Irregular lattices for complex shape grammar facade parsing

Hayko Riemenschneider Ulrich Krispel Wolfgang Thaller Michael Donoser
Sven Havemann Dieter Fellner Horst Bischof
Institute for Computer Graphics and Vision & Computer Graphics and Knowledge Visualization
Graz University of Technology, Austria
(hayko,donoser,bischof)@icg.tugraz.at
(u.krispel,w.thaller,s.havemann,d.fellner)@cgv.tugraz.at

Abstract

High-quality urban reconstruction requires more than multi-view reconstruction and local optimization. The structure of facades depends on the general layout, which has to be optimized globally. Shape grammars are an established method to express hierarchical spatial relationships, and are therefore suited as representing constraints for semantic facade interpretation. Usually inference uses numerical approximations, or hard-coded grammar schemes. Existing methods inspired by classical grammar parsing are not applicable on real-world images due to their prohibitively high complexity. This work provides feasible generic facade reconstruction by combining low-level classifiers with mid-level object detectors to infer an irregular lattice. The irregular lattice preserves the logical structure of the facade while reducing the search space to a manageable size. We introduce a novel method for handling symmetry and repetition within the generic grammar. We show competitive results on two datasets, namely the Paris2010 and the Graz50. The former includes only Hausmannian, while the latter includes Classicism, Biedermeier, Historicism, Art Nouveau and post-modern architectural styles.

1. Introduction

Urban environments contain many man-made objects which differ from natural objects by exhibiting highly regular structures. Man-made objects, for example facades, are typically organized in logical hierarchies, for example, separation into floors. For the goal of globally consistent 2D semantic segmentation of such facades, knowledge about this structure is vital.

A popular method of encoding knowledge about the high-level structure of facades are shape grammars, in particular split grammars. Such knowledge is represented in *parse trees* which represent the logical facade structures. The optimization of parse trees is a hard problem, as the solution space is very large and has complex structure.

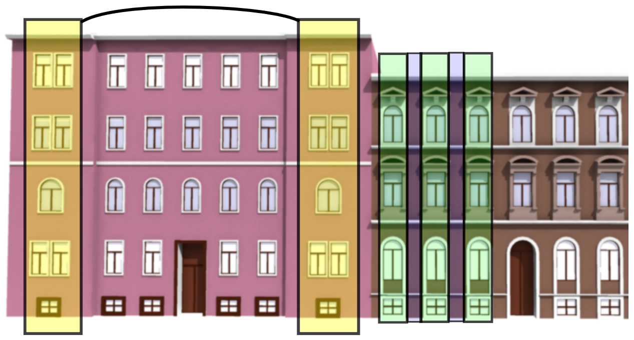
Figure 1. High quality complex architecture modeling through shape grammar using symmetry and repetition.

A well-known algorithm for parsing context-free grammars is the Cocke-Younger-Kasami (CYK) algorithm, a dynamic programming algorithm which iterates over all substrings of the input and all nonterminals of the grammar. Schlesinger *et al.* [19] have adapted this algorithm for two-dimensional split grammars. Unfortunately, the time complexity of $O(w^2h^2(w+h)|G|)$ to match a grammar G with $|G|$ rules to an image with $w \times h$ pixels, is prohibitive for real-world image sizes.

Consequently, research in shape grammar-based facade segmentation has focused on Markov Chain Monte Carlo (MCMC) and related methods to optimize the parse tree. Statistical solutions on a high-dimensional solution space has the problem of local minima, which requires a reduction of degrees of freedom and a reasonable initialization for the optimization. Nevertheless, these sampling methods can yield good results [2, 12, 25].

In contrast, Schlesinger's modified CYK algorithm has different strengths. Its running time depends only linearly on the number of rules in the grammar, so large grammars are possible. It is not a sampling-based method, so it can-not get stuck in local minima, and guessing a good initial solution is not necessary. Finally, the CYK algorithm has no problem with added *local* degrees of freedom, i.e. decisions that are independent between different branches of the parse tree. For MCMC-based approaches, these increase the overall complexity of the problem.

For these reasons, it is worthwhile to investigate how the CYK algorithm's time complexity problems can be overcome in practice. The CYK algorithm is efficient enough for small inputs ($w, h \leq 60$), but limiting input resolution to that size is not feasible.

In this work, we therefore provide two contributions: We combine the results of low-level classifiers and mid-level object detectors to build an *irregular lattice* of sufficiently low resolution. Further, we propose a novel CYK-based algorithm which supports symmetric and repeating structures, and is able to handle input noise from the estimated likelihoods of the different classes for every lattice tile.

