2D parsing case, where the complete knowledge of the best solution at every state might be both computationally intractable and unnecessary.

5.3 2D Parsing

We now generalize the 1D case to the 2D one using Hierarchical MDPs: since the procedural mechanisms are the same in 1D and 2D we can still consider an agent responsible for the derivation process; as in 1D the agent uses a DFS scheme for parsing and takes a sequence of decisions involving actions (i.e. rule applications) and associated rewards (i.e. merit function integrals). However in the 2D case some rules are not placing any terminal symbols but only non-terminals; the agent now faces a hierarchy of tasks. We now define the states and rewards involved in this task hierarchy.

5.3.1 State Aggregation

As outlined in Sec. 5.2 each node (state) is described in terms of four geometric parameters and one label, (c,x,y,w,h). For the 1D case we omitted (y,h) as irrelevant, and eliminated w using $w=W-x$. Even though parameters should in principle be dealt with in 2D parsing, this results in a dramatic increase in computational complexity.

Equally important with speed is the quality of the decisions that such an agent learns. For example two different floors have different vertical positions y and thus we could not prevent the agent from learning two different policies on these floors, even though we expect them to be structured the same way. In order to introduce consistency we force the states representing the positions of the elements in these floors to be the same. Thus, we propose to replace the full-blown state $s=(c,x,y,w,h)$ with a condensed, *aggregated* state:

$$s^\alpha=(c,z), \tag{18}$$

where $z\in\{x,y\}$ is the position along the split direction, namely the direction along which the labeled rectangle is split in the current iteration. This idea is illustrated in Fig. 5: when splitting along a direction the positions along other dimensions are considered to be irrelevant. From now on we will only use aggregated states, so we will omit the α superscript for simplicity. During the execution of our algorithm the full-blown state is only useful in measuring the rewards for terminals: this requires a full-blown indication of their rectangular domains. However the Q functions for all non-terminals are learned using aggregated states.

The first advantage of state aggregation consists in reducing the number of possible states, which now shares the same order of magnitude as in the 1D case. The second advantage consists in ensuring consistency along the facade; without state aggregation one

Algorithm 2 Parsing Algorithm

$\forall s, a \quad Q(s,a) = 0$
$\forall s \quad \pi(s,.) \sim U(\mathcal{A}(s))$
$\epsilon = \epsilon_0$, $k = 1/n \log(\epsilon_0/\epsilon_n)$
for $i = 0 \rightarrow$ n **do**
  reset environment
  $s \leftarrow$ (Axiom, 0)
  $\epsilon \leftarrow \epsilon \cdot e^{-k}$
  while $s \neq$ end state **do**
    runtask(s, Q, π)
  end while
end for
return greedy policy w.r.t. Q

Procedure 3 runtask(s, Q, π)

reward $\leftarrow 0$
while $s \neq$ end task **do**
  $a^* \leftarrow \arg\max_a Q(s,a)$
  $a \leftarrow$ sample from $\pi_{a^*}(s,a)$ (Eq. 14)
  Take action a observe s', r
  if $s' \in \mathcal{N}^0$ **then**
    $r+=$ runtask (s', Q, π)
    $s' \leftarrow$ observe new state
  end if
  reward += r
  $Q(s,a) \leftarrow \alpha\left[r + \max_{a'} Q(s',a')\right] + (1-\alpha)Q(s,a)$
  $s \leftarrow s'$
end while
return reward

would be forced to use Context-Sensitive Grammars or include the whole parse tree in order to enforce symmetry constraints, e.g. for different floors. Instead of such computationally intractable alternatives, we propose to use a common policy over all non-terminals which should be split in a common way. For instance, when splitting floors, the learned policy will depend exclusively on the horizontal coordinate, and not on the height of the floor. This enforces symmetry constraints implicitly, aligning windows across floors, or balconies inside of floors.

These advantages come at the price of stochasticity in the decision process. In particular the uncertainty emerges due to stochasticity in the rewards: since state aggregation can map two different labeled rectangles to the same state (e.g. two windows lying at different floors, as shown in Fig. 5), the agent can obtain different rewards, while performing the same action on the same aggregated state. This is why the ability of Reinforcement Learning to cope with stochastic rewards becomes indispensable in our problem setting.

5.3.2 Hierarchical Learning

As mentioned earlier, the 2D nature of the problem calls for more complex BSGs where a rule may create two non-terminal symbols. In that case, we still need to define a reward subsequent to such an action.

