Image Parsing with Graph Grammars and Markov Random Fields Applied to Facade Analysis

Mateusz Koziński and Renaud Marlet
Université Paris-Est, LIGM (UMR CNRS 8049), ENPC
F-77455 Marne-la-Vallée
mateusz.kozinski@enpc.fr, renaud.marlet@enpc.fr

Abstract

Existing approaches to parsing images of objects featuring complex, non-hierarchical structure rely on exploration of a large search space combining the structure of the object and positions of its parts. The latter task requires randomized or greedy algorithms that do not produce repeatable results or strongly depend on the initial solution. To address the problem we propose to model and optimize the structure of the object and position of its parts separately. We encode the possible object structures in a graph grammar. Then, for a given structure, the positions of the parts are inferred using standard MAP-MRF techniques. This way we limit the application of the less reliable greedy or randomized optimization algorithm to structure inference. We apply our method to parsing images of building facades. The results of our experiments compare favorably to the state of the art.

1. Introduction

We address the problem of semantic segmentation of images with a prior knowledge of the structure of pictured objects. We assume the structure is highly variable, but the patterns of variation are known, so that it is possible to generate any structure we may encounter in the data. Building facades feature variability of this kind, as demonstrated by the work on the generation of building models [15] and facade parsing [14, 23]. Such objects are often modeled by grammars expressing both the structure of the model and the position of its elements [7, 12, 20, 24]. However, inference in these models requires a randomized [24] or greedy [7] methods of exploring the solution space, which cannot be relied on to repeatably produce optimal results.

In this paper we propose: a representation of an object that separates structure from positions of parts, a method to optimize the positions of parts for a given structure and a method to explore the space of model structures. The benefit of this approach is that the application of the less reliable randomized or greedy optimization algorithm is limited to structure inference, while a more effective technique is applied to optimizing positions of the parts. We also introduce a novel energy fusion scheme for combining detections with texture classification. We evaluate experimentally the influence of each of the contributions on parsing performance.

1.1. Related Work

The problem of grammar-based facade analysis has recently received a considerable attention in the computer vision community [13, 16, 17, 23]. The aim is to segment an input image in such a way that the result belongs to the language generated by a given shape grammar. The top-down parser of Teboul *et al*. [23, 24] draws from the idea of split grammars for architectural modeling [15]. Generation of a facade model from the grammar is analogous to derivation of a string or sentence in formal language processing. The goal is to generate a model that corresponds as much as possible to visual cues in the input image. Simultaneous optimization of the sequence of production rules and positions of parts requires a randomized exploration of a large space of possible 'parses'. Such approach does not produce repeatable results: in [23] the optimization is run five times and the best of the five solutions is kept as the final result.

Grammar-free methods for facade segmentation [3, 14] can accommodate a variety of object classes and feature high rates of correctly classified pixels. However, they impose constraints on composition of parts only locally. The bottom-up method proposed by Martinovic *et al*. [14] provides good pixel classification but does not guarantee that the resulting segmentation is a 'valid' facade. Artifacts produced by the method are presented in Fig. 6.

A general, rectangle-based parsing algorithm has been proposed by Han and Zhu [7]. Rectangles are terminal symbols of the grammar and the production rules combine them into rows, columns and grids. Also in this case the space of part positions and the space of model structures are treated jointly. The model is greedily composed of the strongest rectangle candidates detected in the image or generated top-

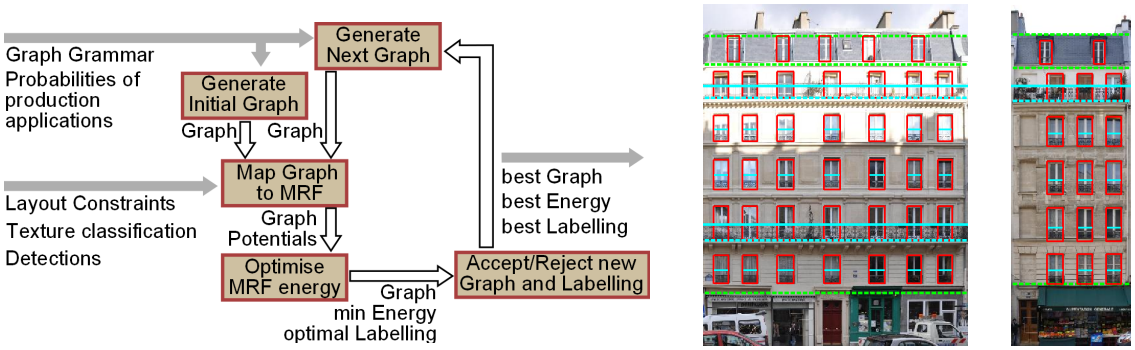

Figure 1: A diagram of our parsing algorithm *(left)*. Results of facade parsing using our approach *(right)*. The green snaplines separate the sky, roof, wall and shop classes. The cyan lines mark borders of running and single-window balconies.

down. Position of a candidate in the image is fixed.