2. Related Work

Urban modeling has seen a wide range of methods to produce 3D models. Multi-view reconstructions are used to derive unstructured point clouds modeling the urban environment as sets of textured points or models [1, 8, 14]. Consequential works assume planarity and derive piecewise planar models which greatly reduce the footprint to planar partitions [15, 10, 29]. This is further refined to partitioning images in terms of semantic regions [20, 11] and research with focus on parsing buildings [4, 22, 30] introduced constraints and prior knowledge about the 3D scene geometry.

Shape grammars are a formal method well suited to describe the set of hierarchical partitions (in form of a parse tree) of an image. They have originally been introduced by Stiny *et al.* [21] for expressing the design of 2D line drawings. Based on the concepts from formal grammars (string replacements), a shape design is expressed by a set of shape matching and replacement rules. Shape grammars have been studied in various areas, e.g. architecture design [16] or brand retaining design of consumer products [7]. In recent years, shape grammars have been successfully applied in computer graphics for automatic creation of variations of a class of models. Wonka *et al.* [28] introduced *split grammars* for the semiautomatic generation of architecture. These concepts have further been extended by Mueller *et al.* [17] into a shape grammar system called Computer Generated Architecture (CGA) shape.

There are a range of algorithms in the literature that determine parse trees for specific grammars [17, 3]. Similarly, the approach of Toshev *et al.* [26] identifies buildings from unorganized 3D point clouds via parse trees of roof structures. However, these approaches require to change the algorithm if the grammar changes. Vanegas *et al.* [27] combine shape grammar and Manhattan planarity for large building outline modeling. A common approach to obtain a solution is to formulate rule applications as a statistical inference problem and use numerical approximation methods, e.g. Markov Chain Monte Carlo (MCMC). This has been shown for facade segmentation in the work of Alegre *et al.* [2] and Ripperda *et al.* [18]. Recently, these methods have also been applied in computer graphics to find productions of grammar based procedural models given a high level description of the desired result [23].

Most related to our approach is that of Teboul *et al.* [25], where a context-free grammar is used to partition rectified building facades into semantic image segmentations. The process involves training a local classifier for the desired terminal symbols and performing hierarchical splits to partition the image. The splitting process is guided by the low-level classifier probabilities and is designed to overcome the limitations of the local structure regularization. However, the optimization scheme, as well as the shape grammar, are limited to a simple style of split combinations. In their work they show results on Hausmannian and skyscraper-style facades, which contain highly regular structures. Such regular structures appear in the form of equally spaced window terminals, simplified door layout and overall low complexity. While it is well-suited for Hausmannian architectural styles, the approach might not scale well to less regular, but still highly structured facade layouts which are present in most other architectural styles.

3. Shape Grammar for Facade Parsing

We now introduced the proposed general grammar for parsing the 2D facade structure and its relationship with the 2D image space. A formal grammar is a tuple $G = (N, \Sigma, P, S)$ consisting of a set N of nonterminal symbols, a set Σ of terminal symbols, a set P of production rules of the form $\alpha \to \beta$, where $\alpha \in (N \cup \Sigma)^+$ and $\beta \in (N \cup \Sigma)^*$, and a starting symbol (axiom) $S \in N$. A context-free grammar is a formal grammar where production rules are limited to the form $A \to \beta$ where $A \in N$.

A split grammar is a context-free grammar where the right-hand side of each rule consists either of a single nonterminal symbol, or of a terminal symbol (an *operator*) followed by zero or more non-terminal symbols, where the number of non-terminal symbols matches the number of arguments expected by the operator.

A *parse tree* is an ordered tree whose interior nodes are labeled by nonterminal symbols, and whose leaves are labeled by terminal symbols (operators) from the grammar, and by an attribute a. The meaning of this attribute depends on the operator. The root of the tree is labeled by the grammar's starting symbol S. It follows from the restrictions we have imposed on the rules that for each interior node (nonterminal), its leftmost child is a leaf (an operator) and its other children are interior nodes (nonterminals) as well.

We use $d = op_a\ d_1 \dots d_n$ to denote a parse tree, and $\mathcal{D}$ for the set of all possible parse trees for a given grammar. The number of arguments n is a constant for each operator op. The attribute value a is taken from a set $\mathcal{A}_{op}$ which may be different for each operator. The subtrees d_1 through d_n are themselves parse trees, but with different start symbols.

3.1. Operators and their Graphical Interpretation

A range is a rectangular area of an image that may or may not be mirrored around the y axis. Formally, a range r is a tuple (x_1, y_1, x_2, y_2) with $y_1 \leq y_2$. We write $p \in r$ for a point $p = (x, y)$ iff $((x_1 \leq x < x_2) \vee (x_2 \leq x < x_1)) \wedge (y_1 \leq y < y_2)$. For the mirrored version of a range, we write $\bar{r} = (x_2, y_1, x_1, y_2)$.

To give a graphical interpretation of a parse tree d, we define a function $D(d, r)$ which maps the parse tree to a labelling of the rectangular range r. This function can be defined recursively for each operator.