A rule is of the form: $A \rightarrow B(x)C$. It creates two nodes $B(x_B, y_B, w_B, h_B)$ and $C(x_C, y_C, w_C, h_C)$, with C being potentially empty. From these two nodes, only one can be *cyclic*. We always start by deriving the acyclic one first and assume here it is B. Then we have only two cases: B is either terminal or it is non terminal and acyclic.

If B is **terminal**, we are facing the same case as in 1D: the reward r is received immediately after the application of the rule and is equal to the region-wise merit $m(B, x_B, y_B, w_B, h_B)$. The next state is either (C, x_C) or (C, y_C) depending on the direction along which C is split.

If B is **acyclic**, it represents a *task*: the reward associated with the rule is only received when the task is completed, namely when the branch rooted at $B(x_B, y_B, w_B, h_B)$ is fully derived. The reward's value equals the sum of the rewards collected during the derivation of this branch and is stochastic; applying twice the same rule might lead to two different rewards since these rewards depend on decisions taken earlier during the completion of the task.

5.3.3 Optimization strategy

As summarized in Algorithm 2 and Procedure 3, we use a Q-learning agent that iteratively segments facades until converging to an optimal policy. In each episode the agent sequentially builds the segmentation by selecting one rule (action) at a time based on a local information (state). By applying a rule, it may create a terminal symbol, a subtask or a cyclic symbol. Then it receives a reward and reaches a new state where it faces a new decision. The value function is iteratively learned by Q-learning updates. After convergence, reached after around 10^3 episodes, we deterministically parse the facade by following the greedy policy with respect to the estimate of $Q^*(s,a)$. By virtue of being deterministic, and using a policy defined on aggregated states, the delivered parse satisfies symmetry constraints. Moreover, despite the large dimensionality of the original space of states and actions, state aggregation allows us to compactly store the action-value function in a few MBs of RAM.

A potential concern is about the optimality of the parsing process in this modified setting. Intuitively we can justify our scheme 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 when parsing floors at those locations and eventually lead also the local policy for floor splitting closer to the optimal one. Similarly, if the floor-parsing 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

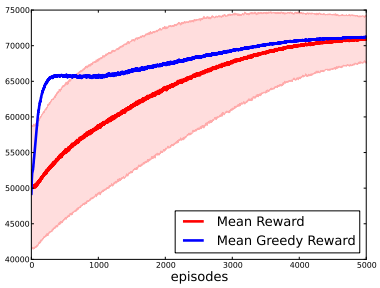


Fig. 6. Convergence of the RL parsing algorithm; we plot returns versus episodes. The trajectories of returns (red) and greedy returns (blue) averaged over 1000 identical agents. In pink is the region representing the standard deviation of the returns. We can see that the fluctuations of the return are decreasing with the iterations. The returns converge to a fixed value.

high- and low- level policies, but as in all EM-based schemes local minima can potentially occur.

5.4 Analysis of Convergence on Artificial Data

To verify the convergence of the Q-learning algorithm, we consider the following test case: the input is a simple binary white/black 2D facade with three floors and four windows per floor that we want to parse with a BSG that has two terminal symbols: `wall` and `window`. On such an artificial image, we already know the ground truth segmentation and therefore we use the following pixel-wise merit function: $m(x, y, c)$ is equal to 1 if c corresponds to the ground truth at (x, y) and 0 otherwise.

We run $N = 1000$ identical Q-learning agents for $T = 5000$ episodes. The learning rates α and the exploration rate ϵ are decreasing exponentially from 1 to 10^{-3}. For each agent i and for each episode k we get a return Γ_i^k, as well as a 'greedy return' G_i^k at episode k. The latter represents what the agent can get while greedily following the policy learned after k episodes. For an individual agent i the trajectories Γ_i^k and G_i^k are noisy. To remove the effects of noise we average over multiple agents and obtain the curves $\mu(\Gamma^k)$ and $\mu(G^k)$ in Fig. 6. We also show the evolution of the standard deviation of the returns $\sigma(\Gamma^k)$ by plotting the curves $\mu(\Gamma^k) \pm 3\sigma(\Gamma^k)$.

We can see in Fig. 6 that the two curves are in average increasing towards the same limit while the parses become better with time. The convergence towards the optimal parse depends on the number of episodes, the learning rate and the exploration rate, which are the only three parameters of the algorithm. In all of our experiments we use a common set of parameters, while in our experience performance was robust to their changes.

6 Enhanced Parsing

The proposed parsing algorithm is a direct application of Q-learning. However, several problem-specific properties may be exploited to obtain faster and better parses. In this section we present two techniques that adapt the RL framework to enhance the parsing algorithm: data-driven exploration and user-defined constraints.

