Open access • Journal Article • DOI:10.1007/S11263-015-0868-Z

ATLAS: A Three-Layered Approach to Facade Parsing — Source link

Markus Mathias, Andelo Martinovic, Luc Van Gool

Institutions: Katholieke Universiteit Leuven

Published on: 01 May 2016 - International Journal of Computer Vision (Kluwer Academic Publishers)

Topics: Facade, Procedural modeling and Segmentation

Int J Comput Vis (2016) 118:22–48
DOI 10.1007/s11263-015-0868-z

ATLAS: A Three-Layered Approach to Facade Parsing

Markus Mathias[1] · **Anđelo Martinović**[1] · **Luc Van Gool**[1]

Received: 30 July 2013 / Accepted: 22 October 2015 / Published online: 20 November 2015

Abstract We propose a novel approach for semantic segmentation of building facades. Our system consists of three distinct layers, representing different levels of abstraction in facade images: segments, objects and architectural elements. In the first layer, the facade is segmented into regions, each of which is assigned a probability distribution over semantic classes. We evaluate different state-of-the-art segmentation and classification strategies to obtain the initial probabilistic semantic labeling. In the second layer, we investigate the performance of different object detectors and show the benefit of using such detectors to improve our initial labeling. The generic approaches of the first two layers are then specialized for the task of facade labeling in the third layer. There, we incorporate additional meta-knowledge in the form of *weak architectural principles*, which enforces architectural plausibility and consistency on the final reconstruction. Rigorous tests performed on two existing datasets of building facades demonstrate that we outperform the current state of the art, even when using outputs from lower layers of the pipeline. Finally, we demonstrate how the output of the highest layer can be used to create a procedural building reconstruction.

Communicated by Yasuyuki Matsushita.

Markus Mathias and Anđelo Martinović have contributed equally to this work.

✉ Markus Mathias
markus.mathias@esat.kuleuven.be

Anđelo Martinović
andelo.martinovic@esat.kuleuven.be

Luc Van Gool
luc.vangool@esat.kuleuven.be

[1] K.U. Leuven, Kasteelpark Arenberg 10, Box 2441, 3001 Heverlee, Belgium

Keywords Semantic segmentation · Facade parsing · Procedural modeling

1 Introduction

The accurate reconstruction of building facades plays an important role in 3D city modeling. Current models built by simple plane fitting and texturing are a good starting point, but provide inadequate 3D visual perception. For instance, artifacts in the 3D shape often show up during unrestricted user movement around the model. Due to diversity of appearance, hierarchical structure of scene objects and the lack of implementing long-range interactions, it appears impossible that improved, bottom-up depth extraction and primitive fitting alone can avoid such artifacts from sneaking in. Furthermore, conventional bottom-up models based on structure from motion lack any semantic knowledge about the scene. Yet, adding a good understanding of what needs to be modeled is a strong cue, not only to improve the visual and 3D quality of the model, but also to substantially widen its usage (e.g. for animation where people should walk through doors, not walls, when wanting to know the average number of floors that the buildings in a street have, etc.). Figure 1 shows an example of our modeling pipeline, that builds on the inclusion of semantic aspects.

Conversely, *procedural modeling* provides an effective way to create detailed and realistic 3D building models that do come with all the semantic labels required. These models are typically generated by iteratively applying procedural shape grammar rules on a starting shape, e.g. a building footprint. Each rule adds more detail to the result of the previous. The resulting models support the addition of visually crucial effects such as window being reflective, balconies to protrude, etc.

