A Three-Layered Approach to Facade Parsing

Anđelo Martinović[1], Markus Mathias[1],
Julien Weissenberg[2], and Luc Van Gool[1,2]

[1] ESAT-PSI/VISICS, KU Leuven
[2] Computer Vision Laboratory, ETH Zurich

Abstract. We propose a novel three-layered approach for semantic segmentation of building facades. In the first layer, starting from an oversegmentation of a facade, we employ the recently introduced machine learning technique Recursive Neural Networks (RNN) to obtain a probabilistic interpretation of each segment. In the middle layer, initial labeling is augmented with the information coming from specialized facade component detectors. The information is merged using a Markov Random Field defined over the image. In the highest layer, we introduce *weak architectural knowledge*, which enforces the final reconstruction to be architecturally plausible and consistent. Rigorous tests performed on two existing datasets of building facades demonstrate that we significantly outperform the current-state of the art, even when using outputs from lower layers of the pipeline. In the end, we show how the output of the highest layer can be used to create a procedural reconstruction.

1 Introduction

One of the biggest challenges in 3D city modeling is the accurate reconstruction of building facades. For many applications, simple plane fitting and texturing is not enough. It is often essential to semantically identify the facade elements (windows, doors, balconies, etc.) and their layout. This task is not only difficult because of the vast diversity of buildings, but also because of shadows, occlusions and reflections.

The state-of-the-art methods for automated facade parsing assume that an appropriate shape grammar is available from the outset [1]. This assumes that one has strong prior knowledge about the structure of the facade, e.g. that it follows the Haussmann style and therefore a grammar restricted to that style can be invoked. Here, we make no such assumptions, yet get better results. The proposed approach can deal with a wide gamut of styles. Yet, as more restrictive knowledge should be used when available, we show that our method outperforms the state-of-the-art by a still larger margin in the presence of style information. Moreover, we demonstrate how procedural rules and thus shape grammars can be derived based on the segmentation, rather than vice-versa. This is an important step forward, avoiding the need for the prior, manual construction of style-specific grammars.

Our proposed facade parsing method consists of three distinct layers. In the first layer, a supervised training method learns the facade labeling based on an initial oversegmentation. For this purpose we utilize the recently developed

The original publication is available at www.springerlink.com.

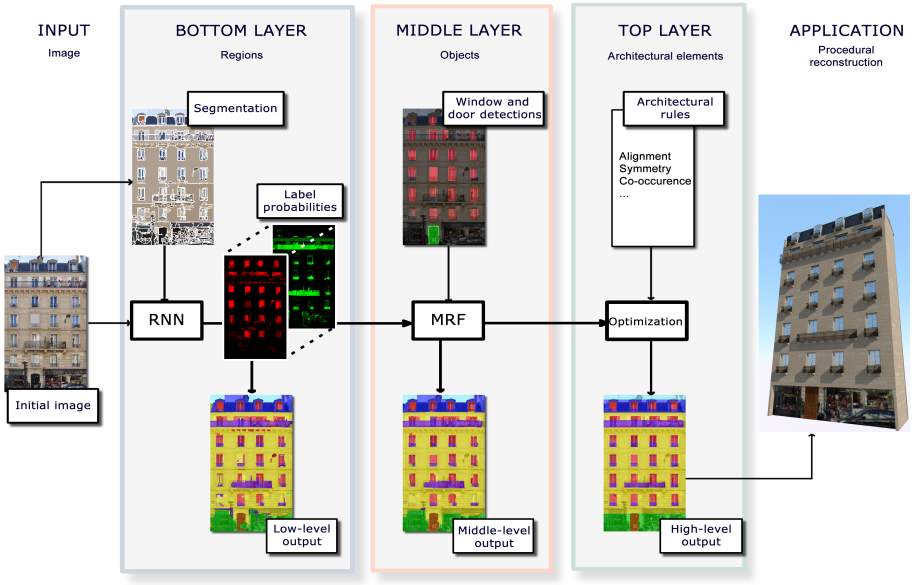


Fig. 1. The proposed three-layered approach to facade parsing.

Recursive Neural Networks (RNN) [2]. In the middle layer we introduce the knowledge about distinct facade elements, such as doors and windows. The raw RNN output is then combined with the information from object detectors trained to detect architectural elements (see Fig. 1). We pose the merging of RNN and detector output as a pixel labeling problem, modeled as a 2D Markov Random Field over the pixels. The multi-label MRF is then solved using graph cuts. Finally, the top layer introduces architectural concepts which encourage the facade reconstruction to assume a more structured configuration. This knowledge is encoded as a set of rules directly observable in the images, in contrast to shape grammar approaches, where some concepts (e.g. vertical alignment) are implicit or hidden in the grammar derivation. Our approach also enables modeling of irregular facades, as we use the architectural concepts as guidelines, not as hard constraints.