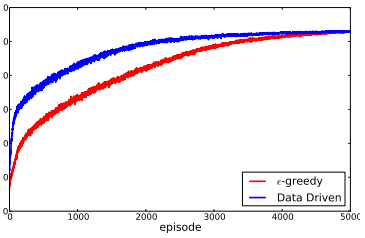


Fig. 7. Speed-up with data-driven exploration; we plot returns versus episodes.. The agents with a data-driven policy converge in average faster towards the solution, that the ones using a plain, ϵ-greedy policy. The curves are averaged over 1000 identical agents.

6.1 Data-Driven Exploration

Following the idea of Data-Driven Markov Chain Monte-Carlo (DDMCMC) [20], we propose to drive the rule selection process using bottom-up cues. The idea is that the border between two semantic regions should match the strong gradients along the split direction. While splitting a labeled rectangle at position x, we should give more credit to actions of parameter w such that the image gradients at $x + w$ are strong.

Similarly to [21], we build two signals, representing the cumulative gradients in both directions.

$$\forall i \leq W, \quad h_X(i) = \sum_{j=1}^{H} ||\nabla_x G_\sigma * \mathcal{I}(i, j)||, \tag{19}$$

where G_σ is a Gaussian Kernel that smooths the signal. We obtain h_Y in a similar way.

Assuming to be in state $s = (c, x)$ with actions $\mathcal{A}(s) = \{a_1, \ldots, a_n\}$ corresponding to parameters $w_1, \ldots, w_n$ and with a^* the greedy action, we define the data-driven policy $\pi(s, a_i)$ as:

$$\pi(s, a_i) = (1 - \epsilon)\delta(a_i, a^*) + \epsilon \frac{e^{h_X(x+w_i)}}{\sum_{j=1}^{n} e^{h_X(x+w_j)}}. \tag{20}$$

An equivalent vertical version can be obtained if s is to be split along Y. We keep the property that the data-driven policy tends to a greedy policy when ϵ goes to zero, while keeping all probabilities $\pi(s, a)$ positive. Such a policy guides the exploration towards interesting locations (namely, with strong gradient) but does not compromise the convergence of the algorithm.

Coming to *experimental validation*, Fig. 7 shows the difference between data-driven and ϵ-greedy policies on an artificial image such as the one used in Fig. 6. Both experiments were conducted with the same settings. We chose $\sigma = 5$. We observe that the blue curve

is always above the red one, which indicates that data-driven exploration enables the agent to converge faster towards the optimal solution.

Quantitatively, we measure on Fig. 7 the number of iterations needed to reach 90% of the final return with and without data-driven exploration. Using data-driven policy an agent reaches this level after 693 iterations in average, and after 1761 iterations with an ϵ-greedy policy. The speed-up obtained on this particular example is 2.5.

6.2 User-Defined Constraints

In real-world applications the user may require some interaction with the parsing algorithm, for instance by manually drawing a labeled rectangle (c, x, y, w, h) on the image; such a constraint can be easily integrated in the RL framework.

In particular we increase the rewards associated with decisions that create (c, x, y, w, h), 'priming' the agent to do so. For example, if the pixel-wise merit function is always between 0 and 1, we decide to give a pixel-wise reward of 2 when the constraint is met.

These constraints are not hard since it is possible that the agent finds a more profitable parse which violates the constraint. In practice these constraints are typically satisfied to a substantial extent, and are easy to define and integrate with real data.

7 Merit Functions

The merit functions are defined on the terminals and are involved in the computation of the rewards. In Sec. 5, we have considered the pixel-wise merit $m(x, y, c)$ as an input of the method without explaining how it can be obtained. One strength of the proposed approach is to support any type of merit functions.

As we want to exploit both cases where training data are available and also cases where only the grammar is available we consider below three different ways to create merit functions.

If training data is available in the form of segmentation annotations we can obtain *supervised merit functions* from the posterior probabilities of multi-class classifiers:

$$m(x, y, c) = p(c|x, y, \mathcal{I}), \tag{21}$$

In particular we have considered both Random Forest (RF) and Gaussian Mixture Model (GMM) classifiers.

The **random forest merit** uses the classifier introduced by Breiman [22]. We use the classifiers of [4], where the RF is made of 10 trees of depth 12 and as feature vectors RGB patches of size 13×13.

The **Gaussian Mixture Models merit** is based on the RGB values of individual pixels selected by the user through brush strokes on the image for each terminal symbol of the BSG. As in [23], we fit a GMM per class assuming the independence of the pixels. We give an example of GMM merit functions for four different labels in Fig. 1; we note that even though the merits are noisy locally, the regularity bias introduced by the model results in a good segmentation.