Fig. 1 The input to our system is cropped and rectified facade image (*left*). We process the image by our three layers to produced a labeled output image (*middle*). From this output we produce a textured procedural model (*right*)

The goal of creating procedural models for *existing* buildings from images or other data thereof, has been coined *inverse procedural modeling*. An early attempt can be found in Müller et al. (2007). Such inverse procedural modeling needs to select the appropriate rules from the style grammar, as well as their parameter settings. As the corresponding search space is huge, solutions typically start from a preprocessed version of the raw data. The semantic segmentation of facades—also referred to as facade parsing—is a good example. This said, such accurate labeling of facade elements (such as windows, doors or balconies) is a difficult problem in its own right, given the great diversity of buildings and the interference of factors like shadows, occlusions and reflections in the images. It is this facade parsing that this paper focuses on. Furthermore, a shape grammar specific to the desired style is not easy to come by. An expert in that style needs to sit down with a person versed in the creation of the grammars. Therefore, our approach also avoids the need for such a style-specific grammar and uses generic architectural principles instead. This stands in contrast to most earlier inverse procedural modeling work (see e.g. Teboul et al. 2013). Assuming that the input facades are of a certain architectural style helps to keep the dimensionality of the search space a bit smaller. In the case of Teboul et al. (2013), this is e.g. the Haussmannian style, ubiquitous in central Paris. Strong prior knowledge about this style is imbued in the Haussmann-specific procedural facade grammar.

This paper extends our previous work (Martinović et al. 2012), which achieves top results on the task of facade parsing, even without using any style-specific prior knowledge. Still, if style information is available, it can be incorporated into the system through the usage of extra "architectural principles". In contrast to full procedural grammars, these principles do not encode the entire facade structure and can be formulated explicitly by laymen. Moreover, we demonstrate how procedural rules and thus simple shape grammars can be derived from facade labeling, rather than vice-versa. By avoiding the need for a style prior, we circumvent the manual construction of style-specific grammars.

Our approach to facade parsing is performed in three layers, representing different levels of abstraction in facade images: segments, objects and architectural elements. An overview is given in Fig. 2.

Bottom layer Initially, the facade is segmented into superpixels, i.e. image regions. Visual features are extracted from the corresponding regions, and subsequently used for classification. Each region is assigned a probability distribution over semantic classes. In this layer, we pay particular attention to the evaluation of different segmentation algorithms and classifiers on the task of semantic segmentation of facades, as well as the effect of segmentation coarseness on the classification performance.

Middle layer The second layer of our approach introduces detectors for objects found in urban scenes, such as windows and doors. The classifier output from the bottom layer is combined with the object detector responses (see Fig. 2) and results in our improved middle layer output. The combination of detections and the labeling from the bottom layer is achieved through a 2D conditional random field defined over the image, which can be efficiently solved with graph cuts. We investigate the performance of different object detectors and show the benefit of using such detectors to improve our initial labeling.

Top layer The generic approaches in the first two layers are complemented with considerations dedicated to the task of facade labeling. In the top layer, we incorporate additional meta-knowledge in the form of *weak architectural principles*. In contrast to shape grammar rules, these principles are easily observable in the images. For instance, the principle of vertical window alignment is often an implicit consequence of grammar rules, never made explicit in any of them. Also, we use these architectural concepts as guidelines, not as hard constraints. Therefore, we are also able to model irregular facades, as demonstrated on the eTRIMS dataset that contains different facade styles. The architectural principles are designed such that each principle either proposes new facade elements, re-arranges their position, or evaluates the current configuration of elements. Finally, we pose the search for the optimal facade labeling as a sampling-based approach. Although the overall pixel accuracy of the semantic segmentation is not greatly influenced by the top layer, we obtain image labelings that are visually more pleasing, with clearly defined object boundaries and structures. These in turn form a stronger basis for further

