Beyond Procedural Facade Parsing: Bidirectional Alignment via Linear Programming

Mateusz Koziński, Guillaume Obozinski and Renaud Marlet

Université Paris-Est, LIGM (UMR CNRS 8049), ENPC
F-77455 Marne-la-Vallée

Abstract. We propose a novel formulation for parsing facade images with user-defined shape prior. Contrary to other state-of-the-art methods, we do not explore the procedural space of shapes derived from a grammar. Instead we formulate parsing as a linear binary program which we solve using Dual Decomposition. The algorithm produces plausible approximations of globally optimal segmentations without grammar sampling. It yields state-of-the-art performance on standard datasets.

1 Introduction

The goal of facade parsing is to segment rectified building images into regions corresponding to architectural elements, like windows, balconies and doors. The resulting segments have to satisfy structural constraints, e.g., alignment of windows on the same floor, or requirement that a balcony is associated to a window and right below it. Applications include creating 3D models of urban scenes.

A common approach to this problem is to let the user specify a shape prior encoding the structural constraints. It often takes the form of a shape grammar and proposed algorithms try to find a sequence of instantiated grammar rules yielding an optimal segmentation [1–3]. But the dimension of the search space is very large. Consequently, these algorithms suffer from the 'curse of structural exploration'. They search the solution space randomly [1, 2], which does not guarantee optimality or repeatability, or severely subsample the image [3].

In this paper we lift the curse of structural exploration by proposing an altenative formulation of priors, which can be mapped to a linear binary program and solved efficiently, yielding state-of-the-art performance on standard datasets.

1.1 Related Work

Most proposed priors that are complex enough to model constraints of building facades rely on shape grammars [4]. The concept has been introduced by Stiny et al. [5] in the 70's, and the idea of representing image contents in a hierarchical and semantized manner traces back to the work of Ohta et al. [6, 7]. Practical applications to image segmentation and interpretation are more recent [8–11].

A grammar is typically given by a set of nonterminal symbols $\mathcal{N}$, a set of terminal symbols $\mathcal{T}$, a start symbol in $\mathcal{N}$, and a set of production rules of the form $A_0 \to A_1 \dots A_n$ where $A_0 \in \mathcal{N}$ and $A_i \in \mathcal{N} \cup \mathcal{T}$ for $1 \le i \le n$.

In the grammar of Han and Zhu [8], terminal symbols are rectangles and production rules combine them into rows, columns or grids, allowing rectangle nesting. The authors resort to a greedy algorithm for constructing the parse tree, which illustrates the difficulty of optimizing over a grammar derivation.

Drawing ideas from architectural modeling [12], where facade generation is analogous to string derivation in formal languages, the top-down parser of Teboul et al. [1, 2] is one of the first attempts to parse facades using 'split grammars'. The input image is recursively split into rectangular subregions which are assigned a class label. Spliting directions as well as the number and class of subrectangles are non-deterministically chosen according to a predefined set of production rules. The process continues until all rectangles have a terminal class. The parser actually samples a number of possible derivations, exploring only a small part of the structural space. Even with a 'smart' sampling strategy [2], it does not produce repeatable results: as reported in [13], inference consists in independently running the exploration five times and keeping the best solution.

To counter the drawbacks of sampling, Riemenschneider et al. [3] propose an adaptation of the Cocke-Younger-Kasami (CYK) algorithm for parsing string grammars to two-dimensional split grammars. Its complexity is $O(w^2h^2N)$, where w and h are image dimensions and N is the number of possible combinations of production rule attributes (including splitting positions). This limits practical applications of the algorithm to grids of about 60 by 60 cells. To circumvent this limitation the authors test different methods of image subsampling.

An attempt to fight the curse of procedural exploration was proposed by Koziński and Marlet [14], using graph grammars and MRF optimization. In contrast to parsers like [2] whose combinatorial search explores both the nature of splits and their position at the same time, sampling here concerns structure only; optimal positions for a given sampled structure are found with a principled and efficient method. The space to explore, which now does not depend on image size, is considerably smaller, but the curse of the procedural space is not eliminated completely as graph-grammar sampling remains.

Some facade segmentation methods [15, 16] do not use any user-defined shape prior. The bottom-up method proposed by Martinovic et al. [15] applies 'soft' architectural principles as a postprocessing step after image segmentation, but cannot accommodate 'hard' structural constraints. It can produce artifacts, like windows extending further than their balconies. A more recent work by Cohen et al. [16] uses a sequence of dynamic programs to recover a segmentation that respects a set of hard-coded constraints and attains state-of-the-art performance on the standard datasets. In our experiments, our method matches the performance of this algorithm while offering full flexibility with respect to shape prior specification.

In this paper we formulate the problem of finding an optimal segmentation as a binary linear program. We solve this program using the Dual Decomposition (DD) approach [17, 18]. Similar techniques include Alternating Direction of Multipliers Method (ADMM) [19]. We chose DD because ADMM, although known

Table 1. Comparison of with state-of-the-art facade parsing methods.

Property	[2]	[15]	[16]	[3]	Ours
User-defined shape prior	✓	–	–	✓	✓
Approximation of global optimum	–	–	–*	✓	✓
No need of image subsampling (for tractability)	✓	✓	✓	–	✓
Simultaneous alignment in two dimensions	✓	✓	–	✓	✓

* Cohen et al. [16] can issue a certificate of optimality if the found solution is optimal.

to feature better convergence properties, requires solving quadratic subproblems. The experiments confirm that DD behaves well in our application.

1.2 Contributions

Our approach for image parsing does not suffer from the curse of procedural exploration. It is based on a shape prior formalism that allows efficient parsing.