Both RF and GMM merits are making use of some training examples and therefore require some amount of user interaction. To accommodate also the common case where training data is not available we consider the learning of *unsupervised merit functions*. In particular for simpler cases where the BSG has only two terminal windows, wall and window, we can separate the two classes based on the heuristic introduced by [24]: the hue value distinguishes the walls from the windows. We then build a merit $m(x, y, c)$ by applying a K-Means on the hue values. The bigger cluster is considered to be the walls and the other one the windows. The merit $m(x, y, c)$ is the relative distance of the hue value of pixel (x, y) to both cluster centroids.

8 Experimental Validation

We validate our parsing framework quantitatively and qualitatively on existing and new benchmarks. For the sake of reproducibility, we make our parsing software available online:
vision.mas.ecp.fr/Personnel/teboul/grapesPage/

8.1 Quantitative Validation

8.1.1 Facades Benchmark [4]

To measure the performance of the parsing algorithm, we run it on the facades benchmark proposed in [4] and compare to the results obtained in [4].

For the reported results we use a Randomized Forest classifier made of 10 trees of depth 12, and using 39-Dimensional features (RGB patches of size 13). The classifier is trained on 20 images and its posterior probabilities are used as merit functions. Besides, we run the RL parsing algorithm on the 10 images for 3000 episodes using the complex Haussmannian BSG made of 281 rules. On average, the agent has to take around 80 decisions per episode.

As can be seen from Fig. 8, our method gives typically better results than [4] on the same benchmark; however the results of our method are delivered in a fraction of the time. In particular [4] needs around 10^6 generations and about 10 minutes to converge, whereas the RL-based approached only needs 2000 iterations and about 30 seconds to converge, while at the same time avoiding local minima more effectively.

8.1.2 Monge Benchmark

We extend the previous benchmark from 10 to 84 facades, to increase the variety of buildings to parse. To make the results commensurate we use the same RF classifier as the one used in the previous experiment.

We call pixel-wise labeling the results obtained while labeling each pixel independently with the most

0	**81**	11	3	0	5	0	0	**27**	15	15	13	19	4	8	**68**	23	4	0	4	2	0	*window*
+1	5	**84**	7	1	1	0	2	5	**63**	11	9	4	2	7	3	**87**	7	0	1	0	1	*wall*
−9	10	26	**63**	0	0	0	1	11	17	**34**	13	12	2	11	9	24	**64**	0	1	0	1	*balcony*
+13	0	2	0	**84**	0	0	14	2	2	1	**81**	3	0	10	0	1	0	**53**	0	0	46	*door*
+6	10	4	0	0	**86**	0	0	10	6	11	4	**54**	10	4	6	3	0	0	**83**	8	0	*roof*
0	2	0	0	0	4	**94**	0	4	3	3	1	14	**75**	1	1	0	0	0	3	**96**	0	*sky*
+2	0	1	0	2	0	0	**97**	6	12	10	42	7	0	**22**	0	6	1	6	0	0	**88**	*shop*

RL Parsing Benchmark [4] | Pixel-wise Monge Benchmark | RL Parsing Monge Benchmark

Fig. 8. Quantitative results: We report confusion matrixes on the Facades Benchmark of [3] and the Monge Benchmark, which is an extended version of the former. The left table compares to the results in [3]; green indicates an increase in the on-diagonal element and red a decrease. We observe that on average performance improves. The middle table reports the performance of the pixel-wise labeling baseline on the Monge benchmark, while the right table reports the performance of our algorithm on the same benchmark.

probable label according to the RF classifier only. We use it as a baseline to compare to our method on this new benchmark.

For each image we repeat five times the Q-learning algorithm, and use 5000 episodes each time. We use data-driven exploration and the Randomized Forest-based rewards. We keep the parse that maximizes the global return and build a confusion matrix for the whole dataset.

First we observe on the diagonal of the third matrix a decrease in the detection performance of windows, balconies and doors, compared to the small dataset of [4] (see diagonal of the first and third matrices).This can be attributed to the high appearance variability and the small element size: even when the method finds the windows, missing some pixels can result in a substantial drop in the detection rate.

Second we observe that the parsing algorithm (third matrix) outperforms the pixel-wise segmentation based on the Randomized Forest only (second matrix), except for the specific case of doors. From the last line of the pixel-wise confusion matrix we notice that shops are more frequently labeled as doors rather than shops. Note that this only happens for doors where symmetry/state aggregation cannot improve the decision by transferring information across multiple floors.

In conclusion, the parsing algorithm performs quite well even when the data term (reward) is poorly informative and noisy as it is the case here. The lack of quality in the data term is compensated by the context introduced by the grammar and a method to parse it properly.