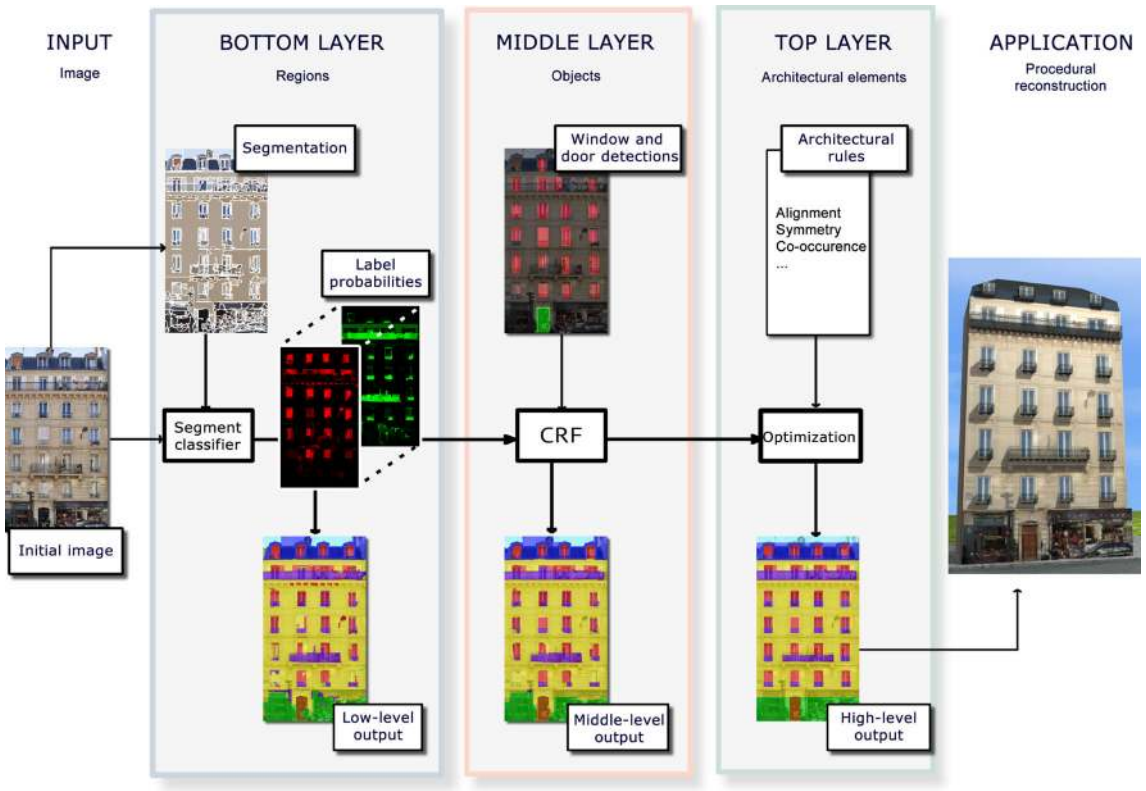

Fig. 2 The proposed three-layered approach to facade parsing

processing, e.g. for deriving style-specific procedural grammars.

While the overall structure of our system is similar to that of Martinović et al. (2012), each layer has been upgraded. In the bottom layer, instead of using a fixed combination of Mean-shift (Comaniciu and Meer 2002) segmentation and the Recursive Neural Network classifier (Socher et al. 2011), we evaluate various segmentation and classification algorithms. In the middle layer, we learn a prior on element locations and calculate the probabilistic detector output in a more robust way. Furthermore, we learn the CRF parameters with structured SVMs (Tsochantaridis et al. 2005). In the top layer, we propose a coupled subsampling-and-optimization technique in a generic framework that allows for addition of new principles.

Our main contributions are as follows:

(1) a new approach for facade parsing, combining low-level information from the semantic segmentation, middle-level detector information about objects in the facade, as well as 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; (3) the concept of *weak architectural principles*, which introduce the high-level knowledge needed for ensuring architectural plausibility.

2 Related Work

This section concisely describes the relation between the proposed work and prior art. We have organized this overview into several main topics.

Scene parsing There exists a significant body of work in this field. Some approaches attempt to estimate labels for each pixel in the image (Shotton et al. 2009; Fröhlich et al. 2013). Others depend on an initial segmentation of the image into super-pixels. Visual features are extracted from the corresponding patches or regions, and subsequently used for classification. In our work, we opt for the region-based approach in the first layer, as state-of-the-art results in semantic scene segmentation are achieved by similar approaches.