Our main contributions are as follows:

(1) a novel three-layer approach for facade parsing, utilizing low-level information from the semantic segmentation, middle-level detector information about objects in the facade, and top-level architectural knowledge; (2) a rigorous evaluation on two different datasets which shows that we outperform the state-of-the-art in facade parsing by a significant margin; (3) the concept of *weak architectural principles*, which introduce the high-level knowledge needed for making the final reconstruction architecturally plausible; (4) updated annotations for the facade dataset of [3], which are closer to the ground truth.

2 Related Work

There already exists a significant body of work on facade extraction and parsing from ground imagery. Zhao et al. [4] presented an algorithm that parses such imagery into buildings, grass and sky, followed by the partitioning of buildings into individual facades. This facade splitting problem was also studied by [5,6], where repetitive patterns provide cues to find the correct boundary between facades. Another approach was presented in [7], where a scene classification step identifies input images that contain facades. After automated image rectification, buildings are split into individual facades. We demonstrate on the eTRIMS [8] dataset that our approach can extract facades even in cluttered scenes. However our focus is the semantic segmentation of already isolated and rectified facades.

Xiao et al. [9,10] target realistic visualization without much semantic encoding in the reconstruction. Other authors attempt to infer structure starting from a set of basic elements, such as rectangles [11] or windows [12]. These approaches, however, also depend on the strong assumption of element repetition. Probabilistic approaches to building reconstruction started with the work of Dick et al. [13], where a building is assumed to be a 'lego' set of parameterized primitives. An expert is needed to set the model parameters and prior probabilities for full Bayesian inference.

For facade parsing, many approaches employ a procedural grammar, explicitly or implicitly. Müller et al. [14] detect symmetries and repetitions using Mutual Information to generate an instance of a procedural model. The approach was extended in [15], where images with strong perspective distortions are used to infer vanishing points and 3D information from a single image. Although these approaches produce good results in facade parsing, they assume strong priors on the input facades, i.e. that they consist of a rather regular window grid. Other grammar-based approaches include stochastic grammars [16], rjMCMC for the construction of a grammar tree [17], and hybrid bottom-up/top-down approaches [18]. Good results were reported by Teboul et al. [1], where facade reconstruction was postulated as a problem of finding the correct parameters of a pre-specified shape grammar. A random-walk algorithm was used to find the optimal values of the parameters. Recently, the approach was enhanced [19] with a novel optimization scheme, based on Reinforcement Learning. In this paper, we evaluate our system on the dataset from [1,19], for which we also provide a more precise set of annotations.

The benefit of relying on shape grammars is that they strongly restrict the search space during parsing. Yet, the grammar may not be expressive enough to cover the variance in real world data. Furthermore, an expert is needed to write the grammars for the relevant styles. Human intervention is also required to preselect the grammar appropriate for each specific building. The latter requirement can be mitigated by applying style classifiers [7] that automatically recognize the building style from low-level image features. Still, using a style grammar would imply it needs to be available beforehand, which at least for the moment is a limiting issue. It is easier to generate style classifiers than style grammars. Therefore, we chose not to assume there is such predefined grammar. In fact,

our guiding principle is to derive procedural grammars based on automatically parsed facades, rather than vice-versa. Some interactive work in that vein has already appeared. Aliaga et al. [20] infer simple grammatical rules from a user-given subdivision of a building. Bokeloh et al. [21] presented a framework applied on synthetic 3D data.

In summary, the current state-of-the-art in semantic facade parsing needs the prior specification of a style-specific grammar. Our aim is to outperform such systems, without needing such a grammar, allowing our approach to deal with a wider variety of buildings. Moreover, the order can thus be reversed by letting the image parsing control the grammar derivation, rather than using the grammar to support the image parsing. The latter selection can be automated by using style classifiers, which, as said, require far less human interaction than the prior construction of entire grammars.

3 Datasets description

We evaluate our approach on two datasets, the "Ecole Centrale Paris Facades Database" [3] and the eTRIMS database [8]. Since we are primarily interested in accurate modeling of building facades, we focus more on the ECP database, as it provides labels for multiple facade elements. To validate our approach, we show that we also outperform the state-of-the-art results reported on eTRIMS.