There is exactly one label operator for each label (wall, window, door, …); a label operator takes no arguments and represents a rectangular image area belonging to that class. The remaining operators each split the range r they operate on into several subranges r_i. Our method supports the standard horizontal and vertical split operators, as well as horizontal and vertical *alternating repeat* and horizontal *mirror with center* operators. The mirror operator describes the common symmetry pattern $\mathtt{AB\bar{A}}$. The alternating repeat operators are an indexed family of operators $alt(3)$, $alt(5)$, …which describe patterns of the form ABA, ABABA, …, respectively.

Each of these operators defines a set A_{op} of possible attribute values. For a standard split operator, the attribute value indicates the (relative) position of the split line between the two parts, for the mirror and the alternating repeat operators, it is the position of the split between the A and B parts. We could have defined a single alternating repeat operator whose attribute a also describes the number of repetitions. Our approach provides more flexibility in that it allows the grammar to express relations between the repetition counts in different parts of the facade. As we can establish reasonable upper bounds on the repetition counts, a variable repetition count can always be expressed by having one rule for each possible count.

To generalize, we observe that for each of the above operators, the number m of subranges is either equal to or greater than the number of arguments n. Each subrange r_i is described by the sub-parse tree $d_{f(i)}$. The values of m, r_i and the function f may depend on the operator op, on the attribute a from the parse tree and range r. We thus write:

$$D(op_a\ d_1 \dots d_n, r) = \bigcup_{i=1}^{m} D(d_{f(i)}, r_i) \tag{1}$$

where she set union operator is used to express the concatenation of labellings on adjacent ranges r_i.

Additionally, our method allows each nonterminal symbol in the grammar to be annotated with minimum and/or maximum sizes. Only parse trees are considered valid if the extent of the nonterminal falls within the allowed range.

3.2. Grammar Factorization

Methods based on MCMC sampling need to keep the total degrees of freedom as low as possible. Teboul *et al.* [25] do this by a process called *factorization*, i.e. where a standard grammar allows different attribute values in different subtrees, they demand that all these attributes are assigned the same value, thus reducing the number of degrees of freedom and imposing a more regular facade structure.

Dynamic programming based approaches are not affected by this. Many local decisions in local subproblems do not pose a problem. Making global choices, however, violates the optimal substructure condition. Factorization can nevertheless be applied as a preprocessing step on the grammar, which increases the number of rules.

Consider a grammar with rules $S \rightarrow \mathit{split}\ A\ A$, $A \rightarrow b$ and $A \rightarrow c$. The decision between the terminal symbols b and c is made independently for both occurrences of A. To force the decision to be the same in both cases, we need to use a *factorized* grammar with rules $S \rightarrow \mathit{split}\ AB\ AB$, $S \rightarrow \mathit{split}\ AC\ AC$, $AB \rightarrow b$, and $AC \rightarrow c$.

We have found this method useful for enforcing constraints like, e.g., equal numbers of floors in different parts of a facade. The repeat and mirror operators also encode some non-local structure that would otherwise need to be expressed via grammar factorization.

4. Data-driven terminal symbols

The grammar inference parses the facade layout by valid transitions between label operator terminals. These are the basic semantic building blocks of a facade, for example, wall, window, door, balcony, etc. They are denoted by labels $\mathcal{L}$ and will be used to guide the data-driven high-level inference process by bottom up merit functions and the split proposal initialization. Hence, for evaluating the probability of a pixel belonging to a final semantic class, we learn two merit functions to infer class terminal labels ψ without spatial extent and object terminal label θ with spatial extent.

4.1. Pixelwise merit

The merit function for the class terminal label is inspired by semantic scene labeling [20, 13]. We use a pixelwise classifier on local image features giving the log-likelihood of each pixel belonging to a class terminal $\psi \in \Sigma$, as

$$\Psi(x_i) = -log(P(\psi|i)) \tag{2}$$

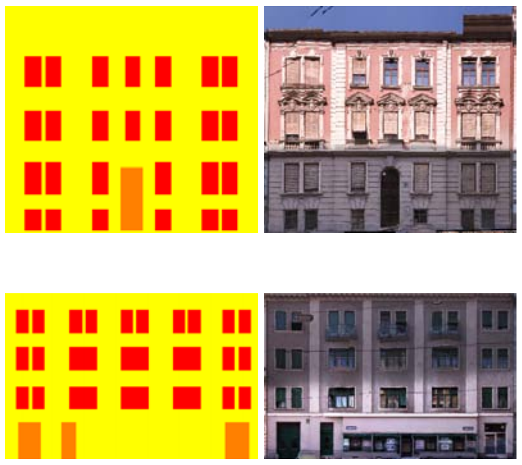

Figure 2. Examples of symmetry, repetition and varying sizes in Historicism and Art Nouveau facades.

where $i \in I$ is a single pixel and P is the probability for class. For training we use Randomized Forests [6] in combination with raw RGB pixel intensities calculated over local patches (size 15x15). Randomized Forests have the ability to efficiently evaluate high-dimension splitting as well as handle noise in training data, coupled with low computational time for the evaluation. They have proven to be well-suited for the task of semantic image classification, as the independent low-level features for stuff classes without specific spatial extent (i.e. sky, wall, roof, shop, etc.) can be modeled sufficiently well by local patches.