Our work is related to the part-based approach to object modeling, that has proven effective in numerous applications including face detection and pose tracking [4, 5]. Although we represent the imaged objects as factor graphs, the graphs are not fixed but generated from a graph grammar. They also feature a highly connected and loopy structure.

Grammars and part-based models have been combined in application to object detection in [6, 19]. The grammars are hierarchical: position of a part is dependent on the position of its 'parent' object and the applied production rule. Unfortunately, the structure of certain objects cannot be organized in a purely hierarchical manner. In the case of facades, windows are aligned horizontally within floors and vertically between different floors. A tree-shaped hierarchy would fail to preserve alignment in one of the directions.

1.2. Our Approach

We model a complex object as a factor graph. We specify a set of models using a graph grammar. A factor graph derived from the grammar represents a structure of an object. The 'variable nodes' correspond to geometric primitives. The 'factor nodes' can correspond to constraints on the composition of the primitives, or to parts (basic objects) defined by a number of geometric primitives. An example factor graph is presented in Section 2. The graph grammar paradigm is presented in Section 3.

A factor graph is transformed into a Markov Random Field (MRF). Variable nodes are assigned variables defining positions of the geometric primitives in space. The factors are assigned potentials that penalize violation of the constraints or evaluate hypotheses of the positions of parts against image cues. The potentials are defined in Section 4.

The minimum energy labeling of the MRF corresponds to positions of the parts that best agree with image cues. The minimum energy is a measure of fitness of the graph to the image. Our algorithm (see the overview in Fig. 1) generates a candidate graph by perturbing the sequence of grammar productions used to generate the current graph. The candidate structure is accepted if the minimum energy of the corresponding MRF is lower than the one of the current graph. The optimization procedure is detailed in Section 6.

1.3. Contribution

The main novelties of the proposed method include:

1. the modeling scheme where inference over part positions and inference of model structure are separated,
2. the application of a graph grammar to modeling variation of model structure and a method of exploring the structure space,
3. a late fusion scheme for combining texture classification and object detections.

2. Object Model

We model parts of an object and their relations as a factor graph. A labeled factor graph $G = (V, F, \mathcal{E}, \lambda^{\mathrm{v}}, \lambda^{\mathrm{f}}, \mathcal{L}^{\mathrm{v}}, \mathcal{L}^{\mathrm{f}})$ consists of a set of variable nodes V, a set of factor nodes F, a set of directed edges $\mathcal{E} \subset (V \times F) \cup (F \times V)$ and functions $\lambda^{\mathrm{v}} : V \to \mathcal{L}^{\mathrm{v}}$, and $\lambda^{\mathrm{f}} : F \to \mathcal{L}^{\mathrm{f}}$, assigning labels $l_v \in \mathcal{L}^{\mathrm{v}}$, and $l_f \in \mathcal{L}^{\mathrm{f}}$ to the nodes and factors.

Each variable node $v \in \mathcal{V}$ represents a variable x_v encoding position of a geometric primitive. The set of factors $\mathcal{F} = \mathcal{F}_o \cup \mathcal{F}_c$, where $\mathcal{F}_c$ and $\mathcal{F}_o$ are the sets of constraint- and object-type factors. A constraint on a relative position of a pair of primitives is expressed by a constraint factor $f_c \in \mathcal{F}_c$, connected to the pair of corresponding variable nodes. A part of the model, defined by several geometric primitives, is represented by an object factor $f_o \in \mathcal{F}_o$, connected to the variable nodes encoding positions of the primitives. Labels assigned to object factors $\lambda^{\mathrm{f}}(f_o)$ correspond to classes of objects represented by the factor. The set of object classes $C \subset \mathcal{L}^{\mathrm{f}}$ can be defined as $C = \{\lambda^{\mathrm{f}}(f_o) | f_o \in \mathcal{F}_o\}$.

Our approach, applied to facade modeling, is illustrated in Fig. 2. To easily accommodate alignment, non-overlap and adjacency of facade elements we assume a line-based

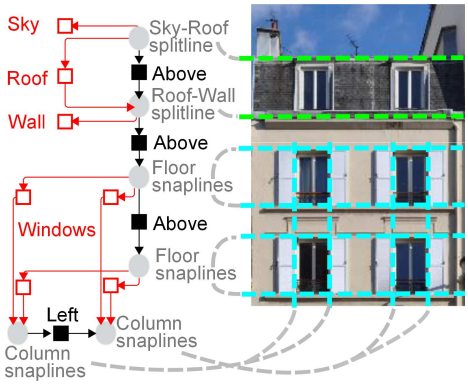

Figure 2: A simplified facade model and its interpretation. The grey disks are **variable nodes**, the rectangles represent factors: the black ones model **constraints** and the red ones represent **parts** of the model (the labels for window factors have been omitted for readability). The dashed lines in the image represent the **splitlines** (green) and the **snaplines** (cyan) encoded by variable nodes.

model. The variable nodes encode positions of splitlines and snaplines (nodes *floor* and *column* represent pairs of lines). The factor nodes correspond to objects (*sky*, *roof*, *wall*, *window*) and constraints (*above* and *left*). The *sky* and *wall* factors are attached to a single splitline each because they extend from the splitline to image boundary.