These approaches ensure labeling consistency by incorporating region context in various ways: estimating geometric labels (Gould et al. 2009a; Tighe and Lazebnik 2013b), using multiple over-segmentations (Kumar and Koller 2010), learning segmentation trees (Socher et al. 2011) or label transfer combined with a simple MRF (Liu et al. 2011; Tighe and Lazebnik 2013b). However, facade structures are difficult to analyse with solely region-based approaches, as the initial segmentation boundaries might not correspond to actual object boundaries in the image. Our work puts more empha-

sis on the combination of the region-based approach with higher-level information, such as object detectors and architectural knowledge.

Combining semantic segmentation with object detectors The effect of positive reinforcement between semantic segmentation and object detection approaches has been demonstrated in several works. Heitz and Koller (2008) use image regions as context for improved object detection. This is an orthogonal approach to our work, as we use object detectors to improve the facade labeling. Joint reasoning about pixel-wise labeling and object detectors in a CRF framework was performed in Wojek and Schiele (2008), while also capturing temporal consistency for video sequences. However, the complexity of their CRF requires slow approximate inference with loopy belief propagation. The work of Ladicky et al. (2010), later extended by Floros et al. (2011), disregards the temporal consistency, but in their CRF framework inference can be performed efficiently via graph cuts. The second layer of our approach is similar to Ladicky et al. (2010), but with two key differences. Firstly, instead of using detector outputs as higher-order potentials, we decompose them into unary potentials, which are learned based on detector output on the training set. This enables us to solve a much simpler CRF optimization problem. Note that the problem of inferring pixel-level cues (or masks) from bounding boxes can also be tackled by using per-exemplar detectors as in Tighe and Lazebnik (2013a) if the objects exhibit high variability in appearance. The second advantage of our approach is that we can efficiently learn the CRF parameters on the validation set based on the structured SVM approach of Tsochantaridis et al. (2005). As shown in Szummer et al. (2008), CRF parameter learning using graph cuts is tractable, fast, and much more efficient than methods based on cross-validation, especially for larger parameter vectors.

Urban reconstruction For an extensive overview of the field, we refer the reader to the survey of Musialski et al. (2012). Our main focus is the semantic segmentation of isolated and rectified facades. These can be obtained from more general street-side imagery by approaches such as Zhao et al. (2010), Wendelet al. (2010), Recky et al. (2011), Mathias et al. (2011b). Furthermore, we demonstrate that even in cluttered scenes with occlusions such as vegetation or cars, our approach can semantically segment the facades.

Xiao et al. (2008, 2009) target realistic visualization with a low level of semantic encoding in the reconstruction. In their work, facades are represented with planes or simple developable surfaces. On the other hand, many approaches employ higher-order knowledge for building reconstruction. Probabilistic approaches to building reconstruction started with the work of Dick et al. (2004), where a building is assumed to be a 'lego' set of parameterized primitives. The inference is performed using a Reversible Jump Markov Chain Monte Carlo (rjMCMC) approach. However, an expert is needed to set the model parameters and prior probabilities. In contrast, the free parameters of our system are learned from validation data.

Certain approaches are based on priors on the facade layout. A grid-based layout is a common assumption (Korah et al. 2008; Shen et al. 2011; Yang et al. 2012; Han et al. 2012). The work of Müller et al. (2007) also assumes a certain degree of facade regularity, and fits procedural grammar rules to the detected subdivision of the facade. Unlike the aforementioned methods, our approach poses no grid constraints on the facade.

Grammar-based approaches are quite popular in the field (Alegre and Dellaert 2004; Ripperda and Brenner 2006; Han and Zhu 2009). They allow the generation of very clean models and labeling results, often demonstrated by approaches where facade reconstruction is postulated as a problem of finding the correct parameters of a pre-specified shape grammar (Teboul et al. 2010, 2013). Depth cues have also been used in the context of grammar-based parsing by Simon et al. (2012), transforming the problem into a multiobjective optimization, solved with a genetic algorithm. In our work, we advocate the usage of weak architectural principles, a more flexible approach than using predefined grammars.