4.2. Terminal merit

The second merit function is built on top of a detection process, where an object detector is trained for each object terminal label. This is a higher-level approach to better model object terminals with known spatial extent (i.e. windows, doors, etc). Equally, a set of pixels c_i is assigned a log-likelihood of belonging to an object terminal $\theta \in \Sigma$, as

$$\Theta(x_{c_i}) = -log(P(\theta|c_i)) \tag{3}$$

where $c_i \in I$ is a clique of pixels which defines the pixels belonging to the same instance of the object terminal θ. The clique of image pixels are determined by the provided rough outline in form of bounding boxes by the object detectors. The merit is usually available as terminal-wise merit, and we transfer it to a pixelwise merit $\Theta(x_i)$ by evaluating it for each $i \in c_i$ of the clique. For training we use Hough Forests [9], where positive and negative object samples are extracted from the training data. Hough Forests share the ability to evaluate high-dimension splitting and further incorporate object center location in the training.

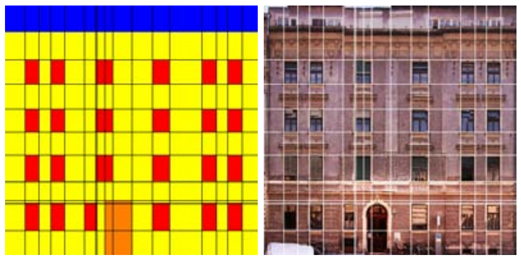

Figure 3. The irregular lattice is a rectangular splitting of the image into tiles which depend on terminal symbols.

4.3. Irregular Rectangular Lattice

Classically the labelling solution is sought by the maximum a posterior (MAP) solution $x* = \mathrm{argmin}_x\, \Psi(x_i) + \Theta(x_i)$, however, we are interested in a shape grammar parsing of the image, which incorporates high-level structure information from class and object terminals labels and logical structures such as floors, symmetries and repetitions.

Unfortunately, any inference approach in this scenario suffers from the curse of dimensionality. Contrary to previous work, which assumes factorization [25], fixed discretization [17], even equal height/width of terminal symbols [24], we allow a wider range of facade interpretation and do not restrict the parsing. Our only assumption is placed on rectified image data, which provides orthogonal frames for buildings. Unlike the above assumptions, this is valid for a wide range of architecture styles, see Figure 2.

Our solution to the curse is based on an irregular rectangular lattice. Such a lattice is a splitting of the orthogonal building frame into lattice tiles of varying width and height. Roughly speaking, each tile represents the range of a terminal symbol, see Figure 3 for an example lattice overlaid over the annotation (left) and image data (right).

The lattice is our initialization for the inference process and is defined by vertical and horizontal split lines. We solve each dimension independently as we do not constrain the solution to be regular. A split line is a transition which divides the image space into tiles, where neighboring tiles may belong to a different terminal label. Given the joint distribution of the image from the pixelwise classifiers, we are looking for the marginalized label transitions for each dimension, which align well with image data. Based on the merit functions, we define the problem of finding optimal label transitions as minimizing the following energy term

$$E(x) = \sum \Psi\left(x_i\right) + \sum \Theta\left(x_i\right) + \lambda \sum \Upsilon\left(x_i, x_j\right) \ , \tag{4}$$

where $\Psi(x_i)$ and $\Theta(x_i)$ are our pixelwise and terminal-wise likelihoods and $\Upsilon(x_i, x_j)$ is a standard contrast sensitive Potts model to align the solution to the image data. Since the likelihoods can be quite noisy, λ is set high (20). We use a standard graph cut to efficiently solve this energy minimization problem [5].

To infer the split lines from the label transitions, we define a transition image $\mathcal{T}$, where the neighborhood function between pixels is defined as

$$\Upsilon'(x_i, x_j) = \begin{cases} 0 & \text{if} \quad x_i = x_j \\ 1 & \text{if} \quad x_i \neq x_j \end{cases}, \tag{5}$$

where x_i and x_j are the inferred labels of two neighboring pixels. This transition image is an indication of semantic change between the terminals. To extract split lines, we marginalize out each dimension by

$$\mathcal{T}(X' = x') = \sum_y \mathcal{T}(X' = x', Y' = y'), \tag{6}$$

where $\mathcal{T}(X' = x')$, for example, are the lattice split lines along the x-axis. Hence, in our facade parsing method we initialize certain split proposals, which remove a large portion of the search space. For the remaining space, the split proposals functions are evaluated to guide the inference method, described in the next section. As it is difficult to write a perfect generic detector for each category with large intra-class variance, the marginalization allows us to only require one terminal detection per floor and column, which is enough to provide the layout initialization.