3. Graph Grammars

Our algorithm generates models of the type presented in Section 2 from a graph grammar. We use the Neighborhood Controlled Embedding grammars of graphs with edge labels (edNCE grammars). Below we review the mechanism briefly. For detailed explanation the reader is referred to [18] and to [8] for the hypergraph case. For consistency with Section 2, we present the paradigm for factor graphs, *i.e.*, we treat factors the way (hyper-) edges are treated in the literature.

A grammar $\mathcal{G} = (S, \mathcal{L}_T^{\mathsf{v}}, \mathcal{L}_N^{\mathsf{v}}, \mathcal{L}^{\mathsf{e}}, \mathcal{P})$ consists of a starting factor graph S, sets of terminal and nonterminal variable node labels $\mathcal{L}_T^{\mathsf{v}}$ and $\mathcal{L}_N^{\mathsf{v}}$, a set of factor labels $\mathcal{L}^{\mathsf{e}}$ and a set of productions $\mathcal{P}$. The starting graph S consists of a single nonterminal node. Productions are of the form $\rho : l_v \to (G^r, C)$, where l_v is a nonterminal node label, called 'mother node label', G^r is a labeled graph, called 'daughter graph', and C is a set of connection instructions. The production can be applied to a node v of the graph G if $\lambda^{\mathsf{v}}(v) = l_v$. First, a graph $G^{r\prime}$, isomorphic to G^r, is inserted into G as a disjoint subgraph. Then, the nodes of $G^{r\prime}$ are connected to neighbors of v in G through factors created according to connection instructions C. We use connection instructions of the form $c_i = (l_e/l'_e, n)$, where l_e and l'_e are factor labels, and n identifies a node in G^r. For each $c_i = (l_e/l'_e, n) \in C$, each factor connected to v and labeled l_e is copied into G. The copy is disconnected from v, reconnected to the node in $G^{r\prime}$ identified by n, and relabeled to l'_e. Finally, node v is removed from G together with all its factors. Examples of production applications are illustrated in Fig. 3.

3.1. Deriving Graphs from a Grammar

A derivation consists of repeated applications of productions until there are no nodes with nonterminal labels in G. A sequence of production applications can be arranged in a derivation tree $\mathbf{z}$, in which nodes correspond to production applications $t = (\rho, n)$, where ρ identifies the production and n identifies the expanded node. A parent-child link between nodes $t = (\rho, n)$ and $t' = (\rho', n')$ in the tree means that production ρ' is applied to a node inserted to the graph by ρ and identified by n' in its daughter graph.

3.2. Energy of a derivation

To express a prior on graphs derived from the grammar we define an energy of the derivation tree $E_{\text{struct}}(\mathbf{z})$ as a sum of energies of individual production applications t. In assigning energies to individual production applications, we would like to distinguish between different nonterminal nodes with the same label. In the facade modeling example from Fig. 3, applications of productions ρ_3 and ρ_4 to the two *columns* nodes should not be evaluated in the same manner because the interpretation of the nodes is different: the first one corresponds to the first window column, the second one to the second. To unambiguously identify a nonterminal node substituted by production application t, we use a sequence of its ancestors in the derivation tree, denoted by $anc(t)$. The energy becomes

$$E_{\text{struct}}(\mathbf{z}) = \sum_{t \in \mathbf{z}} E_{prod}(t, anc(t)) \,. \tag{1}$$

The interpretation of energy of an individual production application $t = (\rho, n)$ is $E_{prod}(t, anc(t)) = -\log p_{n,anc(t)}(\rho)$, the log-likelihood of choosing production ρ out of all alternatives applicable to the nonterminal node identified by $(n, anc(t))$. The likelihoods can be estimated from ground truth parse trees associated to training data.

4. Fitness Energy

A factor graph derived from a grammar defines the number and type of objects present in the scene and the relations between them. The precise positions of the objects and geometric primitives are defined by variables x_v associated with the nodes of the model. We denote a concatenation of all the parameters x_v by $\mathbf{x}$. In order to evaluate how well the model explains the input image I we define an energy function $E_{pos}(\mathbf{x})$, evaluating positions of the parts.

The energy is defined as a sum of potentials assigned to the factors of the graph, forming a Markov Random Field