Object detection has also been considered in grammar-based approaches. In Mathias et al. (2011a), 3D reconstructions of Greek Doric temples are created using a specialized procedural grammar, 3D Structure-from-Motion (SfM) point clouds, and object detectors. Several approaches use detector outputs to augment the bottom-up merit functions for grammar-based facade parsing. Ok et al. (2012) use a simple approach where the merit of undetected classes is zeroed out in every detection. In a work similar to our first two layers, Riemenschneider et al. (2012) combines a pixel-wise classifier with Hough forest detectors using a MRF framework. This labeling is then used to create an irregular grid which is labeled by using a predefined grammar. In contrast to this work, we utilize much stronger bottom-up classifiers and detectors, without restricting the final output to a grid.

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 pre-select the grammar appropriate for each specific building. The latter requirement can be mitigated by applying style classifiers (Mathias et al. 2011b) that automatically recognize the building style from low-level image features. Still, using a style-specific grammar would imply that it needs to be available beforehand, which at least for the moment is a limiting issue. Therefore, in the earlier

version of this work (Martinović et al. 2012), we did not assume the existence of such a predefined grammar. Other authors have also recognized the limitation of relying on expert-written procedural grammars, e.g. (Dai et al. 2012), replacing them with weaker, or learned priors. 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. (2007) infer simple grammatical rules from a user-given subdivision of a building. Bokeloh et al. (2010) presented a framework applied on synthetic 3D data. Very recently, approaches that perform automatic grammar induction from labeled images have been proposed (Martinović and Van Gool 2013; Weissenberg et al. 2013; Wu et al. 2013; Zhang et al. 2013).

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 be reversed by letting the image parsing control the grammar inference, rather than using the grammar to control the process of 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

Our approach is evaluated on two datasets, the "Ecole Centrale Paris Facades Database Benchmark 2011" (Teboul 2010) and the eTRIMS database (Korč and Förstner 2009). The ECP database provides labels for multiple facade elements, while the eTRIMS dataset also contains non-building classes, such as vegetation. Since we are primarily interested in the accurate parsing of building facades, our main focus will be on the ECP database. We additionally validate our approach on eTRIMS and show that we outperform previous state-of-the-art results.

The ECP Database contains 104 annotated images of single rectified and cropped facades in the Haussmannian style. The dataset has 7 different labels $\Psi = \{$*window, wall, balcony, door, roof, sky, shop*$\}$. We use the new and more precise set of annotations provided by Martinović et al. (2012). Our evaluations are performed with a fivefold cross-validation on this dataset. For each fold, we use 60 images for training, 20 for validation, and 20 for testing.

The eTRIMS Database provides accurate pixel-wise annotations and contains 60 images. Unlike the ECP dataset, the images are not rectified and the facades uncropped. We use the automatic rectification algorithm of Liebowitz and Zisserman (1998) as a preprocessing step. To allow for a fair comparison to previously reported results, we un-rectify our output prior to evaluation. The labels of this dataset $\Psi = \{$*building, car, door, pavement, road, sky, vegetation, window*$\}$ are quite different compared to the ECP dataset, as there are several non-building classes. As in Yang and Förstner (2011b), we evaluate our algorithm by performing a fivefold cross-validation with random sub-sampling. However, instead of using 40 images for training, we use only 30, leaving 10 images as a validation set. 20 images are used for evaluation.

4 Bottom Layer: Initial Semantic Segmentation

The purpose of the bottom layer is to provide the initial classification of each pixel into one of the semantic classes. As a single pixel does not contain enough information for accurate classification, one must consider its context.

In a *patch-based* approach (e.g. the baseline of Teboul et al. (2010)) the context of a pixel is an image patch of certain size, centered on the pixel. Each pixel is then classified separately, based on the features extracted from the corresponding patch. The downside of this method is that the final result can be quite noisy, since neighboring pixels can be assigned to completely different classes.