5. Grammar Matching

In this section we describe our novel inference for matching a lattice to the grammar. The modified CYK algorithm finds a solution by minimizing a cost function $c(d, I)$ over all possible $d \in \mathcal{D}$:

$$d^* = \underset{d \in \mathcal{D}}{\operatorname{argmin}}\, c(d, I), \tag{7}$$

where the set $\mathcal{D}$ denotes the set of all possible parse trees. Dynamic programming algorithms like CYK require the so-called *optimal substructure property*, i.e. it must be possible to efficiently calculate the optimal solution from the optimal solutions to its subproblems. A subproblem in our case means finding the optimal way of matching a given nonterminal against a given rectangular subrange of the input. We write $q(I, r, S)$ to denote the optimal match for a nonterminal S on the subrange r of input I.

We therefore require that given an operator *op* (that takes n arguments), an attribute value a and a range r, we can efficiently determine n subproblems (r_i, S_i), such that

$$\min_{d_1 \ldots d_n} c(op_a\, d_1 \ldots d_n, I, r) = c(op_a\, d_1^* \ldots d_n^*, I, r), \tag{8}$$

where $d_i^* = q(I, r_i, S_i)$ are the optimal solutions for the subproblems.

The optimal solution for (r, S) can thus be determined by calculating the costs for all rules applicable to S and for all possible values of a for the operator mentioned in each rule and choosing the minimum. This assumes that all subproblems (r_i, S_i) have already been processed, which can easily be guaranteed by processing smaller ranges first.

5.1. Cost Function

Let us now define a cost function that implements a maximum a posteriori probability estimator.

From the grammar, we get a prior probability distribution over all facades. As we are not currently using a stochastic grammar, this distribution is uniform for all segmentations that can be described by a parse tree, and zero for all "impossible" segmentations. Thus, we maximize the posterior probability over the set of all parse trees $\mathcal{D}$ (rather than over the set of all possible labellings). We want our cost function to be the log-likelihood of the parse tree:

$$c(d, I, r) = -\log P(D(d, r)|I). \tag{9}$$

In order to implement the estimator, we need a cost function that fulfills the optimal substructure condition (8) and approximates this "ideal" cost function. First, we check whether any size constraints for the nonterminal symbol at the root of the current parse tree are violated. If so, the parse tree is assigned infinite cost. Otherwise, we proceed according to the operator used at the root of the parse tree. For label operators, we calculate c by summing up the pixel-wise merits Ψ from Section 4.1 for all pixels in the range r. For the other operators, we get (using Equation 1 and the assumption that the subtrees d_i are statistically independent from each other):

$$\begin{aligned} c(op_a\, d_1 \ldots d_n, I, r) &= -\log \prod_{i=1}^{m} P(D(d_{f(i)}, r_i)|I) \\ &= \sum_{i=1}^{m} c(d_{f(i)}, I, r_i) \end{aligned} \tag{10}$$

In the case of the standard (vertical and horizontal) split operators, this boils down to $c(d_1, I, r_1) + c(d_2, I, r_2)$, which fulfills the optimal substructure condition (8).

In the general case, which includes the mirroring and alternating repetition operators, we get a cost function that violates the optimal substructure condition by matching the same subtree against multiple subranges:

$$\begin{aligned} c(mir_a\, d_1 d_2, I, r) &= c(d_1, I, r_1) + c(d_2, I, r_2) \\ &\quad + c(d_1, I, \bar{r_3}) \end{aligned} \tag{11}$$

$$\begin{aligned} c(alt(m)_a\, d_1 d_2, I, r) &= \sum_{i=1,3\ldots}^{m} c(d_1, I, r_i) \\ &\quad + \sum_{i=2,4\ldots}^{m} c(d_2, I, r_i) \end{aligned} \tag{12}$$

Clearly, the d_i which minimize these costs are not necessarily the optimal solutions $d_i^* = q(I, r_i, S_i)$ on any of the subranges. It is, however, a reasonable approximation to assume that they are.