Instead of expressing a shape prior using grammar rules, we propose to represent the structural decomposition of a scene as a hierarchy of classes, complemented by a specification of forbidden configurations of neighboring elements.

The parsing problem can then be turned into a linear binary program, which we solve efficiently using Dual Decomposition, eliminating the need for a procedural exploration of the solution space. As shown in the experiment section, our algorithm features the accuracy of methods using hard-coded structural constraints [15, 16] while retaining the flexibility of grammar-based methods [2, 3]. The comparison to state of the art is summarized in table 1.

2 Proposed Model

Although it departs from the grammar-based methods, our approach to structural segmentation is inspired by the process of hierarchical image subdivision into rectangular regions, which we will refer to as rectangles in the rest of the paper. The shape prior consists of a tree of rectangle classes and a specification of pairwise potentials penalizing unlikely or invalid configurations of adjacent rectangles. In the tree, child nodes represent classes of rectangles resulting from splitting a rectangle of a parent class. We require that a rectangle of a class resulting from a vertical split can only be split horizontally, and vice versa. Consequently, all non-leaf nodes at a given tree depth are split along the same direction. Each nonterminal is assigned a table of pairwise potentials penalizing pairs of child classes assigned to neighboring rectangles. Our algorithm can handle infinite values of the potentials and in our experiments we only use binary potentials that take the value of zero or infinity, preventing some configurations of neighbors and allowing the others.

In contrast to split grammars, which are context-free and cannot be used to express simultaneous alignment in two dimensions (other than with implementation tricks that introduce some context dependency [2]), we require rectangles of

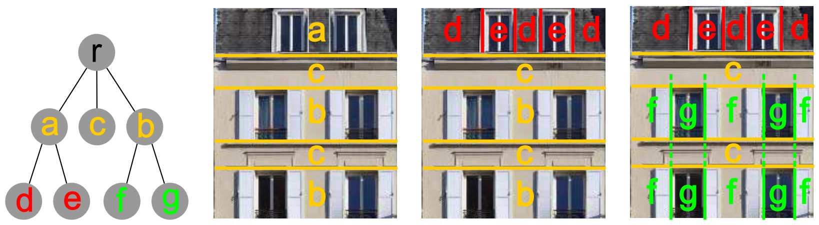


Fig. 1. A shape prior consists of a hierarchy of classes (image 1) and a table of pairwise potentials for each nonterminal node (not shown here). Each image (2-4) shows substitution of all rectangles of a particular class with rectangles of child classes.

the same class to be aligned both vertically and horizontally. This requirement can be enforced by constraining all rectangles of the same class that are aligned along the splitting direction to be split in the same positions into subrectangles of the same classes. A tree example and corresponding segmentations are presented in figure 1. Note the bidirectional alignment of windows (class g).

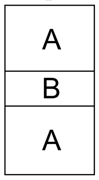
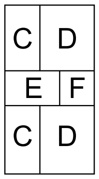
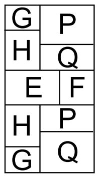
2.1 Optimal Segmentation as a Binary Linear Program

We denote the set of indices of image pixels by $\mathcal{I} = \{(i, j)|i \in I, j \in J\}$, $I = \{1, \dots, h\}$ and $J = \{1, \dots, w\}$, where h is image height and w is image width. We denote the set of rectangle classes by $\mathcal{C} = K \cup L$, where K denotes the set of classes that result from a horizontal split, also called row-classes, and L is the set of classes that result from a vertical split, called column-classes, and $K \cap L = \emptyset$. The root of the tree r is a 'starting class', corresponding to the whole image. Without loss of generality we assume that r is split horizontally and by convention we consider $r \in L$. We recall that nodes in K can only have children in L and vice versa. In consequence, all nodes at a given level of the tree are either col-classes or row-classes. We denote the set of children of class $n \in \mathcal{C}$ by $Ch(n)$ and the set of descendants of n, including n, by $Desc(n)$. Similarly, we denote the set of ancestors of n, including n, by $Anc(n)$, and its parent by $Pa(n)$. The set of siblings of n is denoted $Sib(n)$. We call $t \in \mathcal{C}$ corresponding to the leaves of the tree terminal classes and denote their set by T .

A sequence of vertical and horizontal splits assigns a sequence of rectangle class labels to every pixel of the image. For any row i, it is thus possible to list all the classes that are assigned to at least one pixel on the row. Below we show that a segmentation consistent with a prior of the proposed form can be encoded in terms of the sets of classes assigned to each image row and column. This row- and column-based formulation enables global alignment of distant rectangles of the same class. We define variables $y_{ik}, y_{il}, x_{jk}, x_{jl} \in \{0, 1\}$ such that $y_{ik} = 1$ if k is present in row i and $x_{jl} = 1$ if l appears in column j. We make a distinction between the variables encoding assignment of row-classes $k \in K$ and column-classes $l \in L$, because they behave differently for horizontal and vertical splits.

In table 2 we present how the process of shape derivation changes the sets of row- and column-classes present in image rows and columns, and formulate

Table 2. Illustration of the splitting process and interpretation of the variables x_{jl} and y_{ik}. The splitting process is just a concept that helps us to introduce our formulation and not a mode of operation of the proposed algorithm.