8.2 Qualitative Validation

The proposed quantitative validation is restricted the Haussmann BSG and the Randomized Forest reward. In order to demonstrate the generality of our algorithm we test it on different BSGs, rewards, and topologies, as well as challenging viewing conditions.

8.2.1 Other Grammars and Rewards

In Fig. 10 we give an example of parsing obtained with a grammar describing a class of binary segmentations. Fig. 11 shows examples of parsing of a more complex grammar that we call the 4-color BSG; this grammar involves four kinds of terminal elements, combined with GMM rewards. Even if the obtained results are inaccurate, they can provide excellent initializations for subsequent segmentations.

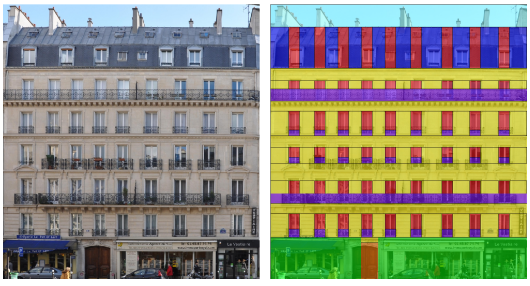


Fig. 9. Parsing Parisian facades. The segmentation of the roof is wrong since the grammars forces it to be the same as the one of the floors. In spite of some geometrical mistakes, the structure of the building is properly parsed.

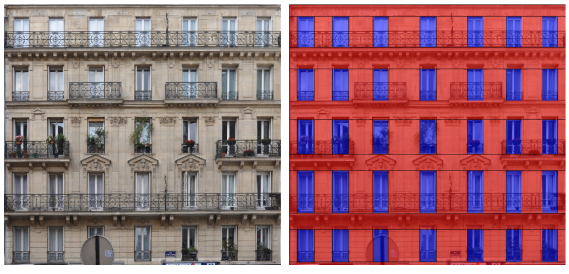


Fig. 10. Parsing facades with a binary BSG. On the left the original image, on the right the optimal parse.

8.2.2 Other Facade Topologies

One of the main advantages of our algorithm is that it does not require us to know the number of architectural elements. Our algorithm takes as input a grammar that restricts segmentations to be axis-aligned but allows for an arbitrary number of parts. In Fig. 12 we apply our parsing method to skyscrapers. The difficulty of these examples lies in the number of decisions the agent must take to perform a segmen-

Fig. 11. Parsing facades with a 4-color BSG. From left to right: original image, user's brush strokes to train a GMM classifier, pixel-wise segmentation using the GMMs, optimal parse with our algorithm.

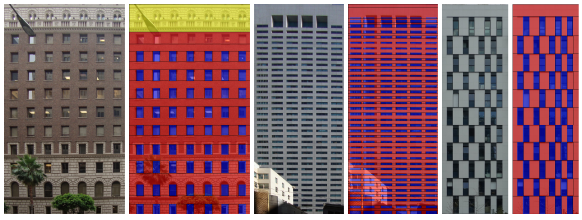


Fig. 12. Despite the large and unknown number of floors, our algorithm accurately parses skyscraper images.

tation, since the problem's complexity scales with the number of possible decisions. To further illustrate the flexibility of our method, in the rightmost example we parse the input facade with a BSG that enforces an alternation of two kinds of floors.

8.2.3 *Robustness to Viewing Conditions*

An important property of the proposed framework is directly inherited from the procedural modeling and the state aggregation: since all the floors of a facade as treated as one through a BSG, our approach is robust to occlusions and lighting conditions, both of which result in corrupted merit functions. Our method leverages upon the symmetry constraint, and transfers segmentation information across structures with uncorrupted merit functions, for instance floors above/below a corrupt one. In Fig. 13 we provide examples of occlusions and lightings that are correctly handled by the proposed framework. We observe that the algorithm 'hallucinates' windows behind the occlusions, due to the model's bias for symmetry. One can argue that this may be too hard a constraint in general; but in particular for buildings it is common enough to be worth 'hard-wiring' into a model. A simple alternative to explicitly accommodate occlusions in our model would be to introduce an occlusion 'class', as e.g. in [25], and then penalize its occurrences, so that occlusions are introduced wherever the class suggested by the symmetry cue can no longer explain the data; we intend to explore this option in future work.

Fig. 13. Robustness to viewing conditions and occlusions.