Making that assumption still does not make the algorithm efficiently implementable. The cost of the optimal sub-solutions, $c(d_i^*, I, r_i)$, has already been calculated by previous iterations of the dynamic programming algorithm, but the costs $c(d_i^*, I, r_j)$ for $i \neq j$ have not. We therefore estimate these costs based on values that are more readily available, such as the costs $c(d_i^*, I, r_i)$ and $c(d_j^*, I, r_j)$, which are already pre-calculated. We also introduce a dissimilarity estimate $\Delta(I, r_i, r_j)$, which indicates the probability that two ranges should have the same labeling:

$$\Delta(I, r_i, r_j) := -logP(x_{r_i} = x_{r_j} | I) \tag{13}$$

This can be calculated from Ψ, again assuming statistical independence between the pixels. In Section 5.3, we will see how it can be precalculated for efficient lookup.

We can estimate the probability that two ranges can be explained by the same parse tree using the probability that one of the ranges can be explained by the parse tree and the probability that the ranges have the same labelling:

$$\begin{aligned} & P(x_{r_i} = D(d_i^*, r_i) \wedge x_{r_j} = D(d_i^*, r_j) | I) \\ = \; & P(x_{r_i} = D(d_i^*, r_i) \wedge x_{r_i} = x_{r_j} | I) \\ \approx \; & P(x_{r_i} = D(d_i^*, r_i) | I) P(x_{r_i} = x_{r_j} | I) \end{aligned} \tag{14}$$

Taking this estimation together with the fact that $c(d_i^*, x, r_j) \geq c(d_j^*, x, r_j)$, we get

$$c(d_i^*, x, r_j) \approx \max(c(d_j^*, x, r_j), \Delta(x, r_i, r_j)). \tag{15}$$

5.2. Inference on the Lattice

For the basic horizontal and vertical split operators, the irregular lattice amounts to limiting the choices for the attribute values a at the various parse tree nodes. The result is equivalent to maximizing posterior probability under the assumption that each lattice tile has a homogenous label.

For mirror and repeat operators, the situation is slightly more complicated. In an application of the mirror operator, if the subrange r_1 is exactly representable on the lattice, the subrange $\bar{r_3}$ that is its mirror image might not be, and vice versa. Likewise, the various subranges r_i of a repetition operator will usually not fit exactly.

The subproblem solutions d_i^* are therefore calculated on the closest range actually available in the lattice, and then used as an approximation for the exact range. This has the effect of adding the split lines required by the symmetries and repetitions to the output labelling, even if they have not been found during the lattice generation phase. We define the dissimilarity Δ on ranges of different sizes to be the Δ between the smaller range and the corresponding subrange in the center of the larger range, plus an extra penalty value proportional to the difference in areas.

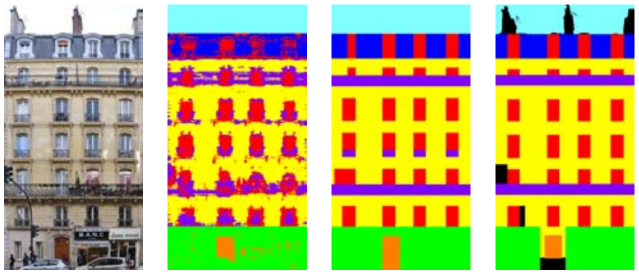
Figure 4. Left to right: orthophoto, MAP segmentation, grammar segmentation and ground truth labels (Paris2010).

5.3. Algorithmic Complexity

Given a width w and a height h, the number of subranges is $O(w^2h^2)$. The space complexity of a simple implementation of the algorithm is therefore $O(w^2h^2|N|)$, where N is the set of nonterminal symbols.

For every subrange and nonterminal, a rule and the appropriate value for the attribute $a \in A$ has to be chosen by exhaustive search; thus, the total number of cost function evaluations is $O(w^2h^2|A||G|)$.

For our set of operators, $|A| \leq w$ for all horizontal operators and $|A| \leq h$ for all vertical operators, yielding $A = O(w + h)$.

The dissimilarity estimate $\Delta(x, r_i, r_j)$ can be precalculated in order to allow constant-time lookup. To achieve that, we make use of the fact that r_i and r_j either occupy the same range of x coordinates (when used by the vertical alternating repeat operator), or the same range of y coordinates (mirror and horizontal alternating repeat operators). Furthermore, note that Δ of a pair of ranges of the same height is the sum of the Δ values on the individual rows of pixels. We therefore need a lookup table with entries for every y coordinate and for every possible pair of same-size ranges of x coordinates for the horizontal operators, and another similar lookup table for the vertical operators.

The size of these lookup tables is $O(w^3h + h^3w)$, filling the tables takes $O(w^4h + h^4w)$ time. As w and h are on the same order of magnitude, this is dominated by the running time of the CYK algorithm itself.

6. Experimental Evaluation

In this section, we evaluate our method on two datasets, which show rectified building facades with high intra-class variance and occlusions. Both consists of semantic classes such as window, wall, balcony, door, roof, shop and sky. The images are annotated pixelwise with an additional (void/outlier) class. The train/test protocol specifies training on 60% of the images, and testing on the remaining images.