	Example sequence of splits	Specific constraints for the example	General constraints
First split $r \to \{A, B\}$		$x_{jA} = 1$ $x_{jB} = 1$ $y_{iA} + y_{iB} = 1$	$\forall k \in Ch(r),\ x_{jk} = 1$ $\sum_{k \in Ch(r)} y_{jk} = 1$
Vertical splits $A \to \{C, D\}$ $B \to \{E, F\}$		$x_{jC} + x_{jD} = x_{jA}$ $x_{jE} + x_{jF} = x_{jB}$ $y_{iA} = y_{iC} = y_{iD}$ $y_{iB} = y_{iE} = y_{iF}$	$\forall k \in K,\ \sum_{l \in Ch(k)} x_{jl} = x_{jk}$ $\forall k \in K, \forall l \in Ch(k),\ y_{jl} = y_{jk}$
Horiz. splits $C \to \{G, H\}$ $D \to \{P, Q\}$		$x_{jC} = x_{jG} = x_{jH}$ $x_{jD} = x_{jP} = x_{jQ}$ $y_{iC} = y_{iG} + y_{iH}$ $y_{iD} = y_{iP} + y_{iQ}$	$\forall l \in L, \forall k \in Ch(l),\ x_{jk} = x_{jl}$ $\forall l \in L,\ \sum_{k \in Ch(l)} y_{ik} = y_{il}$

constraints on y_{ik}, y_{il}, x_{jk} and x_{jl} that reflect this behaviour. As shown in the second row of the table, a vertical split of a rectangle of parent class results in a number of rectangles of child classes. Because the split is along the vertical axis, only one child rectangle is going to appear in each image column previously occupied by the parent. However, all children are going to occur in each image row where the parent was present. We emphasize that all vertically aligned rectangles of the same class are split simultaneously along the same lines, so that the child rectangles are aligned and their classes are consistent along the splitting axis. The same reasoning applies to horizontal splits.

The corresponding constraints on x_{jl}, x_{jk}, y_{il} and y_{ik} are presented in the third column of table 2. We note that for vertical splits the state of each y_{il} for $l \in Ch(k)$ is determined by y_{ik} and that the same holds for horizontal splits, x_{jk} and x_{jl}, as shown in the fourth column of table 2. We therefore eliminate the redundant variables x_{jk} and y_{il}. This will result in a formulation where row-classes are assigned to image rows and col-classes are assigned to image columns. We combine the two first equations and the two second ones from rows two and three of the table to get

$$\forall i \in I, \forall l \in \mathring{L}, \sum_{k' \in Ch(l)} y_{ik'} = y_{iPa(l)}, \quad \forall j \in J, \forall k \in \mathring{K}, \sum_{l' \in Ch(k)} x_{jl'} = x_{jPa(k)}, \quad (1a)$$

where $\mathring{L} = L \setminus (T \cup \{r\})$ and $\mathring{K} = K \setminus T$. We visualize the domain of the constraints in fig. 2.

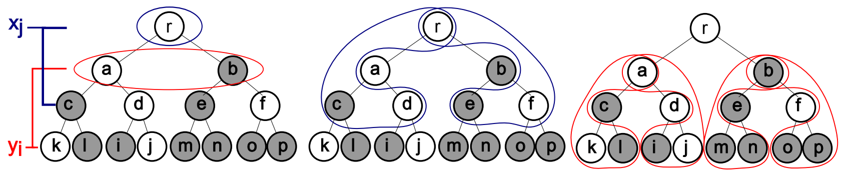


Fig. 2. Visualization of the state of variables y_{ik} and x_{jl} for some pixel i, j. The white nodes correspond to classes k and l for which $y_{ik} = 1$ and $x_{jl} = 1$. The gray nodes correspond to classes with $y_{ik} = 0$ or $x_{jl} = 0$. The domains of constraints (1) on y_{ik} are circled in blue and the domains of constraints on x_{jl} are circled in red. Left: the domains of (1b). Middle and right: the domains of (1a). Note that only one leaf is connected to the root by a path of white nodes. This illustrates the uniqueness of pixel class given the state of variables corresponding to its row and column.

In the interest of maintaining the convention of assigning row-classes $k \in K$ to rows and column-classes $l \in L$ to columns, we modify the constraint from the first row of the table. We require that the root class is assigned to each column and that the first horizontal split assigns a unique class to each row:

$$\forall j \in J, x_{jr} = 1\ , \quad \forall i \in I, \sum_{k \in Ch(r)} y_{ik} = 1\ . \tag{1b}$$

From table 2 it is evident that at each stage of the splitting process each pixel is assigned a unique class. Below we show that constraints (1) also capture this property and that the class assigned to pixel (i, j) is unambiguously determined by vectors (y_{ik}) for given i and (x_{jl}) for a fixed j.

Lemma 1. *Consider a hierarchy of classes given as a tree, as defined earlier. Denote the depth of the tree by M, the set of column-classes at the m-th level of the tree by L^m and the set of row-classes at the m-th level of the tree by K^m. Note that L^m is nonempty only for even m and K^m for odd m. Denote the vectors of y_{ik} and x_{jl} by $\mathbf{y}$ and $\mathbf{x}$. Denote the set of $\mathbf{y}$ and $\mathbf{x}$ satisfying constraint (1) by C_h. Then*

$$(\mathbf{y}, \mathbf{x}) \in C_h \implies \quad \forall (i,j) \in \mathcal{I} \quad \forall m \in \{0, \dots, M\},$$
$$\exists! l_j^m \in L^m : \quad \forall n \in Anc(l_j^m), \quad (x_{jn} = 1) \vee (y_{in} = 1) \quad \text{if } m \text{ is even} \tag{2}$$
$$\exists! k_i^m \in K^m : \quad \forall n \in Anc(k_i^m), \quad (x_{jn} = 1) \vee (y_{in} = 1) \quad \text{if } m \text{ is odd}\ . \tag{3}$$

In words, for any pixel $(i, j) \in \mathcal{I}$, for any values of variables y_{ik} and x_{jl}, that satisfy constraints (1), at any depth of the tree there exists exactly one row-class, or one column-class such that the variables x_{jl} and y_{ik} corresponding to the class and all its ancestors are equal to one.