Another approach is to use *regions* (super-pixels), i.e. to segment the image in coherent regions, which ideally share the same semantic label. Classification is then performed on the region level, which provides three main advantages over the patch-based approach. First, since all pixels within a region share the same class, the result is generally less noisy. Second, the dimensionality of the problem is significantly reduced as the number of regions in the image is typically two orders of magnitude lower than the number of pixels. Third, coherent regions can provide a stronger clue for a classifier e.g. by their specific shape. Yet, any errors in the segmentation step will propagate to the classification, since the final labeling is restricted to follow the super-pixel boundaries.

In our work, we opt for a region-based approach, as state-of-the-art results in semantic scene segmentation have been achieved by similar approaches (Gould et al. 2009a; Tighe and Lazebnik 2013a; Kumar and Koller 2010). Our experiments validate this choice, as we show in Sect. 7. The implementation of a region-based classification approach consists of three steps: segmenting the images into regions, extracting features from the regions, and using a classifier to obtain probabilistic estimates of classes, or labels, for each region. In this section we investigate how different segmentation algorithms and classifiers affect the speed and quality of facade labeling.

4.1 Image Segmentation

One of the most important choices in region-based segmentation is the number of regions created. We define the *maximum achievable accuracy (MAA)* as the accuracy (pixel-average or class-average) obtained by using an oracle classifier, which assigns each region the label of the pixel majority in the ground truth. Clearly, a pixel-based oracle classifier achieves the *MAA* of 100 %, since every pixel is classified separately. By using region-based segmentation we introduce the constraint that all pixels in a single region share the same class. On the one hand, a more fine-grained segmentation tends to result in a higher *MAA*. On the other hand, classifiers tend to perform better on discriminative and therefore larger regions. Even though coarse-grained segmentation is better suited for classification purposes, this process introduces errors when semantically different regions merge together, which reduces the *MAA*. Intuitively, finding a good segmentation of the image is equivalent to discovering the optimal trade-off between region size and discrimination potential.

Over the years, a large number of image segmentation algorithms have been developed (Comaniciu and Meer 2002; Felzenszwalb and Huttenlocher 2004; Achanta et al. 2010; Arbelaez et al. 2011; Van den Bergh et al. 2012). In this work, we chose to evaluate three dissimilar algorithms on the task of facade segmentation. The first, *Mean-shift* (Comaniciu and Meer 2002), is a popular algorithm that was demonstrated to perform well for facade parsing in the previous version of this paper (Martinović et al. 2012). Second, we evaluate one of the fastest segmentation algorithms to date, *SEEDS* (Van den Bergh et al. 2012). This method was shown to have competitive results while running in real-time. Finally, the third algorithm in our comparison is *gPb* by Arbelaez et al. (2011), which sacrifices running time for an accurate calculation of the segmentation tree. In order to perform a fair comparison to the other algorithms, we consider only a single level in the gPb tree.

4.2 Feature Extraction

We use the same feature extraction algorithm in all of our experiments. Following the procedure of Gould et al. (2009a), we extract appearance (color and texture), geometry, and location features for each region. This choice was motivated by the fact that the same features are used in several top-performing scene segmentation approaches (Gould et al. 2009a; Kumar and Koller 2010; Socher et al. 2011). Additionally, the publicly available implementation in form of the Stair Vision Library (Gould et al. 2009b) enables us to quickly extract features from pre-segmented facade images. With default parameters, this results in feature vectors of size 225.

4.3 Classifiers

Given its feature vector, each region needs to be assigned to one of the semantic classes described in Sect. 3. We consider five different multinomial classifiers:

1. LOG: Multiclass logistic regression classifier (Gould et al. 2009b)
2. CRF: An extension of LOG (Gould et al. 2009b)
3. MLP: Multilayer Perceptron (Demuth and Beale 1993)
4. SVM: Multiclass Support Vector Machine (Chang and Lin 2011)
5. RNN: Recursive Neural Network (Socher et al. 2011)

The classifier output is a confidence score for each class. These scores can be transformed into a probability distribution using a softmax function.