The Ecole Centrale Paris Facades (short: Paris2010) is a dataset by Teboul *et al.* [25], which consists of 30 images taken from rue Monge in the fifth district of Paris, resem-

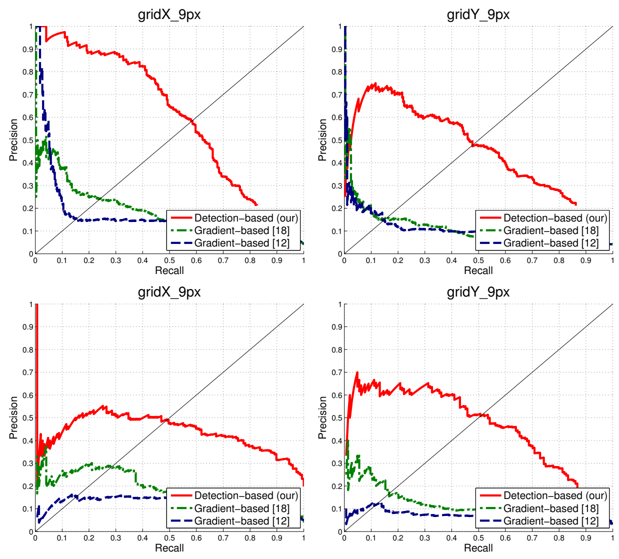

Figure 5. Lattice confidence: Compared to standard split proposals based on gradients, our detection-based proposals achieve a higher precision/ recall (+35% at EER, top: Graz50, bottom: Paris2010).

bling solely Hausmannian architecture with highly regular structures, where factorization assumptions work well.

Furthermore, we created a new dataset (Graz50) containing 50 images taken from various locations in the historical Austrian city of Graz. It includes architectural styles such as Classicism, Biedermeier, Historicism (neo-renaissance and neo-baroque), Art Nouveau and as well as various modern styles, demonstrating a wider range of facade layouts than the Paris2010 dataset. Therefore, the constraints for factorization and shop-level separation limit the grammar inference more than they help. The ground-level floor contains high variation in layout for shop and residential floors, which may or may not follow the remaining floor layout.

6.1. Lattice generation

The irregular lattice for each facade provides the splitting proposals for the grammar inference process. Compared to related work [14, 25] which uses gradient-based split proposals, we base our proposals on mid-level information of the terminal detection. To evaluate the effectiveness of such mid-level information, we compare the ground truth splits from the annotation data to the split proposals and evaluate classical recall and precision. As shown in Figure 5 we substantially outperform (+20-35% at EER) the gradient-based methods for both split directions (x,y) and both datasets.

6.2. Segmentation

Since the goal is to derive the logical structure of facade layouts, we want to show how well the facade models our algorithm infers fit the actual layout. We follow the segmentation-based evaluation common to semantic scene

Method	MAP	[25]	[24]	Our
Window	29	81	81	68
Wall	63	83	84	87
Balcony	42	72	63	69
Door	90	71	84	56
Roof	62	80	86	83
Sky	95	94	94	95
Shop	26	95	97	97
Average	58	82	84	80

Table 1. Single architecture (Paris2010): Semantic class-wise interpretation evaluation. See text for details.

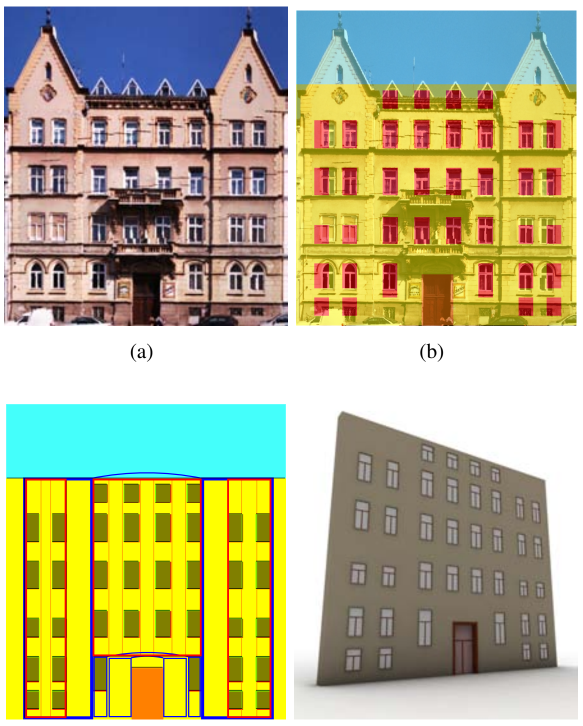
(a) (b) (c) (d)

Figure 6. (a) Orthophoto, (b) grammar segmentation overlay, (c) parsed structure with detected symmetry and repetition for this Historicism example, and d) parse tree transformed into 3D model.