Proof. We prove the lemma by induction on the depth of the tree.

The root r is the only node at depth $m = 0$ of the tree and, by constraint (1b), it holds that $x_{jr} = 1$ for all $j \in J$. Therefore the lemma holds for $m = 0$.

For depth $m = 1$ the tree is formed of the root and its children. By constraints (1b), we have that for each i there exists a single $k_i \in Ch(r)$ such that $y_{ik_i} = 1$ and $y_{ik} = 0$ for $k \neq k_i$. This proves the lemma for the case of a tree of depth 1.

Assume lemma 1 holds at depth m. If the m is even, then by assumption for each j we have a single l_j^m such that the variables associated to all its ancestors are equal one. By constraint (1a), exactly one child of l_j^m will have its associated variable $y_{ik_i^m}$ equal to one. Similar reasoning applies to odd levels of the tree. $\square$

We model the assignment of terminal classes to pixels by variables $z_{ijt} \in \{0, 1\}$, where $z_{ijt} = 1$ if pixel (i, j) is of class $t \in T$ and $z_{ijt} = 0$ otherwise. A single terminal class has to be assigned to each pixel

$$\forall (i, j) \in \mathcal{I}, \quad \sum_{t \in T} z_{ijt} = 1 \; . \tag{4}$$

By lemma 1, all ancestors of the class assigned to pixel (i, j) have the variables y_{ik} and x_{jl} equal to one, which leads to the inequalities

$$\forall (i, j) \in \mathcal{I}, \forall k \in K, \sum_{t \in Desc(k)} z_{ijt} \leq y_{ik}, \quad \forall (i, j) \in \mathcal{I}, \forall l \in L, \sum_{t \in Desc(l)} z_{ijt} \leq x_{jl}. \tag{5}$$

Each nonterminal class has a table of pairwise potentials defined on its children. The potentials determine the likelihood of observing neighboring rectangles of the child classes. We implement the potentials with variables $y_{ikk'}$ and $x_{jll'}$

$$\forall i \in \{1, \ldots, h-1\}, \forall k \in K \sum_{k' \in Sib(k)} y_{ikk'} = y_{ik} \; , \tag{6a}$$

$$\forall i \in \{1, \ldots, h-1\}, \forall k' \in K \sum_{k \in Sib(k')} y_{ikk'} = y_{i+1k'} \; , \tag{6b}$$

$$\forall j \in \{1, \ldots, w-1\}, \forall l \in L \sum_{l' \in Sib(l)} x_{jll'} = x_{jl} \; , \tag{6c}$$

$$\forall j \in \{1, \ldots, w-1\}, \forall l' \in L \sum_{l \in Sib(l')} x_{jll'} = x_{j+1l'} \; . \tag{6d}$$

We denote the cost of assigning type t to pixel (i, j) by c_{ijt}, and the pairwise cost for column- and row-classes by $c_{kk'}$ and $c_{ll'}$. We define the sets of pairs of row- and column-classes that are siblings in the tree by $Iiblings$ and $Jiblings$. The segmentation task can be formulated as minimizing the following objective

$$E = \sum_{(i,j) \in \mathcal{I}} \sum_{t \in T} z_{ijt} c_{ijt} + \sum_{i=1}^{h-1} \sum_{(k,k') \in SK} y_{ikk'} c_{kk'} + \sum_{j=1}^{w-1} \sum_{(l,l') \in SL} x_{jll'} c_{ll'} \tag{7}$$

subject to constraints (4) to (6).

Algorithm 1 Dual Decomposition

```
$\forall_m \lambda_m^0 \leftarrow 0, \quad n \leftarrow 1$
while not converged do
    $\forall_m \quad \hat{x}_m^n \leftarrow \arg\min E_m(x_m) + (\lambda_m^{n-1})^\intercal x_m$
    $\forall_m \quad \lambda_m^n \leftarrow \lambda_m^{n-1} + \alpha_n(\hat{x}_m^n - \frac{1}{m}\sum_m \hat{x}_m^n)$
    $n \leftarrow n + 1$
end while
$\hat{x} \leftarrow$ GetFinalX$(\hat{x}_m, \lambda_m)$
```

3 Inference

The formulated problem is linear and has a large number of binary variables. We relax the binary domain constraint and let the variables take values within the range $[0, 1]$. We apply dual decomposition to the resulting continuous problem.

3.1 The Dual Decomposition Algorithm

The dual decomposition algorithm is based on the idea of decomposing a difficult problem into a number of 'slave' subproblems that are easy to solve. Given an original problem $\hat{x} = \arg\min \sum_m E_m(x)$, $x \in C$, where C is a feasible set, we construct a number of copies of the variable x, denoted x_m, and couple them by means of a new constraint $x_m = x$. We formulate the dual problem $\max_{\lambda_m} \min_{x,x_m} \sum_m (E_m(x_m) + \lambda_m^\intercal(x - x_m))$, subject to $x_m \in C$, where λ_m is a vector of Lagrange multipliers. The problem is solved using a projected subgradient algorithm. Calculating the subgradient of the dual objective requires solving $\hat{x}_m = \arg\min E_m(x_m) + \lambda_m^\intercal x_m$, subject to $x_m \in C$, separately for each m. The latter minimizations are called slave problems. We present Dual Decomposition in algorithm 1 and refer the reader to [17, 18] for a detailed derivation. We denote the values of variables in iteration n by a superscript and the stepsize by α. The algorithm is run with decaying step size. The values of $\hat{x}_m$ eventually converge and heuristics, represented in algorithm 1 by procedure GetFinalX, can be used to decide on the components of $\hat{x}$ on which $\hat{x}_m$ disagree [17].