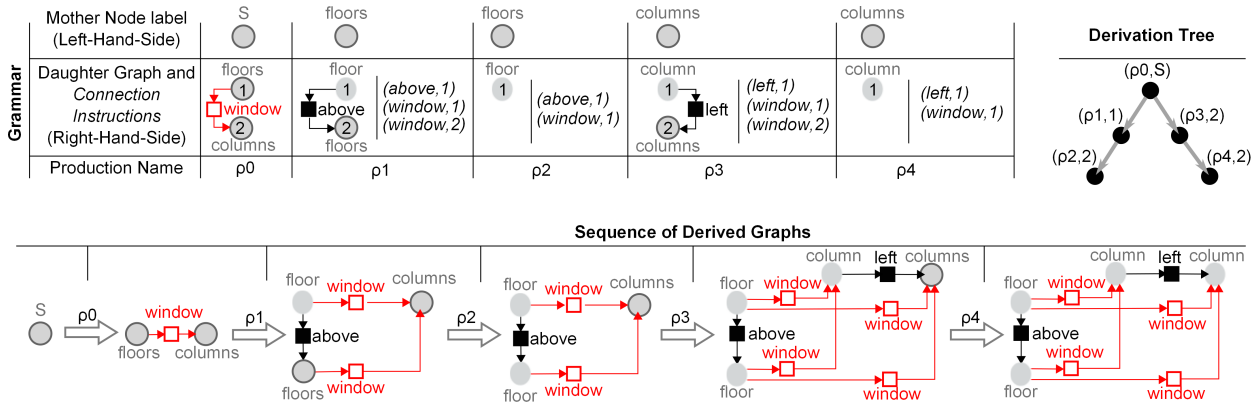

Figure 3: A toy edNCE grammar (*top-left*), its application to derive a graph modeling a grid of windows (*bottom*) and the corresponding derivation tree (*top-right*). The numbers assigned to nodes of the daughter graphs are used by the connection instructions (l_e, n), which say that an edge labeled l_e should be reconnected to a node n in the daughter graph of the production. In the presented grammar productions do not relabel edges, thus the short notation for connection instructions: (l_e, n) instead of $(l_e / l_e, n)$. The outlined disks represent nonterminal nodes; the other disks correspond to terminals, *i.e.*, they are not rewritten. See caption of Fig. 2 for the interpretation of remaining symbols.

(MRF). To each factor f we assign a potential $\phi_f(x_f)$ defined in terms of the variables of neighboring nodes $x_f = (x_1, \ldots, x_n)$. Different types of factors are assigned different potentials. Constraint-type factors $f \in \mathcal{F}_c$ receive potentials ϕ_f^c that penalize violation of predefined constraints on relative position of geometric primitives. Object-type factors $f \in \mathcal{F}_o$ receive potentials ϕ_f^o evaluating object bounding boxes defined by the involved geometric primitives. The energy is a sum of the potentials:

$$E_{pos}(\mathbf{x}) = \sum_{f \in \mathcal{F}_o} \phi_f^o(x_f) + \sum_{f \in \mathcal{F}_c} \phi_f^c(x_f)\,. \qquad (2)$$

4.1. Object-type Potentials

We use two kinds of bottom-up cues for measuring fitness of the model to an input image: texture classification and object detections. Texture classification is given in the form of an energy value $E_{\text{pix}}(I_p|c)$ for each point p belonging to image I and of color I_p, and for each class of parts $c \in C$. The energy $E_{\text{pix}}(I_p|c)$ measures the log-likelihood of observing value I_p if the pixel represents an object of class c. Texture classification is available for each class of parts. Detections might be available for a subset of the classes $C_{\text{det}} \in C$ only. They have the form of bounding boxes y_i with weights w_i. Recall that a class of an object represented by a factor f is encoded in its label l_f. Object factors are expressed in terms of texture and detection-based energies E_t and E_d as:

$$\phi_f^o(x_f) = \begin{cases} \alpha_t E_t(x_f, l_f) + \alpha_d E_d(x_f, l_f) & \text{if } l_f \in C_{\text{det}} \\ E_t(x_f, l_f) & \text{otherwise,} \end{cases} \qquad (3)$$

where α_t and α_d are constant coefficients. We call the above scheme 'late fusion', because it uses detections directly in the energy formulation, contrary to the 'early fusion' [14, 16, 17, 23], where detections are projected down to pixel level and energy is evaluated on individual pixels only. Together with the detection-based potential presented below, late fusion prevents the effect of 'blurring' detections by early fusion, presented in Fig. 4. The experimental comparison of the two approaches is presented in Section 7.

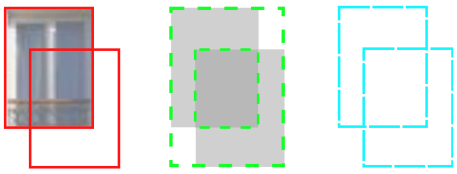

Figure 4: The advantage of late fusion against early fusion. *Left*: a true positive and an inaccurate detection (red boxes); *Middle*: a confidence map produced by early fusion and possible optimal object hypotheses, depending on parameters (green boxes). *Right*: possible optimal object hypotheses for late fusion and our detection-based potentials.

Estimating the coefficients α_t and α_d requires knowledge of the true model structures and part positions for images of the training set, and can be formulated as max-margin MRF training (see, *e.g.*, [10]).