Method	Window	Wall	Door	Sky	Global	Class	IoU
MAP	60	66	57	80	66	65	43
Our	60	84	41	91	78	69	58

Table 2. Multi architecture style dataset (Graz50): Semantic segmentation evaluation by global, class-wise, and IoU average.

interpretation and evaluate the accuracy in terms of pixel average, class-wise pixel average, and the intersection/union pixel average. As shown in Table 1, we reach competitive performance on the simple Hausmannian architecture. Figure 4 demonstrates results on a single facade from this dataset. Additionally, we show that our grammar matching also works well for more difficult architectural styles using a different grammar, as shown in Table 2. Since a segmentation may not capture the actual structure, we also show a qualitative example (see Figure 6) of a parsed facade, which shows how flexible symmetry and repetition is detected as opposed to factorization constraints.

7. Conclusion and Future Work

In this work we showed a novel approach to general facade reconstruction which is not limited to fixed grammar rules or hard-coded inference implementations. We proposed a method to combine low-level classifiers with mid-level object detectors to infer an irregular lattice which reflects the semantic changes on a facade. This lifts the constraints of high complexity of existing methods and allows practical solutions for real world images and generic grammars. Our matching algorithm supports hierarchical spatial relationships as well as symmetries and repetitions. The experiments on two datasets with various architectural styles show, our approach is competitive in terms of segmentation accuracy, while at the same time is able to handle grammars with more complex structures at reduced runtime.

Future work is directed at reducing the search space for valid non-terminal configurations and further introducing depth and 3D information into the process.

Acknowledgments The authors gratefully acknowledge the generous support from the Austrian Research Promotion Agency (FFG) and Microsoft Photogrammetry for the research project CITYFIT (High-Quality Urban Reconstructions by Fitting Shape Grammars to Images and derived Textured Point Clouds), grant number 815971/14472.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building rome in a day. In *ICCV*, 2009.

[2] F. Alegre and F. Dellaert. A probabilistic approach to the semantic interpretation of building facades. Technical report, Georgia Institute of Technology, 2004.

[3] S. Becker and N. Haala. Grammar supported facade reconstruction from mobile lidar mapping. In *IAPRS*, 2009.

[4] A. Berg, F. Grabler, and J. Malik. Parsing images of architectural scenes. In *ICCV*, 2007.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001.

[6] L. Breiman. Random forests. *Machine Learning*, 2001.

[7] H. Chau, X. Chen, A. McKay, and A. de Pennington. Evaluation of a 3d shape grammar implementation. In *Design Computing and Cognition*, 2004.

[8] A. Dick, P. Torr, S. Ruffle, and R. Cipolla. Combining single view recognition and multiple view stereo for architectural scenes. In *ICCV*, 2001.

[9] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *CVPR*, 2009.

[10] D. Gallup, J. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR*, 2010.

[11] S. Kluckner, T. Mauthner, P. Roth, and H. Bischof. Semantic image classification using consistent regions and individual context. In *BMVC*, 2009.

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

[13] L. Ladicky, C. Russell, P. Kohli, and P. Torr. Associative Hierarchical CRFs for Object Class Image Segmentation. In *ICCV*, 2009.

[14] S. Lee and R. Nevatia. Extraction and integration of windows in a 3d building model from ground images. In *CVPR*, 2004.

[15] B. Micusik and J. Kosecka. Multi-view superpixel stereo in urban environments. *IJCV*, 2010.

[16] W. Mitchell. *The Logic of Architecture: Design, Computation, and Cognition*. 1992.

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

[18] N. Ripperda and C. Brenner. Application of a formal grammar to facade reconstruction in semiautomatic and automatic environments. In *AGILE*, 2009.

[19] M. Schlesinger and V. Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition*. 1990.

[20] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image category segmentation. In *CVPR*, 2008.

[21] G. Stiny and J. Gips. Shape grammars and the geneative specification of painting and sculpture. In *IFIP*, 1972.

[22] P. Sturgess, K. Alahari, L. Ladicky, and P. Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC*, 2009.

[23] J. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. Metropolis procedural modeling. *ACM Graphics*, 2011.

[24] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *CVPR*, 2011.

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

[26] A. Toshev, P. Mordohai, and B. Taskar. Detecting and parsing architecture at city scale from range data. In *CVPR*, 2010.

[27] C. Vanegas, D. Aliaga, and B. Benes. Building reconstruction using manhattan-world grammars. In *CVPR*, 2010.

[28] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. Instant architecture. In *ACM Graphics*, 2003.

[29] J. Xiao, T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan. Image-based facade modeling. In *SIGGRAPH Asia*, 2008.

[30] P. Zhao, T. Fang, J. Xiao, H. Zhang, Q. Zhao, and L. Quan. Rectilinear parsing of architecture in urban environment. In *CVPR*, 2010.