ECP Database contains 104 images of rectified and cropped facades of Haussmannian style buildings in Paris, with corresponding annotations. There are 7 different labels in the dataset. We define this set as $\Psi = \{$*window, wall, balcony, door, roof, sky, shop*$\}$. An earlier version of the dataset contained only 30 images but with very accurate annotations. Unfortunately, in the larger dataset images are labeled using a Haussmannian-style grammar. This results in a labeling closest to the ground truth given the restriction that the labeling is an instantiation of the grammar. As a consequence, these annotations are often imprecise or even plainly wrong. For example, windows that are not vertically aligned with the rest of the facade are not supported. We provide a new set of annotations that better fits the actual ground truth, which we present in the supplementary material. For evaluation purposes, we perform a 5-fold cross-validation on this dataset. In each fold, we use 60 images for training, 20 for validation, and 20 for testing.

eTRIMS Database contains 60 images, along with accurate pixel-wise annotations. In contrast to the ECP dataset, images are not rectified and the facades are not cropped. In order to compare our approach with the reported result from [22] we automatically rectify the input images using the algorithm of [23]. Furthermore, the labels of this dataset are quite different compared to the former dataset: $\Psi = \{$*building, car, door, pavement, road, sky, vegetation, window*$\}$. For evaluation, we perform a 5-fold cross-validation as in [22] with random subsampling of 40 images for training and 20 for testing. Note that we do not create a validation set due to the limited number of images in the dataset.

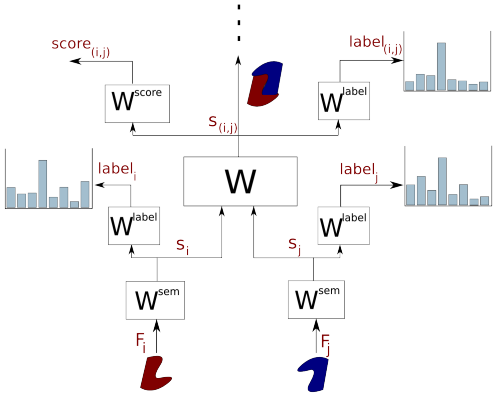


Fig. 2. Basic RNN structure. Two input segments are transformed into a semantic space and merged into a supersegment. The supersegment's semantic vector is recursively combined with other semantic vectors by repeating the same network structure.

4 Bottom Layer: Recursive Neural Network for Semantic Segmentation

Principles of RNN. The Recursive Neural Network [2] is a parsing algorithm designed to capture the recursive structure commonly found in natural and man-made scenes. Starting from an initial oversegmentation of an image, we use the network to create a binary parse tree of the whole image. This bottom-up approach is performed by recursively combining segments into supersegments until all the segments have been merged. See Fig. 2 for an illustration.

The starting point of our bottom layer is an oversegmentation of the image into small regions which represent objects or object parts. We extract features from these regions, and present them to the input layer of the RNN. This layer serves the purpose of transforming the input feature space to a semantic space of given dimensionality. The representation is computed by:

$$s_i = f(W^{sem} F_i + b^{sem}) \tag{1}$$

where F_i represents the input feature vector, W^{sem} the network's input layer, and s_i the semantic representation of the segment. b^{sem} is the bias, which we set to zero in all our experiments.

There are exponentially many possible parsing trees for a given oversegmentation. As no efficient dynamic programming solution exists for this problem, the RNN performs a greedy approximation. At each step of the process, all neighboring pairs of segments are considered for merging. The semantic vector of a supersegment created by merging a pair of segments is computed by:

$$s_{(i,j)} = f(W[s_i; s_j] + b) \tag{2}$$

where $s_i; s_j$ is a concatenation of the two semantic vectors, W is the merging layer of the RNN, and $s_{(i,j)}$ represents the semantic vector of the supersegment. The latter has the same dimensionality as the input segments, and it can be recursively combined with other segments.

We use an additional scoring layer of the network to compute the score of the semantic vector of the supersegment. This layer is trained to output a high score when two segments correspond to the same object (same label in the training data). Ideally, the network will learn to always combine segments of the same label before combining the neighboring objects to form the scene. The score of the supersegment is computed by:

$$score_{(i,j)} = W^{score} s_{(i,j)} \tag{3}$$

where W^{score} represents the scoring layer of the RNN.

In addition to computing the scores, we compute the class label of each segment. More precisely, a softmax layer is added on top of the semantic representation, which produces a probability distribution of the labels for the given segment. The likelihood of each pixel belonging to a label $\psi \in \Psi$ is:

$$P^{rnn}(\psi \mid c_i) = softmax\left(W^{label} s_i\right) \tag{4}$$

where c_i is a clique of pixels corresponding to the segment with the semantic representation s_i and W^{label} represents the class prediction layer of the network. We obtain the pixelwise merit $P^{rnn}(\psi \mid x_i)$ by evaluating the segment-wise likelihood on each pixel of the segment.