The main design decision to be made when applying dual decomposition is how to decompose the original objective function into slave objectives. The main criterion is the ability to efficiently solve the slave problems. Below we present a decomposition of the objective (7) into subproblems that can be solved by means of dynamic programming in time linear in the number of pixels.

3.2 Application of Dual Decomposition to the Problem

To make the slave problem tractable we need to decouple the variables y_{ik} from the variables x_{jl} corresponding to columns. The resulting slaves would assign sets of classes to rows or columns of the image, and terminal classes to pixels.

This decoupling is however not sufficient since feasible configurations of the sets, encoded by vectors (y_{ik}) and (x_{jl}), are determined by constraints (1), that

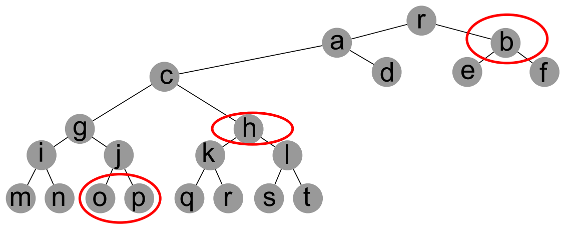


Fig. 3. The structure of set H_l, for $l = j$, visualized on a tree of classes. Elements of the set are outlined in red. Exactly one of them has to be assigned to each image row.

have a complex structure. We propose a further decomposition, that results in a larger number of slaves with simpler constraints. The slaves assign a single class to each pixel and each image row or column.

Each instantiation of constraints (1) can be transformed by recursively plugging its left-hand side to a left-hand side of another equation of type (1) until the resulting sum equals one. Consequently, we get

$$\forall l \in L \setminus T \quad \sum_{k \in H_l} y_{ik} = 1 \ , \quad \forall k \in K \setminus T \quad \sum_{l \in V_k} x_{jl} = 1 \ , \tag{8}$$

where $V_k = Ch(k) \cup [L \cap (Ch(A_k) \setminus A_k)]$, $A_k = Anc(k)$ and $Ch(A_k)$ is the set of all children of all elements of A_k. Informally, V_k is the smallest set containing all children of k and such that if l belongs to V_k, then all siblings of its grandparent do as well. The structure of V_k is illustrated in figure 3. H_k is defined similarly. Note that (8) can be transformed back to (1). It is enough to subtract from constraint (8) for some $l \in L \setminus T$ a constraint of the same type for $l' = Pa(Pa(l))$ to get a constraint of type (1). The reader can verify that on the example from figure 3. Thus, constraints (8) are equivalent to their original form (1).

The advantage of constraint (8) is that it is an intersection of simplex constraints, which entails that the problem can be naturally decomposed into a number of subproblems, one for each $l \in L \setminus T$ and each $k \in K \setminus T$.

3.3 Structure of Slave Subproblem

We create one slave for each $l \in L \setminus T$ and one for each $k \in K \setminus T$. Below we present the structure of a slave subproblem for some l. The slaves for k are created symmetrically. We denote by SH_l the set of pairs of sibling row classes k, k' such that $k, k' \in H_l$. Copies of variables that appear in many slaves are denoted with a superscript l. By $n_{kk'}$ we denote the number of times the pair k, k' appears in different slaves. The cost coefficients of a slave: $\tilde{c}_{ijt} = \frac{c_{ijt}}{\left(|L \setminus T| + |K \setminus T|\right)}$ and $\tilde{c}_{kk'} = \frac{c_{kk'}}{n_{kk'}}$, sum to the costs of the original objective. The slave objective is

$$\min_{z^l_{ijt}, y^l_{ik}, y^l_{kk'}} \sum_{\substack{(i,j) \in \mathcal{I} \\ t \in T}} (\tilde{c}_{ijt} + \lambda^l_{ijt}) z^l_{ijt} + \sum_{\substack{i \in I \\ k \in H_l}} \lambda^l_{ik} y^l_{ik} + \sum_{\substack{i \in \{1,\ldots,h-1\} \\ k,k' \in SH_l}} \tilde{c}_{kk'} y^l_{ikk'} \quad , \tag{9}$$

where λ^l_{ijt} is a Lagrange multiplier corresponding to a constraint coupling the variables z^l_{ijt} for different slaves and λ^l_{ik} is a Lagrange multiplier coupling y^l_{ik} for different slaves. The derivation of the slave objective from the original objective (7) is straightforward and is omitted here but detailed in the supplementary material. The feasible set of each slave problem is a projection of the original feasible set, defined by constraints (4) to (6), to the space of the slave variables:

$$\forall (i,j) \in \mathcal{I},\ \forall t \in T,\ z^l_{ijt} \geq 0\ , \qquad \forall (i,j) \in \mathcal{I}, \quad \sum_{t \in T} z^l_{ijt} = 1\ , \tag{10a}$$

$$\forall i \in I,\ \forall k \in H_l,\ y^l_{ik} \geq 0\ , \qquad \forall i \in I, \qquad \sum_{k \in H_l} y^l_{ik} = 1\ , \tag{10b}$$

$$\forall (i,j) \in \mathcal{I}, \qquad \forall k \in H_l \qquad \sum_{t \in Desc(k)} z^l_{ijt} \leq y^l_{ik}\ , \tag{10c}$$

$$\forall i \in J \setminus \{h\}, \quad \forall k \in H_l, \qquad \sum_{k' \in Sib^l(k)} y^l_{ikk'} = y^l_{ik}\ , \tag{10d}$$