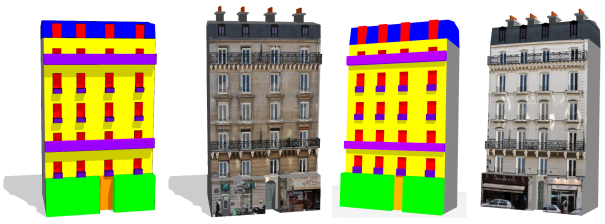


Fig. 14. Image-based procedural modeling of buildings: we turn the estimated 2D rule sequence into a 3D sequence, and manually add the depth variables. The input image is used as a texture map for the grammar-generated 3D building model.

8.3 Image-based Modeling

The procedural modeling framework adopted in this paper makes it easy to bridge the gap between 2D and 3D: after parsing we turn the 2D grammar rules into 3D ones by adding an arbitrary realistic depth to each labeled rectangle.

In Fig. 14 we apply this approach on some Haussmannian buildings. Furthermore the procedural modeling framework also enables us to decorate the final models by adding primitives consistently with the ones found by parsing. Cornices, or balcony supports are difficult to detect but their positions and sizes are closely related to the ones of balconies and windows.

9 Related Work

Even though image parsing has been of interest since the early days of computer vision [26], flat, CRF- or MRF-based approaches [27], [28], [29] have prevailed in the image labeling task over the last decade. Recently several works have investigated the use of hierarchical models for scene [30], [10] and object parsing [20], [31], but to the best of our understanding none of these are directly applicable to facade parsing, where a large number of unknown structuring elements is involved.

Coming to works that address facade parsing, bottom-up approaches rely on low-level processing for lattice [32], [33], [34] or windows detection [21], [35] or on repetitions and symmetry of keypoints [36], [37] and regions [38], [39], [40]. In order to recover from front-end failures certain approaches have also proposed to put users in the parsing loop [41]. Even though our method can exploit these ideas, e.g. through data-driven exploration, the main advantage

of our work consists in making a model-based approach practical.

Grammar-based methods have been recently proposed to parse facades: [42], [43] embed single grammar rules in an MRF, but do not employ a procedural grammar, which allows for increased flexibility. Some other methods cope with procedural 3D reconstruction from multiple views [44]. The works which use procedural grammars, and are closest in spirit to our work are [11], [12], [4]. The first one introduces the facade parsing problem with a shape grammar. The rjMCMC optimization iteratively modifies the current parse and evaluates its quality based on low-level image cues. Working in a similar vein [12] introduce an MDL criterion, while [4] use simpler optimization but a more sophisticated merit term obtained through supervised learning. Our method retains the model-based constraints on the facade segmentation of [11], [12], and the discriminatively trained merit of [4], but replaces the time-consuming rjMCMC of [11] and the prone to local minima approach of [12] with a robust and efficient optimization approach based on RL.

10 Conclusion

In this paper we propose an optimization-based framework for parsing rectified images of facades. We first define a subclass of shape grammars we call Binary Split Grammars that are well suited to describe various facade layouts. Then we introduce a new parsing algorithm based on Reinforcement Learning principles that formulates the parsing problem as a hierarchical decision process. This algorithm empirically outperforms existing methods both in terms of segmentation quality and speed. It supports any kind of BSGs, does not make any assumptions on the number of elements of the facade, and by virtue of being model-based delivers solutions which are robust to occlusions or specific lighting conditions. In addition, we have introduced a benchmark of Parisian facades, which was used to compare our algorithm to existing approaches. Our results have demonstrated that we achieve state-of-the-art results in a fraction of the time required by other methods. In future work we plan first to transpose this 2D parsing method to 3D parsing from multiple views that may require to integrate techniques developed for 3D stereo reconstruction in our problem.

Acknowledgments

O. T., L. S and P. K. were supported by the Microsoft Research Cambridge PhD scholarship program; I. K. was supported by Project ANR-10-JCJC-0205; we thank the Reviewers for their constructive feedback.

References

[1] P. Wonka, M. Wimmer, F. X. Sillion, and W. Ribarsky, "Instant architecture," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 669–677, 2003.

[2] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool, "Procedural modeling of buildings," *ACM Transactions on Graphics*, vol. 25, no. 3, p. 614, 2006.

[3] M. I. Schlesinger and V. Hlavác, *Ten Lectures on Statistical and Structural Pattern Recognition (Computational Imaging and Vision)*. Springer, 2002.

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

[5] G. Stiny and J. Gips, "Shape Grammars and the Generative Specification of Painting and Sculpture," in *Information Processing 71*, C. V. Freiman, Ed., vol. 71. North-Holland, 1972, pp. 1460–1465.

[6] G. Stiny and W. J. Mitchell, "The Palladian grammar," *Environment and Planning B: Planning and Design*, vol. 5, pp. 5–18, 1978.