Implementation Details. We set the length of vectors in the semantic space to 50. We do not observe any significant improvement in the results if we use larger vectors, while the training time becomes much longer. However, using shorter vectors leads to a noticeable drop in performance. For the preparation of the data, we follow the approach of [2], with some modifications. First, the input image is oversegmented into regions using the mean-shift segmentation algorithm of [24]. We prefer to have a more fine-grained segmentation, so as not to combine different facade elements in a single region. On average, we obtain 643 regions per image (average image size is 600*400 pixels). Next, the appearance (color and texture), geometry, and location features are extracted for each region using the procedure of [25]. We use the default parameters from the implementation in STAIR Vision Library [26], which results in feature vectors of size 225.

Training. For the training of the network, we provide the oversegmented images, as well as the ground truth annotations. The training attempts to minimize the error between the parse tree proposed by the network and the set of correct parse trees which are defined by the annotations. A variant of backpropagation is used to find the model parameters: W^{sem}, W, W^{score} and W^{label}.

Interpreting the Results. After the training has finished, a query image is presented to the network. The trained RNN builds a parse tree for this image, assigning a score to each merger of the segments and a multinomial label distribution to each segment. We can then read out the probabilities in the leaves of the tree, and label the superpixels with the most likely label. However, we

keep the distributions $P^{rnn}(\psi \mid x_i)$ and propagate them to the higher levels of our pipeline, as they contain more information than the maximum-likelihood estimate.

5 Middle Layer: Introducing Objects Through Detectors

The RNN requires pre-segmented images as input, where the segmentation parameters are fixed for all images. Consequently, the results of the bottom layer depend on the noisy boundaries of the initial segmentation. By using object detectors we receive labeling information from a second source, so we can estimate better boundaries for elements such as doors and windows.

Window and Door Detection. We use our own GPU-based implementation of the Dollar's Integral Channel Features detector [27,28]. This detector provides state-of-the-art results for pedestrian detection and proves to be equally suited for the task of window and door detection (see Fig. 3). Following the settings in the original paper we use depth-2 decision trees boosted by discrete AdaBoost. For feature evaluation we use 6 gradient orientation channels, 1 gradient magnitude channel and the 3 LSV color channels. The training is performed on rectified images.

Detector Output. Each detector outputs a set of detections d_i, with respective scores r_i. We describe our algorithm on the example of the window detector. The label distribution of a pixel $x_i \in d_i$ in the test image can be set naively: 1 for the window label, 0 for others. However, in practice, a typical window detection will not only contain window pixels, but also pixels from other classes to a certain extent. We wish to capture this distribution by analysing all the detections in the training set. For each of the N detections in the training set, we create a probability distribution Q:

$$Q_l(d) = n_l / n_{total} \tag{5}$$

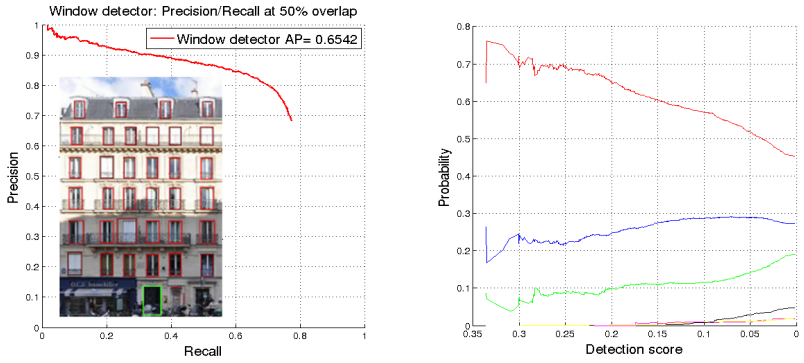
where d is the detection, n_l is the number of pixels in the ground truth corresponding to the label l, and n_{total} the total number of pixels within the bounding box. For each of the scores $r_i \in R$, we create a cumulative probability distribution:

$$P_c(r_i) = \frac{1}{\gamma} \sum_{r \in R : r \geq r_i} Q(d_r) \tag{6}$$

where R is the set of all detection scores in the training set, d_r denotes a detection with score r, and γ is the normalization constant that makes sure that the elements of $P_c(r_i)$ sum up to one. P_c is then a set of distributions which we visualize in Fig. 3(b). Finally, the label distribution of every pixel in the test image is given by:

$$P^{det}_{win}(x_i) = \begin{cases} P_c(NN(r)), & x_i \text{ is covered by } d_r \\ P_{uniform}, & \text{otherwise} \end{cases} \tag{7}$$

where $NN(r_i)$ represents the nearest-neighbor score in the training set. All pixels not covered by a detection are assigned a uniform probability distribution.