$$\forall i \in J \setminus \{h\}, \quad \forall k' \in H_l, \qquad \sum_{k \in Sib^l(k')} y^l_{ikk'} = y^l_{i+1k'}\ , \tag{10e}$$

where $Sib^l(k)$ denotes the set of sibling of class k that belong to the set H_l. The nonnegativity constraints (10a) and (10b) are introduced due to the relaxation of the variables from binary to continuous domain. Constraints (10c) to (10e) have the same form as the corresponding constraints (4) to (6) in the original problem. The constraint (10b) represents constraints (1) of the original problem, transformed according to (8). Summarizing, an intersection of the feasible sets of the slaves is equivalent to the feasible set of the original problem, in the sense that if for all i, j, k, l, t we have $z_{ijt} = z^k_{ijt} = z^l_{ijt}$ and $y_{ik} = y^l_{ik}$, $x_{jl} = x^k_{jl}$, then $(z, y, x) \in C \iff \forall l \in L \setminus T,\ (z^l, y^l) \in C^l \wedge \forall k \in K \setminus T,\ (z^k, x^k) \in C^k$, where C, C^k and C^l denote the feasible sets of the original problem and of the slaves, respectively.

3.4 Solving the Slave Subproblem

It can be proven that the feasible set of the slave problem has integral vertices. A proof can be found in the appendix. In consequence, each slave can be seen as a labelling problem where we assign a label $k \in H_l$ to each row i and a label t to each pixel $(i, j) \in \mathcal{I}$. We find the optimal labelling by dynamic programming.

Given row-class k assigned to row i by slave l, it is easy to determine for each pixel in the row the optimal class t^{lk}_{ij}. Constraint (10c) restricts the set of classes that can be used in a row labelled k to descendants of k or to ones that do not descend from any $k \in H_l$. We denote the latter set by $\tilde{T}_l = T \setminus \bigcup_{k \in H_l} Desc(k)$. The class assigned by the slave to pixel (i, j) is

$$t^{lk}_{ij} = \arg\min_{t \in Desc(k) \cup \tilde{T}_l} (\tilde{c}_{ijt} + \lambda^l_{ijt}) \quad . \tag{11}$$

Algorithm 2 Dynamic program solving the slave subproblem.

```
for all $k \in H_l, i \in I$ do                                    ▷ dyn. prog. on $t_{ij}$
    for all $j \in J$ do
        $t_{ij}^{lk} \leftarrow \arg\min_{t \in Desc(k) \cup \tilde{T}_l} (\tilde{c}_{ijt} + \lambda_{ijt}^l)$
    end for
    $c_{ik}^l \leftarrow \sum_j (\tilde{c}_{ijt_{ij}^{lk}} + \lambda_{ijt_{ij}^{lk}}^l) + \lambda_{ik}^l$
end for
for all $k \in H_l$ do                                             ▷ dyn. prog. on $k_i$
    $\phi^l(1, k) \leftarrow c_{1k}^l$
end for
for $i = 2, \ldots, h$ do
    for $k \in H_l$ do
        $\phi^l(i, k) \leftarrow \min_{k' \in H_l} \phi^l(i-1, k') + c_{ik}^l + \tilde{c}_{k'k}$
        $k^l(i-1, k) \leftarrow \arg\min_{k' \in H_l} \phi^l(i-1, k') + c_{ik}^l + \tilde{c}_{k'k}$   ▷ store opt. prev. class
    end for
end for
$k_i, t_{ij} \leftarrow$ BACKTRACK$(\phi^l, k^l)$            ▷ extract optimal $k_i$ and $t_{ij}$ from recorded info
```

From objective (9) we derive the optimal cost of assigning row class k to image row i, which is the sum of costs for each pixel and the per-row cost

$$c_{ik}^l = \sum_{j=1}^{w} (\tilde{c}_{ijt_{ij}^{lk}} + \lambda_{ijt_{ij}^{lk}}^l) + \lambda_{ik}^l \quad . \tag{12}$$

The optimal cost of assigning classes for the i first rows, denoted $\phi^l(i, k)$, where k is the row class assigned to row i, can be recursively defined as

$$\phi^l(i, k) = \begin{cases} c_{1k}^l & \text{if} \quad i = 1 \\ \min_{k' \in H_l} \phi^l(i-1, k') + c_{ik}^l + \tilde{c}_{k'k} & \text{otherwise.} \end{cases} \tag{13}$$

We use the recursive structure of the subproblem to formulate algorithm 2 for finding its optimal solution. First, for each row and each candidate row label $k \in H_l$, optimal pixel classes t_{ij}^{lk} are determined for each pixel in the row according to (11). They are then used for determining costs c_{ik}^l of assigning classes $k \in H_l$ to image rows according to (12). Finally we run the Viterbi algorithm according to (13) to retrieve the optimal labelling of all rows.

4 Experiments

We tested the performance of our algorithm on two datasets. For each of them, we created a shape prior consisting of a tree hierarchy of classes and a table of pairwise potentials for each nonterminal node. We used binary potentials to penalize invalid pairs of neighboring classes, like sky under wall, with infinite cost. For each image, we run the DD algorithm for 100 iterations, with a fixed

Table 3. Performance on the ECP dataset. The rows corresponding to classes present class accuracy (the diagonal entries of confusion matrices, or recall). The bottom rows contain average class accuracy and total pixel accuracy. Starting from left, we present the performance of three layers of Martinovic's solution [15], and the results of Cohen et al. [16], using 'raw' per-pixel energies, and with SVM scores on top of the energies.