Detection-based Potentials. We need the energy function to be robust to missing detections and false positives, and to rank object hypotheses based on bounding boxes of individual detections to avoid the effects presented in Fig. 4. Given a function $d(x_f, y_i)$, evaluating a distance between an object bounding box defined by vector x_f and a detection y_i of weight w_i, the detection-based potential is expressed by the distance to the 'best' detection in its neighborhood, $\min_i(d(x_f, y_i) - C_{\text{w}} w_i)$, or by a constant penalty C_{d} if there are no nearby detections. The term $C_{\text{w}} w_i$ increases the radius of influence of detections depending on

weights w_i. The detection-based potentials take the form:

$$E_d(x_f) = \min\{\min_{i \in D_f}(d(x_f, y_i) - C_\mathrm{w} w_i), C_\mathrm{d}\}\,, \qquad (4)$$

where D_f is a set of indexes of detections of class l_f. We let the potentials take negative values for good object hypotheses, so that when comparing models with two different structures the model with missing hypotheses gets a higher energy. A possible interpretation of C_d is the distance after which the detection does not influence the potential and the term $C_\mathrm{w} w_i$ is the 'extra' influence range given to a detection depending on its weight. We currently adjust these parameters manually.

Texture-based Potentials. We define the texture-based energies E_t as in [24]. For a factor f the energy is evaluated over the region $B(x_f)$ of the corresponding object:

$$E_t(x_f, l_f) = \sum_{p \in B(x_f)} E_\mathrm{pix}(I_p, l_f)\,. \qquad (5)$$

where l_f is the factor label, encoding its class.

Objects positioned on the background of other objects, like windows on a wall in Fig. 2, are treated specially. We denote the class of the background object associated with factor label l_f as $c^\mathrm{bg}(l_f)$. The potential of the background object is evaluated over its whole bounding box (including the regions of the foreground object) and the potential of the foreground object becomes:

$$E_t(x_f, l_f) = \sum_{p \in B(x_f)} E_\mathrm{pix}(I_p, l_f) - E_\mathrm{pix}(I_p, c^\mathrm{bg}(l_f))\,, \qquad (6)$$

so that the sums of background potentials over the foreground region cancel out when the potentials are added together.

4.2. Composition Potentials

In general, the potentials ϕ^c that express priors on composition of parts can take arbitrary forms. We use them to enforce a set of predefined constraints $g_f(x_f)$:

$$\phi^c_f(x_f) = \begin{cases} 0 & \text{if} \quad g_f(x_f) = \text{true}\,, \\ \infty & \text{otherwise}\,. \end{cases} \qquad (7)$$

The particular types of constraints that we use in our experiments are presented in Section 7.

5. Total Energy

The total energy of a model is a combination of the derivation energy, evaluating the structure of the model, and the fitness energy, evaluating the positions of parts:

$$E(\mathbf{x}, \mathbf{z}) = \alpha_{struct} E_\mathrm{struct}(\mathbf{z}) + \alpha_{pos} E^{\mathbf{z}}_{pos}(\mathbf{x})\,. \qquad (8)$$

We denote the position energy $E^{\mathbf{z}}_{pos}$ with the superscript $\mathbf{z}$ to emphasize that it is defined for a given structure.

Given the optimal energy values for $E_\mathrm{struct}(\mathbf{z})$ and $E^{\mathbf{z}}_{pos}(\mathbf{x})$ for the true and perturbed structures of the ground truth images, the coefficients α_{struct} and α_{pos} can be estimated by max-margin training.

6. Optimization

We propose to employ different methods for minimizing the energy over the spaces of possible structures $\mathbf{z}$ and part positions $\mathbf{x}$. Optimization over the structure is inherently an ill-posed problem and requires a randomized [20, 23] or greedy [7, 24] exploration strategy. On the other hand, although inferring optimal $\mathbf{x}$ for a given factor graph is known to be NP-hard, there exists a number of approaches experimentally proven to perform well in this task (for example [9, 11, 21]). Therefore, we propose to limit the application of the algorithm proposed by Teboul *et al.* [24] to structure exploration and use a state-of-the-art solver [9] to optimize over $\mathbf{x}$ for a fixed $\mathbf{z}$.

Given a derivation tree $\mathbf{z}^j$, the algorithm of [24] generates a candidate tree $\mathbf{z}^\mathrm{cand}$, sets $\mathbf{z}^{j+1} := \mathbf{z}^\mathrm{cand}$ if $\min_\mathbf{x} E(\mathbf{x}, \mathbf{z}^\mathrm{cand}) < \min_\mathbf{x} E(\mathbf{x}, \mathbf{z}^j)$, otherwise sets $\mathbf{z}^{j+1} := \mathbf{z}^j$. This greedy technique differs from a Metropolis-Hastings random walk (see, *e.g.*, [2]) in that it never accepts a candidate of lower energy. To generate $\mathbf{z}^{j+1}$ given $\mathbf{z}^j$, we randomly select a production application in the derivation tree and resample the production application and all the descending ones. The confluence of our grammars lets us keep the remaining part of the derivation. This enables both global and local changes to the structure, *e.g.*, altering the number of columns without changing the number of floors.

Optimization over positions of the parts $\mathbf{x}$ is performed by means of the TRW-S algorithm [9]. It solves the dual to a linear relaxation of the minimum energy MRF labeling problem. We exploit the fact that the dual objective is a lower bound on minimum primal energy by stopping the optimization as soon as it exceeds the value of the best energy attained so far.