(a) (b)

Fig. 3. (a) Precision-recall curve of our window detector and an example image with detector output. (b) Visualization of the P_c look-up table for the window detector. Red color indicates *window* class, green *wall*, blue *balcony*. As the detection score decreases, the uncertainty between labels increases.

Incorporating Detector Knowledge With Markov Random Fields. To merge the information coming from the lower level of the pipeline and the middle level knowledge introduced by the detectors, we formulate a labeling problem by placing a 2D Markov Random Field over the image pixels. We seek to minimize the total energy, defined as the sum of unary potentials for each node, and the sum of all pairwise potentials between neighboring pixels:

$$E(\psi) = \sum_{x_i} \Phi_s\left(\psi_i \mid x_i\right) + \lambda \sum_{x_i} \sum_{x_j \sim\, x_i} \Phi_p\left(\psi_i, \psi_j \mid x_i, x_j\right) \tag{8}$$

where x_i is an image pixel, while the relation $\sim$ represents the 4-pixel neighborhood. Here, λ corresponds to the smoothing parameter, as the pairwise potentials follow the Potts model:

$$\Phi_p\left(\psi_i, \psi_j \mid x_i, x_j\right) = \begin{cases} 0, & \text{if } \psi_i = \psi_j \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

The unary potential of a pixel is a weighted sum of the low-level information (RNN labeling) and detector potentials.

$$\Phi_s\left(\psi_i \mid x_i\right) = -\log P^{rnn}\left(\psi \mid x_i\right) - \sum_{l \in \{win, door\}} \alpha_l \log P_l^{det}\left(\psi \mid x_i\right) \tag{10}$$

We solve this labeling problem using graph cuts [29] and obtain a solution depicted in Fig. 1.

Parameters. All detectors went through three iterations of training. For the first iteration we randomly select negative samples and the following two

iterations extend the set of negatives using bootstrapping. All positive samples are vertically mirrored to enlarge the training set size. 30000 randomly selected features build the feature pool for the AdaBoost algorithm. The window detector is trained on a diverse set of windows outside the ECP and eTrims datasets, which contains 2154 examples of windows. The door detector is trained on the same data as the RNN, respecting the 5-fold cross-validation splits.

There were only 3 parameters to be selected in this layer: the smoothing factor λ and the weights for the two detectors α_{win} and α_{door}. We estimated these parameters on a validation set for the ECP dataset. We set them to $\lambda = 6.5$, $\alpha_{win} = 0.75$ and $\alpha_{door} = 7$.

6 Top Layer: Weak Architectural Principles

In the first two layers we have not used any information about the facade structure. This reflects in results which, although convincing quantitatively, suffer from errors such as missing or misplaced facade elements, which makes it difficult to derive convincing models. To combat this problem, we introduce the concept of *weak architectural principles*, summarized in Table 1.

Table 1. Weak architectural principles used to complement the segmentation results of the first 2 layers. A "x" in the "alter" column denotes that the principle adjusts element borders. The principle may also remove or add new elements. Last two columns indicate which principles are used for each of the datasets.

Principle	Alter	Add	Remove	ECP	eTrims
(Non-)alignment: vertical and horizontal	x	-	-	x	x
Similarity of different *windows* of the same *facade*	-	x	-	x	x
Facade symmetry	-	x	x	x	x
Co-occurence of elements	-	x	x	x	-
Equal width/height in a row or column	x	-	-	x	-
Door hypothesis: first floor, touching ground	x	x	x	x	-
Vertical region order: {*shop**, *facade*$^+$, *roof**, *sky**}	x	-	-	x	-
Running *balcony* in the 2nd and 5th floor	x	x	x	x	-

The principles listed above are used to encode high-level architectural knowledge, and they can be directly evaluated in facade images. Some of them can be applied on a vast amount of different facades, while others may apply specifically for a certain architectural style. Furthermore, we can use the ground-truth labeling of the validation set to automatically deduce which principles should hold.

The principles we formulated mostly apply to objects in the facade (window, balcony, door). The initial bounding boxes of these objects are computed from the connected components of the pixelwise maximum over Φ_s. The minimal bounding rectangle $R = (x_1, y_1, x_2, y_2)$ around the connected components is the starting hypothesis for all elements.