	[15]-L1	[15]-L2	[15]-L3	[16]	[16]-SVM	Ours	Our confusion matrix							
roof	70	73	74	93	90	91	91	0	0	2	2	0	5	roof
shop	79	86	93	96	94	95	0	95	0	0	0	0	4	shop
balcony	74	71	70	92	91	90	1	0	90	0	4	0	5	balc.
sky	91	91	97	96	97	96	4	0	0	96	0	0	0	sky
window	62	69	75	87	85	85	3	1	4	0	85	0	5	wind.
door	43	60	67	82	79	74	0	22	0	0	0	74	4	door
wall	92	93	88	88	90	91	1	3	2	0	3	0	91	wall
class aver.	73.0	77.6	80.6	90.6	89.4	88.8								
pixel accur.	82.6	85.1	84.2	90.3	**90.8**	**90.8**								

sequence of decaying step size $\alpha_n = a/\sqrt{n}$, where n is iteration number and a is a constant. The average running time was about 4 minutes per image.

The ECP dataset [2] consists of about 100 rectified images of Haussmannian building facades with annotations of 7 classes: sky, roof, wall, window, balcony, shop and door. The ground-truth annotations are consistent with the grammar used in [2], which models facade structure as a grid of windows with balconies constrained to individual windows or extending over the width of the facade. In consequence the ground truth is incorrect on a number of images. We use the annotations provided by [15], which do not respect semantic constraints (balconies and windows can be misaligned, small pieces of doors may float above the ground), but are more accurate in terms of pixel classification.

The Graz50 dataset [3] is composed of 50 rectified images of facades of different architectural styles. They feature more structural variation than the ones of the ECP dataset. The labels include four classes: sky, wall, window and door.

Performance on the ECP Dataset has been tested using per-pixel energies that follow the description from [16]. We use a multi-feature extension of TextonBoost, implemented by the authors of [20]. We use SIFT and Color SIFT, Local Binary Patterns and location features. Features of each type are clustered using K-means into 512 clusters. The final feature vector is a concatenation of histograms of appearance of cluster members in a neighborhood of 200 randomly sampled rectangles. The per-pixel costs c_{ijt} are output by a multi-class boosting classifier [21]. We follow the protocol of [15] and [16] in performing experiments on five folds with 80 training and 20 testing images. The results are presented in table 3. Our method attains the same performance as [16]. However, our algorithm can accept a user-defined shape prior as input, which makes it more general. The shape prior can express constraints on alignment of architectural elements in two dimensions, which is beyond the expressive power of [16].

Table 4. Results of the experiment on the Graz50 dataset. The second and third columns of the table show diagonal entries of the confusion matrices for results reported by Riemenschneider et al. [3] and our results. The right-hand side of the table contains the confusion matrix for our results. Example result is shown on the right hand side.

	[3]	Ours	conf. mat.				
sky	91	93	93	0	0	6	sky
window	60	82	0	82	0	17	window
door	41	50	0	14	50	36	door
wall	84	96	0	3	0	96	wall
class average	69.0	80.3					
total pixel accur.	78.0	**91.8**					

Performance on the Graz50 dataset has been tested using the same type of pixel costs as for the ECP dataset. Five folds were used, each time the dataset was split into 40 training and 10 test images. The results are presented in table 4. One reason why our results are superior to those in [3] is that their method requires severe subsampling of the image to be tractable. Our method is more computationally efficient and can be run on full-resolution images.

5 Conclusion and Future Work

We presented a novel approach to shape prior-based facade analysis in which the task of parsing is formulated as a binary linear program. Our formulation does not suffer from the curse of procedural exploration, that is typical for existing split grammar parsers. It enables approximating globally optimal segmentations by means of efficient optimization algorithms. We established a new state-of-the-art level of performance on the ECP and Graz50 facade datasets.

As a direct extension of this work, we are considering learning the pairwise potentials using the approach presented in [22]. In longer perspective, due to increasingly good results reported recently on rectified images, we see the relevance of addressing difficulties arising from the use of real life images, like modeling projection of three-dimensional geometry of buildings on the image plane and handling occlusions (e.g., by cars, vegetation, pedestrians or other buildings).

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

References

1. Teboul, O., Simon, L., Koutsourakis, P., Paragios, N.: Segmentation of building facades using procedural shape priors. In: CVPR. (2010) 3105–3112