7. Experiments

We evaluate our algorithm in the task of parsing rectified photographs of building facades. In a number of experiments, we prove the advantage of separating the structure inference from inference of positions of parts, compare the performance of our algorithm to that of a state-of-the-art parser, and compare the performance of late and early fusion. In the first three experiments we use a Haussmannian facade grammar described in the following subsection.

We use **models of Haussmannian buildings** of the type presented in Fig. 5. Each model contains a variable node encoding the position of each of the following geometric

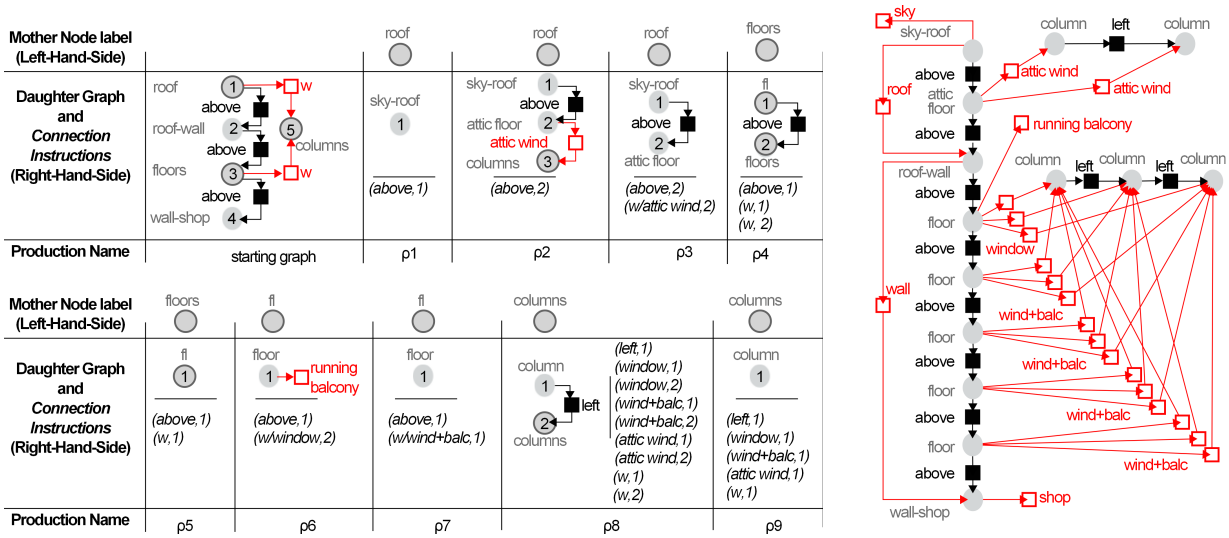

Figure 5: *Left*: the grammar of Haussmannian facades. S is the starting symbol of the grammar. Productions ρ_1, ρ_2 and ρ_3 express different patterns of attic window alignment. Productions ρ_4, ρ_5, ρ_6 and ρ_7 recursively encode a variable number of floors with two possible types of balconies: a running balcony and a balcony limited to a single window. Productions ρ_8 and ρ_9 encode a variable number of window columns. Factors labelled *w* can be interpreted as 'general' windows - they are specialized to *window* or *wind+balc* factors by productions ρ_6 and ρ_7. Connection instructions that do not relabel edges are presented in the short form (l_e, n). *Right*: a model of the facade presented on the right of Fig. 1. See the captions of Fig. 2 and 3 for an explanation of graphical symbols.

primitives: a *sky-roof* line (x_{sr}), a *roof-wall* line (x_{rw}) and a *wall-shop* line (x_{ws}). Additionally, the models contain a certain number of *floor* nodes encoding the positions of the window top, window bottom and balcony top snaplines (x_t, x_b, x_{tb}), and *column* nodes encoding the positions of the window left and window right snaplines (x_l, x_r). Parts of the facade are encoded as object factors, defined in terms of the geometric primitives. For example, a *window* factor encodes a window on a floor with a running balcony and a *wind+balc* factor encodes a window with a small balcony limited to the window cavity.

The constraints for relative positions of the geometric primitives are expressed by potentials of constraint-type factors (Equation 7). They are summarized in Table 1. The constants in the constraints represent ranges for relative sizes of basic parts. The potentials penalize segmentations which clearly violate our concept of a 'valid' facade. For instance, the constraint on the maximum gap between two consecutive floors prevents having a floor missing due to unusual appearance or occlusions. The method is insensitive to moderate changes of the values. We set them manually on the basis of relative sizes of ground truth objects in the training images, although they could easily have been estimated automatically.

The **grammar** we use for Haussmannian buildings is presented in Fig. 5. It encodes three different variations of window layout: i) there are no attic windows, ii) the attic windows are aligned with the windows on the facade, iii) the attic windows are not aligned with the windows on the facade. The grammar is confluent and enables 'local' derivation of different parts of the graph. In particular, the number of floors, the number of columns and the type of attic are selected independently. This enables local changes to the structure of the graph during optimization.