The (non-)alignment principle is based on the observation that many facade elements are either exactly aligned or clearly off-center (see Fig. 4 left). We formulate this principle as an energy optimization problem where we estimate a locally optimal solution using the BFGS Quasi-Newton method. The energy for an object class is defined as:

$$E = \sum_{r_1,r_2 \in R} \Big(\omega_l \rho_{\tau_w}(x_1^{(r_1)} - x_1^{(r_2)}) + \omega_r \rho_{\tau_w}(x_2^{(r_1)} - x_2^{(r_2)}) + \tag{11}$$

$$\omega_t \rho_{\tau_h}(y_1^{(r_1)} - y_1^{(r_2)}) + \omega_b \rho_{\tau_h}(y_2^{(r_1)} - y_2^{(r_2)}) \Big) \tag{12}$$

$$\rho_{\tau_i}(z) = \begin{cases} \frac{\tau_i^2}{6}(1 - [1 - z/\tau_i]^2)^3, & \text{if } |z| \leq \tau_i \\ \frac{\tau_i^2}{6}, & \text{if } |z| > \tau_i \end{cases} \tag{13}$$

For each pair of rectangles the bounded influence function (13) rates their weighted top, bottom, left and right (ω_t, bottom ω_b, ω_l and right ω_r) alignment. The function has a constant value as soon as the distance between boundaries exceeds τ_i.

The similarity principle is applied similar to [30]. Every detected element votes for similar elements using an ISM-like voting scheme. Self similarity features [31] are calculated at Harris corner points. For all features inside the window bounding boxes, a vote vector to the center of the box is cast from the positions of the n nearest neighbors into a global voting space. The maxima of that voting space belong to both the initial detections and new detection hypotheses.

Harris corners are also used as a simple measure for vertical **symmetry**. The interest points are mirrored about a symmetry line hypothesis. A match is an interest point that has a mirrored counterpart. The maximum of the matches divided by the points under consideration defines the symmetry line and the symmetry score. If symmetry is detected (symmetry score $> \tau_{sym}$), symmetric elements are mirrored, overlapping ones are removed and for the remaining ones we have 3 possibilities: add a new mirrored element, remove or keep the existing one. The decision between adding or removing an element is based on the confidence score of each element, computed from Φ_s. If an element has a confidence higher than τ_{keep}, we simply keep it. We employ the same procedure for the 3 possibilities when we observe only one of two **co-occuring elements**.

If the principle of **equal width or height** of elements along a row or column holds in the validation set, this property will also be enforced in the testing set. If there has not been a door detection in the image, a **door hypothesis** is generated based on gradients in the probability map, average probability and relative estimated sizes to other facade elements.

The **vertical region order principle** searches for the wall area and optionally for shop, roof and sky in the given order. The split lines are optimized over the probability densities of the regions and the split lines between regions.

Parameters. In Equations 11 and 12 the parameter τ_w is set to half the median of the objects' width, and τ_h to half the median of the objects' height. With these settings, completely misaligned windows are not shifted by the minimization. The threshold τ_{sym}, for considering a facade as being symmetric, is set conservatively to the highest score of an actually non-symmetric facade in the validation set. The number n of the nearest neighbors for similarity voting is set to 10. τ_{keep} is also estimated from the validation set.

7 Results

We compare our results with the state-of-the-art algorithms reported on the two evaluated datasets. Table 2 shows the results that we obtain on each layer of the pipeline. Due to limited space, we report only the class accuracies, while the full confusion matrices may be found in the supplementary material. Figure 4 shows several examples of the final output of our system.

ECP Database. On the left side of Table 2, our results are compared with the approach of [19]. The baseline is run as described in [32]. We perform the 5-fold cross validation on the same dataset.

We see a clear performance boost already at the first layer. This improvement is in part due to better bottom-up features that we use, compared to the pixel level classifier used in [32]. Another improvement comes from the classifier itself: if we replace the RNN region classifier with an SVM, the results drop sharply by 13%. The middle layer increases the accuracies of window and door classes, as expected due to the introduction of object detectors. The drop in balcony class comes from the fact that they are often transparent, and the window detector finds the whole window structure partially occluded by the balcony. Furthermore, the usage of the smoothness term in the MRF slightly improves some of the other classes. By introducing high-level knowledge through the top layer, we obtain even better results in almost all of the classes (Table 2, left). However, when parsing the image on the highest level, pixel accuracy becomes an unreliable

Table 2. Results on ECP (left) and eTrims (right) dataset, for each of the layers of our system. Class accuracies are shown in percent, as well as the total pixel accuracy.