[7] S. Havemann, *Generative mesh modeling, PhD Dissertation*. Braunschweig, 2005.

[8] S. T. Teoh, "Generalized Descriptions for the Procedural Modeling of Ancient East Asian Buildings," *Computational Aesthetics in Graphics, Visualization, and Imaging*, 2009.

[9] E. Whiting, J. Ochsendorf, and F. Durand, "Procedural modeling of structurally-sound masonry buildings," *ACM Transactions on Graphics*, vol. 28, no. 5, p. 1, Dec. 2009.

[10] S.-C. Zhu and D. Mumford, "A stochastic grammar of images," *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 4, pp. 259–362, 2006.

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

[12] N. Ripperda and C. Brenner, "Application of a formal grammar to facade reconstruction in semiautomatic and automatic environments," in *Proceedings of the 12 th AGILE Conference on GIScience, Hanover, Germany*, 2009, pp. 1–12.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[14] S. Thrun, "The role of exploration in reinforcement learning," in *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, D. White and e. D.A. Sofge, Eds. Van Nostrand Reinhold, Florence, 1992.

[15] C. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, Cambridge University, Cambridge, England, 1989.

[16] A. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, no. 4, pp. 341–379, 2003.

[17] T. G. Dietterich, "Hierarchical reinforcement learning with MAXQ," *Journal of Artificial Intelligence Research*, vol. 13, 2000.

[18] B. Marthi, S. J. Russell, and J. Wolfe, "Angelic semantics for high-level actions," in *ICAPS*, 2007.

[19] G. Neu and C. Szepesvári, "Training parsers by inverse reinforcement learning," *Machine Learning*, 2009.

[20] S.-C. Zhu, R. Zhang, and Z. W. Tu, "Integrating Top-Down/Bottom-Up for Object Recognition by Data-Driven Markov Chain Monte Carlo," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[21] S. C. Lee and R. Nevatia, "Extraction and Integration of Window in a 3D Building Model from Ground View images," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004.

[22] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[23] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, "Interactive image segmentation using an adaptative GMMRF model," in *European Conference on Computer Vision (ECCV)*, 2004.

[24] C. Liu and A. Gagalowicz, "Image-based Modeling of Haussmannian Facades," *International Journal of Virtual Reality*, vol. 9, no. 1, pp. 13–18, 2010.

[25] R. Tylecek and R. Sára, "Modeling symmetries for stochastic structural recognition," in *ICCV Workshops*, 2011.

[26] Y. Ohta, T. Kanade, and T. Sakai, "An analysis system for scenes containing objects with substructures," in *International Joint Conference on Pattern Recognitions*, vol. 1, 1978.
[27] A. C. Berg, F. Grabler, and J. Malik, "Parsing Images of Architectural Scenes," *IEEE International Conference on Computer Vision (ICCV)*, pp. 1–8, 2007.
[28] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 2–23, Dec. 2007.
[29] J. Tighe and S. Lazebnik, "SuperParsing: Scalable Nonparametric Image Parsing with Superpixels," in *European Conference on Computer Vision (ECCV)*. Heraklio, Greece: Springer, 2010, pp. 352–365.
[30] Z. W. Tu and S.-C. Zhu, "Image segmentation by data-driven Markov chain Monte Carlo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, 2002.
[31] Y. Chen, A. Yuille, and L. L. Zhu, "Unsupervised Learning of a Probabilistic Grammar for Object Detection and Parsing Unsupervised Learning of a Probabilistic Grammar for Object Detection and Parsing," *Science And Technology*, 2007.
[32] T. Leung and J. Malik, "Detecting, localizing and grouping repeated scene elements from an image," *European Conference on Computer Vision (ECCV)*, no. April, pp. 546–555, 1996.
[33] Y. Liu, Y. Tsin, and W. Lin, "The promise and perils of near-regular texture," *International Journal of Computer Vision*, vol. 62, no. 1, pp. 145–159, 2005.
[34] B. Weber, P. Müller, P. Wonka, and M. Gross, "Interactive Geometric Simulation of 4D Cities," *Computer Graphics Forum*, vol. 28, no. 2, pp. 481–492, 2009.
[35] S. Reznik and H. Mayer, "Implicit Shape Models, Model Selection, and Plane Sweeping for 3D Facade Interpretation," in *Photogrammetric Image Analysis (PIA)*, 2007, p. 173.
[36] P. Musialski, P. Wonka, M. Recheis, S. Maierhofer, and W. Purgathofer, "Symmetry-based facade repair," in *Vision, Modeling, and Visualization Workshop (VMV)*. Citeseer, 2009.
[37] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu, "Translation-Symmetry-based Perceptual Grouping with Applications to Urban Scenes," in *Asian Conference on Computer Vision*, Queenstown, New Zealand, 2010.
[38] P. Koutsourakis, L. Simon, O. Teboul, G. Tziritas, and N. Paragios, "Single View Reconstruction Using Shape Grammars for Urban Environments," in *IEEE International Conference on Computer Vision (ICCV)*, 2009.
[39] P. Müller, G. Zeng, P. Wonka, and L. Van Gool, "Image-based procedural modeling of facades," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 85, 2007.
[40] C.-H. Shen, S.-S. Huang, H. Fu, and S.-M. Hu, "Adaptive partitioning of urban facades," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH ASIA 2011*, vol. 30, no. 6, 2011.
[41] P. Musialski, M. Wimmer, and P. Wonka, "Interactive Coherence-Based Façade Modeling," *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2012)*, no. 2, 2012.
[42] J. Cech and R. Sara, "Windowpane detection based on maximum aposteriori probability labeling," in *International Workshop on Combinatorial Image Analysis*, Buffalo, NY, USA, 2008.
[43] ——, "Languages for constrained binary segmentation based on maximum a posteriori probability labeling," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 69–79, Jun. 2009.
[44] L. Simon, O. Teboul, P. Koutsourakis, L. Van Gool, and N. Paragios, "Parameter-free/Pareto-driven Procedural 3D Reconstruction of Buildings from Ground-Level Sequences," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Providence, USA: IEEE, 2012.