Factor type	Constraints
sky-roof $\xrightarrow{above}$ *floor*	$x_{sr} \leq x_t$
floor $\xrightarrow{above}$ *roof-wall*	$x_t \leq x_{rw}$
roof-wall $\xrightarrow{above}$ *floor*	$x_t - x_{rw} \leq 0.7\,(x_b - x_t)$ $x_t - x_{rw} \geq 0$
floor $\xrightarrow{above}$ *floor*'	$(x'_b - x'_t) \leq 1.4\,(x_b - x_t)$ $(x_b - x_t) \leq 1.4\,(x'_b - x'_t)$ $(x'_t - x_b) \geq 0.1\,(x'_b - x'_t)$
floor $\xrightarrow{above}$ *wall-shop*	$(x_{ws} - x_b) \leq 0.5\,(x_b - x_t)$ $x_{ws} \geq x_b$
column $\xrightarrow{left}$ *column*'	$(x'_r - x'_l) \leq 1.4\,(x_r - x_l)$ $(x_r - x_l) \leq 1.4\,(x'_r - x'_l)$ $(x'_l - x_r) \leq 6.0\,(x_r - x_l)$ $(x'_l - x_r) \geq 0.5\,(x_r - x_l)$

Table 1: Constraints assigned to the constraint factors. Factors are defined by the labels of nodes to which they are connected and their own labels: *node label 1* $\xrightarrow{factor\ label}$ *node label 2*. See the text for explanation of variable names.

To define the **potentials** of our model, we use a random forest classifier [1] for texture classification, trained as in [24]. Windows are detected with the Viola-Jones classifier [25]. We use squared euclidean distance for detection-based potentials defined in Equation 4. By experimenting with

Ground truth	[23]			[14]		[14]	
Method	[23]	Ours(*)	Ours	[14]	Ours	Early Fusion	Late Fusion
Window	65	**75**	74	**75**	73	62	**73**
Wall	83	86	**88**	88	**91**	**91**	**91**
Balcony	58	66	**73**	70	**71**	67	**71**
Roof	64	60	**69**	74	**80**	**80**	**80**
Sky	87	**91**	**91**	**97**	90	**90**	**90**
Shop/door	**98**	**98**	**98**	**94**	**94**	**94**	**94**

Table 2: Experimental results. The entries correspond to the percentage of correctly classified pixels (diagonal entries of the confusion matrices). *Left*: results of [23] compared to our method, also (*) without the detection term. *Middle*: results of [14] compared to our method. The results for the 'shop/door' class are estimated based on the 'shop' and 'door' entries in the confusion matrices. *Right*: Comparison of the two approaches for fusing texture classification and detection results.

several values, we set $C_{\mathrm{d}} = 16$ and $C_{\mathrm{w}} = 48$. This means that poor detections with weights $w_i \approx 0$ would have a radius of influence equal to 4 pixels in the four-dimensional space of bounding boxes and the best detections with $w_i \approx 1$ influence the potential up to a distance of 8 pixels.

7.1. Implementation

We run a Matlab **implementation** of the algorithm (with an external MRF solver) on an office computer (Intel Core i7 processor, 2.66 GHz). We apply the same train-test scheme as in [14], with 80 training and 20 testing images, repeat the validation 5 times and average the results.

7.2. Separate Structure and Position Inference

To demonstrate the advantage of separating structure and position inference, we compare our algorithm against the method proposed by Teboul *et al*. [22, 23]. The comparison is made on the dataset presented with their paper and against their ground truth. Note that their annotated segments tend to be inaccurate: the ground truth, as well as their grammar, impose the false constraint that the attic windows are necessarily aligned with the windows on the facade image, which can be wrong because of architectural variation, or because both kinds of windows are in different planes and the images are rectified. For a fair comparison with this work, we introduce the same constraint to our grammar. The algorithm of [23] is based on texture classification, so we restrict our energy to texture-related terms. We run our algorithm for 5 minutes per image. The algorithm of Teboul *et al*. is re-run 10 times (instead of 5 times as reported in [22]) for 30 seconds and the best result is kept. As shown in Table 2, for 'easy' objects like shop, sky and wall, the position of which is determined by just a pair of splitlines, the performance is similar. The advantage of the more reliable optimization algorithm shows for the classes with a higher number of interactions, like windows and balconies.

7.3. Performance in Facade Parsing

To estimate the performance in facade parsing we evaluate our algorithm on the dataset of [23] against the ground truth proposed in [14]. Our results are comparable to the ones reported in their paper (cf. Table 2 left) with a slight advantage to our algorithm. The strong benefit of our method is that it guarantees that the resulting segmentations belong to the language generated by the grammar. Artifacts like the ones shown in Fig. 6 are thus avoided.