Class	Baseline[32]	Layer 1	Layer 2	Layer 3
window	62	62	69	**75**
wall	82	91	**93**	88
balcony	58	**74**	71	70
door	47	43	60	**67**
roof	66	70	73	**74**
sky	95	91	91	**97**
shop	88	79	86	**93**
Pixel acc.	74.71	82.63	**85.06**	84.17

Class	Baseline[22]	Layer 1	Layer 2	Layer 3
building	71	89	**90**	86
car	35	**67**	66	**67**
door	16	**24**	20	18
pavement	22	**35**	**35**	**35**
road	35	**47**	**47**	**47**
sky	78	**91**	90	**91**
vegetation	66	82	**83**	81
window	75	72	75	**80**
Pixel acc.	65.8	81.11	**81.94**	80.81

measure of performance, as slight displacements of elements might produce high error rates. Therefore, a qualitative evaluation would be more appropriate.

eTRIMS Database. The results we show for this dataset are obtained without using the door detector in the middle layer, as there was insufficient training data. After the 5-fold cross-validation, we obtain the results on the right of Table 2. Example output of the top layer may be seen in Fig. 5. It is clear that the bottom layer of our approach already outperforms the baseline method[22], by a margin of 15%. The only underperforming class is *window*. By introducing the window detector in the middle layer we increase the performance of the window class, and slightly decrease the accuracy of doors, due to the lack of a door detector and the fact that windows are often detected on door regions. The results of the top layer require some discussion. First, it is important to note that the buildings in this dataset come from different styles and have a huge variety of appearances. They also contain a significant amount of clutter and occlusions. Also, the only existing facade elements in this dataset are windows and doors. As we introduce additional windows through our architectural principles, it happens that we reconstruct a window occluded by vegetation, which explains the small drop in detection of vegetation class. Similarly, in some cases windows may be introduced or enlarged in the wall area, which accounts for the small decrease in building and door class accuracy, however still above the state-of-the-art.

Application: Image-based Procedural Modeling. The output of the top layer is used in a straightforward procedural modeling scenario. We define the elements of the facade to be the terminal symbols of a procedural grammar: *window*, *balcony* and *door*. As our top-layer segmentation output has a rather strong structure with straight boundaries, it is a simple task to subdivide the facade in a recursive way, following the borders between elements in the output. The subdivision is stopped at the terminal symbols. This subdivision is encoded as a set of splitting CGA rules in CityEngine [33]. A rendered model can be seen to the far right in Fig.1.

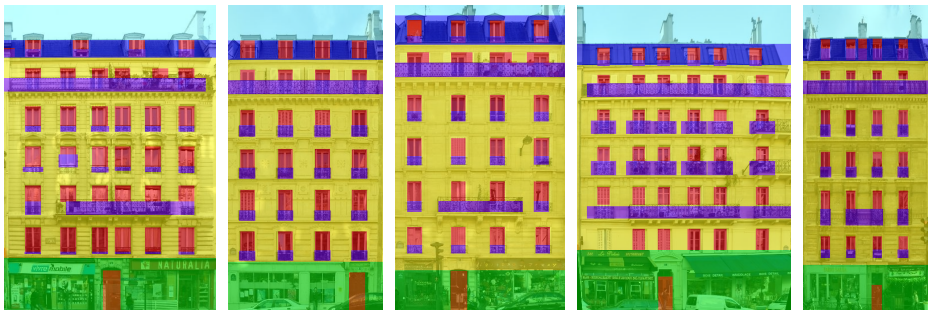


Fig. 4. Examples of top-layer output on various buildings from the ECP dataset.

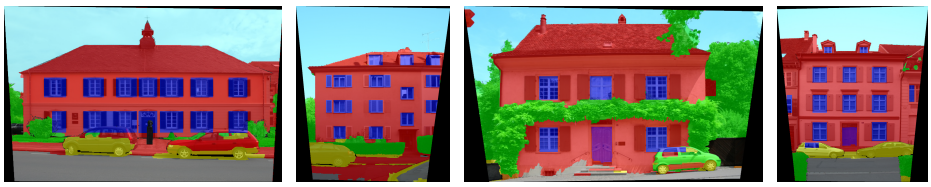


Fig. 5. Examples of top-layer output on various buildings from the eTrims dataset.

8 Conclusion and future work

We introduce a new method for facade parsing, operating on three levels of abstraction on a facade image. A bottom-up RNN semantic segmentation is utilized in the first layer, and then augmented with object detectors in the second layer. The introduction of a third, architectural layer, enables us to utilize the facade structure in order to make the final segmentation architecturally plausible. We show that our method clearly outperforms the current state-of-the-art on two different facade parsing datasets.

We also demonstrate how the output of our system can be used for image-based procedural modeling of facades. This enables us to infer procedural rules from the output of our system, instead of relying on a priori defined procedural shape grammars. However, the inferred rules are currently building instance-specific, and non-parametric. Therefore, the next logical step is to extend the approach to enable generalization between buildings of the same style. In that way the tedious task of defining grammars for different styles of buildings should be automated.