Olivier Teboul graduated from Ecole Centrale Paris in 2007 in Applied Mathematics, received a M.Sc from Ecole Normale Superieur de Cachan in 2007 and a Ph.D from Ecole Centrale Paris in 2011 in Applied Mathematics. He is software engineer at Google Brazil since 2011.

Iasonas Kokkinos (S.M. 2002, M. 2006) obtained the Diploma of Engineering in 2001 and the Ph.D. Degree in 2006, both from the School of Electrical and Computer Engineering of the National Technical University of Athens in Greece. In 2006 he joined the Center for Image and Vision Sciences in the University of California at Los Angeles as a postdoctoral scholar. As of 2008 he is an Assistant Professor at the Department of Applied Mathematics of Ecole Centrale Paris and is also affiliated with the Galen group of INRIA-Saclay in Paris. His research interests are in the broader areas of computer vision, signal processing and machine learning, while he has worked on nonlinear speech processing, biologically motivated vision, texture analysis and image segmentation. His currently research activity is focused on efficient algorithms for object detection, shape-based object recognition and learning-based approaches to feature detection. He has been awarded a young researcher grant by the French National Research Agency, and serves regularly as a reviewer for all major computer vision conferences and journals; he has served as an area chair for CVPR 2012, co-organized POCV 2012 and is an associate editor for the Image and Vision Computing Journal.

Loic Simon received a B.Sc from Ecole Normale Superieure de Cachan (ENS Cachan) in Mathematics in 2004, a B.Sc from ENS de Cachan in Physics in 2004, his aggregation in Mathematics in 2006 a M.Sc in applied Mathematics in 2007. He received a Ph.D from Ecole Centrale Paris in 2011 in Applied Mathematics. He is a post-doc in ENS Cachan since 2011.

Panagiotis Koutsourakis received a M.Sc from University of Crete in 2006.

Nikos Paragios is professor of Applied Mathematics and Computer Science, director of the Center for Visual Computing of Ecole Centrale de Paris and Ecole des Ponts - ParisTech, member of the Laboratoire d'informatique Gaspard-Monge and scientific leader of GALEN group of Ecole Centrale de Paris/INRIA Saclay, Ile-de-France. Professor Paragios is an IEEE Fellow, has co-edited three books, published more than two hundred papers in the most prestigious journals and conferences of medical imaging and computer vision, and holds twenty one US patents. He has served/serves as an associate/area editor/member of the editorial board for the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), the Computer Vision and Image Understanding Journal (CVIU), the International Journal of Computer Vision (IJCV), the Medical Image Analysis Journal (MedIA), the Journal of Mathematical Imaging and Vision (JMIV), the Imaging and Vision Computing Journal (IVC), the Machine Vision and Applications (MVA) Journal and the SIAM Journal in Imaging Sciences (SIIMS) and regularly serves as an area chair for the top conferences of medical imaging and computer vision conferences (ICCV, CVPR, ECCV, MICCAI,...). Professor Paragios is member of the scientific council of SAFRAN conglomerate. His research interests are in medical imaging and computer vision.