7.4. Late vs. Early Fusion

We also compare the performance of early and late fusion for combining window detection and texture classification. We assume a general early fusion scheme where a new energy term is defined for each pixel. It is a convex combination of the per-pixel texture and detection energies: $E'_t = E_{\mathrm{pix}}(I_p|c) + E_{\mathrm{det}}(w_p|c)$. The per-pixel detection-based energy E_{det} depends on w_p, the cumulated weight of window detections overlapping pixel p. We set $E_{\mathrm{det}}(w_p|c) = -\log p(w_p|c)$ and estimate the probabilities from the training data. As presented in the last segment of Table 2, parsing results for window and balcony are notably better with late fusion.

8. Conclusions

We have presented a novel method for modeling complex objects, where the model structure and part positions are optimized separately. The validity of this approach has been confirmed experimentally. In another experiment we have shown that the performance of the proposed method in parsing facade images compares favorably to the state of the art. A third experiment shows that the performance is partly due to the proposed scheme of fusing texture classification and object detections. An important advantage of our method over the grammar-free segmentation is that it produces results that belong to the specified language.

The work raises a number of challenges, including efficient bottom-up methods of building the initial structure graph, methods for warm-starting MRF inference after perturbing its structure, modeling 3D objects by means of graph grammars and learning grammars from training data.

Figure 6: Comparison of parsing results. *Odd:* ours, overlaid on the input images. Splitlines are in green, balconies are outlined in cyan and magenta. *Even:* results reported in [14]. Note the artifacts: windows wider than the corresponding balconies, unaligned balconies.

Acknowledgments

This work has been carried out in IMAGINE, a joint research project between École des Ponts ParisTech (ENPC) and the Scientific and Technical Centre for Building (CSTB). It was supported by Agence Nationale de la Recherche (Semapolis project, ANR-13-CORD-0003).

References

[1] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 6

[2] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995. 5

[3] D. Dai, M. Prasad, G. Schmitt, and L. V. Gool. Learning domain knowledge for facade labeling. In *ECCV*, 2012. 1

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61:2005, 2003. 2

[5] M. A. Fischler and R. A. Elschlager. The Representation and Matching of Pictorial Structures. *Computers, IEEE Transactions on*, C-22(1):67–92, Jan. 1973. 2

[6] R. B. Girshick, P. F. Felzenszwalb, and D. A. McAllester. Object detection with grammar models. In *NIPS*, pages 442–450, 2011. 2

[7] F. Han and S.-C. Zhu. Bottom-up/top-down image parsing with attribute graph grammar. *IEEE Trans. PAMI*, 31(1):59–73, 2009. 1, 5

[8] R. Klempien-Hinrichs. Context-free hypergraph grammars with node rewriting. *Electronic Notes in Theoretical Computer Science*, 51:202 – 211, 2002. 3

[9] V. Kolmogorov. Convergent Tree-Reweighted message passing for energy minimization. *IEEE Trans. PAMI*, 28(10):1568–1583, Oct. 2006. 5

[10] N. Komodakis. Efficient training for pairwise or higher order crfs via dual decomposition. In *CVPR*, 2011. 4

[11] N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *IEEE Trans. PAMI*, 33(3):531–552, 2011. 5

[12] P. Koutsourakis, L. Simon, O. Teboul, G. Tziritas, and N. Paragios. Single view reconstruction using shape grammars for urban environments. In *CVPR*, 2009. 1

[13] A. Martinovic and L. V. Gool. Bayesian grammar learning for inverse procedural modeling. In *CVPR*, 2013. 1

[14] A. Martinovic, M. Mathias, J. Weissenberg, and L. Van Gool. A three-layered approach to facade parsing. In *ECCV 2012*. Springer, 2012. 1, 4, 7, 8

[15] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Transations on Graphics*, 25(3):614–623, 2006. 1

[16] D. Ok, M. Kozinski, R. Marlet, and N. Paragios. High-level bottom-up cues for top-down parsing of facade images. In *2nd Joint 3DIM/3DPVT Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, 2012. 1, 4

[17] H. Riemenschneider, U. Krispel, W. Thaller, M. Donoser, S. Havemann, D. Fellner, and H. Bischof. Irregular lattices for complex shape grammar facade parsing. In *CVPR*, 2012. 1, 4

[18] G. Rozenberg, editor. *Handbook of graph grammars and computing by graph transformation: volume I. foundations*. World Scientific Publishing Co., Inc., 1997. 3

[19] Z. Si and S.-C. Zhu. Learning and-or templates for object recognition and detection. *IEEE Trans. PAMI*, 99(PrePrints):1, 2013. 2

[20] 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 *CVPR*, 2012. 1, 5

[21] D. Tarlow, D. Batra, P. Kohli, and V. Kolmogorov. Dynamic tree block coordinate ascent. In *ICML*, 2011. 5

[22] O. Teboul. *Shape Grammar Parsing : application to Image-based Modeling*. PhD thesis, 2011. Ecole centrale Paris. 7

[23] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *CVPR*, pages 2273–2280, 2011. 1, 4, 5, 7

[24] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape priors. In *CVPR*, pages 3105–3112, 2010. 1, 5, 6

[25] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001. 6