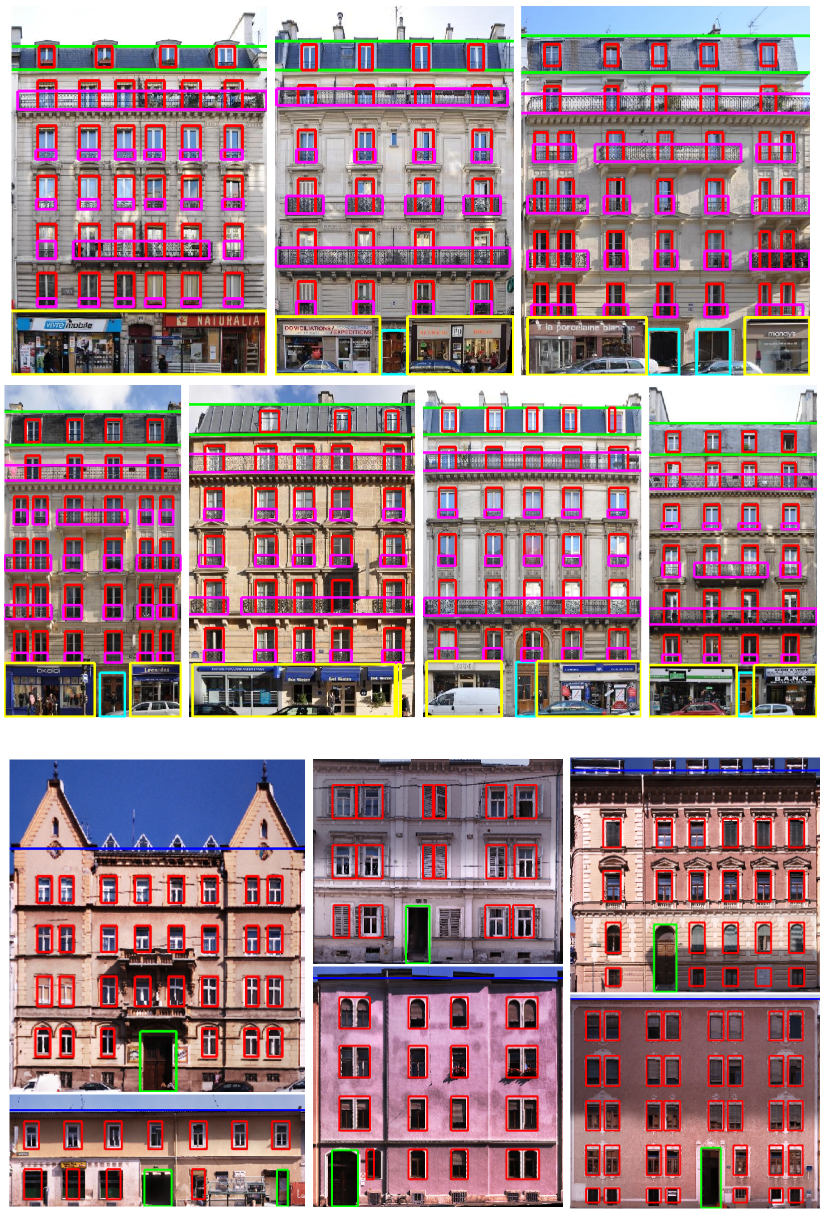


Fig. 4. Example parsing results on the ECP dataset (top) and on the Graz50 dataset (bottom). Green lines separate sky from roof and roof from facade. Balconies are outlined in magenta, shops in yellow, and doors in cyan. Note the variety of alignment patterns supported by the algorithm (top right). Typical errors are missed doors and missed roof windows.

2. Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., Paragios, N.: Shape grammar parsing via reinforcement learning. In: CVPR. (2011) 2273–2280
3. Riemenschneider, H., Krispel, U., Thaller, W., Donoser, M., Havemann, S., Fellner, D., Bischof, H.: Irregular lattices for complex shape grammar facade parsing. In: CVPR. (2012)
4. Zhu, S.C., Mumford, D.: A stochastic grammar of images. Foundations and Trends in Computer Graphics and Visions **2** (2006) 259–362
5. Stiny, G.N.: Pictorial and Formal Aspects of Shape and Shape Grammars and Aesthetic Systems. PhD thesis, University of California, Los Angeles (1975) AAI7526993.
6. Ohta, Y., Kanade, T., Sakai, T.: An analysis system for scenes containing objects with substructures. In: Proceedings of the Fourth International Joint Conference on Pattern Recognitions. (1978) 752–754
7. Ohta, Y., Kanade, T., Sakai, T.: A production system for region analysis. In: Proceedings of the Sixth International Joint Conference on Artificial Intelligence. (1979) 684–686
8. Han, F., Zhu, S.C.: Bottom-up/top-down image parsing with attribute graph grammar. IEEE Transactions on Pattern Analysis and Machine Intelligence **31** (2009) 59–73
9. Wang, W., Pollak, I., Wong, T.S., Bouman, C.A., Harper, M.P.: Hierarchical stochastic image grammars for classification and segmentation. IEEE Transactions on Image Processing **15** (2006) 3033–3052
10. Jin, Y., Geman, S.: Context and hierarchy in a probabilistic image model. In: CVPR (2). (2006) 2145–2152
11. Ahuja, N., Todorovic, S.: Connected Segmentation Tree - A Joint Representation of Region Layout and Hierarchy. In: CVPR. (2008)
12. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. ACM Transations on Graphics **25** (2006) 614–623
13. Teboul, O.: Shape Grammar Parsing : Application to Image-based Modeling. PhD thesis, Ecole centrale Paris (2011)
14. Koziński, M., Marlet, R.: Image parsing with graph grammars and markov random fields. In: WACV. (2014)
15. Martinovic, A., Mathias, M., Weissenberg, J., Van Gool, L.: A three-layered approach to facade parsing. In: ECCV, Springer (2012)
16. Cohen, A., Schwing, A., Pollefeys, M.: Efficient structured parsing of facades using dynamic programming. In: CVPR. (2014)
17. Komodakis, N., Paragios, N., Tziritas, G.: MRF energy minimization and beyond via dual decomposition. IEEE Trans. PAMI **33** (2011) 531–552
18. Sontag, D., Globerson, A., Jaakkola, T.: Introduction to dual decomposition for inference. In Sra, S., Nowozin, S., Wright, S.J., eds.: Optimization for Machine Learning. MIT Press (2011)
19. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning **3** (2011) 1–122
20. Ladický, L., Russell, C., Kohli, P., Torr, P.H.S.: Associative hierarchical random fields. IEEE Transactions on Pattern Analysis and Machine Intelligence **99** (2013) 1
21. Shotton, J., Winn, J.M., Rother, C., Criminisi, A.: *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: ECCV (1). (2006) 1–15

22. Komodakis, N.: Efficient training for pairwise or higher order CRFs via dual decomposition. In: CVPR. (2011) 1841–1848