Acknowledgements. This paper was supported by EC FP7 Integrated Project 3D-COFORM, grant no. 231809, and ERC Advanced Grant VarCity.

References

1. Teboul, O., Simon, L., Koutsourakis, P., Paragios, N.: Segmentation of building facades using procedural shape priors. In: CVPR. (2010)
2. Socher, R., Lin, C.C., Ng, A.Y., Manning, C.D.: Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In: ICML. (2011)
3. Olivier Teboul: Ecole centrale paris facades database. `http://www.mas.ecp.fr/vision/Personnel/teboul/data.php` (2010)
4. Zhao, P., Fang, T., Xiao, J., Zhang, H., Zhao, Q., Quan, L.: Rectilinear parsing of architecture in urban environment. In: CVPR. (2010)
5. Wendel, A., Donoser, M., Bischof, H.: Unsupervised facade segmentation using repetitive patterns. In: DAGM. (2010)
6. Recky, M., Wendel, A., Leberl, F.: Façade segmentation in a multi-view scenario. In: 3DIMPVT. (2011)
7. Mathias, M., Martinovic, A., Weissenberg, J., Haegler, S., Gool, L.V.: Automatic architectural style recognition. In: 3D-ARCH. (2011)
8. Korč, F., Förstner, W.: eTRIMS Image Database for interpreting images of man-made scenes. Technical Report TR-IGG-P-2009-01 (April 2009)

9. Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., Quan, L.: Image-based façade modeling. In: SIGGRAPH Asia. (2008)
10. Xiao, J., Fang, T., Zhao, P., Lhuillier, M., Quan, L.: Image-based street-side city modeling. SIGGRAPH **28**(5) (2009)
11. Korah, T., Rasmussen, C.: Analysis of building textures for reconstructing partially occluded facades. In: ECCV. (2008)
12. Mayer, H., Reznik, S.: Mcmc linked with implicit shape models and plane sweeping for 3d building facade interpretation in image sequences. In: ISPRS. (2006)
13. Dick, A.R., Torr, P.H.S., Cipolla, R.: Modelling and interpretation of architecture from several images. IJCV **60** (2004)
14. Muller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. SIGGRAPH **26**(3) (2007)
15. Gool, L.J.V., Zeng, G., den Borre, F.V., Müller, P.: Towards mass-produced building models. In: PIA. (2007)
16. Alegre, O., Dellaert, F.: A probabilistic approach to the semantic interpretation of building facades. In: Workshop on Vision Techniques Applied to the Rehabilitation of City Centres. (2004)
17. Ripperda, N., Brenner, C.: Reconstruction of façade structures using a formal grammar and rjmcmc. In: DAGM. (2006)
18. Han, F., Zhu, S.C.: Bottom-up/top-down image parsing with attribute grammar. IEEE TPAMI **31**(1) (2009)
19. Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., Paragios, N.: Shape grammar parsing via reinforcement learning. In: CVPR. (2011)
20. Aliaga, D.G., Rosen, P.A., Bekins, D.R.: Style grammars for interactive visualization of architecture. TVCG **13**(4) (2007)
21. Bokeloh, M., Wand, M., Seidel, H.P.: A connection between partial symmetry and inverse procedural modeling. SIGGRAPH **29**(4) (2010)
22. Yang, M.Y., Förstner, W.: Regionwise classification of building facade images. In: PIA. LNCS 6952, Springer (2011)
23. Liebowitz, D., Zisserman, A.: Metric rectification for perspective images of planes. In: CVPR. (1998)
24. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE TPAMI **24**(5) (2002)
25. Gould, S., Fulton, R., Koller, D.: Decomposing a scene into geometric and semantically consistent regions. In: ICCV. (2009)
26. Gould, S., Russakovsky, O., Goodfellow, I., Baumstarck, P., Ng, A.Y., Koller, D.: The stair vision library (v2.2). `http://ai.stanford.edu/~sgould/svl` (2009)
27. Dollar, P., Tu, Z., Perona, P., Belongie, S.: Integral channel features. In: BMVC. (2009)
28. Benenson, R., Mathias, M., Timofte, R., Van Gool, L.: Pedestrian detection at 100 frames per second. In: CVPR. (2012)
29. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE TPAMI **23**(11) (2001)
30. Mathias, M., Martinovic, A., Weissenberg, J., Gool, L.V.: Procedural 3d building reconstruction using shape grammars and detectors. In: 3DIMPVT. (2011)
31. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: CVPR. (2007)
32. Teboul, O.: Shape Grammar Parsing: Application to Image-based Modeling. PhD thesis, Ecole Centrale Paris (2011)
33. Procedural: CityEngine. `http://www.procedural.com/` (2010)