For the first two methods, a boosted one-vs-all classifier is learned for each class using Adaboost. Then, the outputs of the classifiers are used as features for learning the multiclass logistic model (LOG) with a linear predictor function. The CRF model is obtained by adding a pairwise term between neighboring segments, which has a smoothing effect. For more details about the implementation, please consult Gould et al. (2009b). The multilayer perceptron we use is a feed-forward artificial neural network with a single input, hidden and output layer. The number of neurons in the input layer is 225, equal to the number of features. The output layer contains as many neurons as there are semantic classes. Using a rule-of-thumb that states that the optimal size of the hidden layer is usually between the size of the input and the size of the output layers, we set the number of hidden neurons to 75. As the SVM classifier we use the publicly available one-vs-one multiclass SVM with a Gaussian kernel function (Chang and Lin 2011). The parameters C and γ are determined from the validation set. Finally, the RNN classifier was shown to perform well for the semantic segmentation of general scenes (Socher et al. 2011) and building facades (Martinović et al. 2012). In line with (Martinović et al. 2012), we set the length of vectors in the semantic space to 50.

4.4 Analysis

By setting the average number of regions per image to a fixed value, we evaluate the interplay between different segmentations and classifiers. Second, we select the best combination of segmentation algorithms and classifiers, and investigate how changing the number of segments affects the classification accuracy. For completeness, we calculate both pixel-wise (PW) and class-wise (CW) accuracies. The former is defined simply as the percentage of correctly classified pixels. We define the CW accuracy as the unweighted

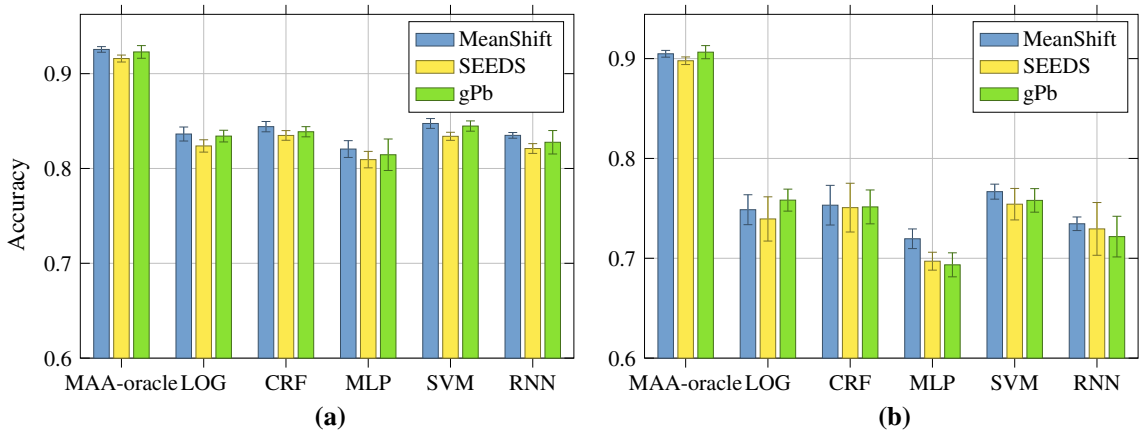

Fig. 3 **a** Pixel-wise and **b** class-wise accuracy of different segmentation algorithms and classifiers on the ECP dataset. The results are calculated as the mean, and *error bars* as the standard deviation of results calculated from five cross-validation folds

average of all class accuracies (the latter being the % of pixels of a class that were correctly classified), which provides an insight into classification performance on smaller classes. All of the presented experiments are performed on the ECP dataset.

4.4.1 Segmentation and Classification

Keeping the average number of segments per image equal for all three segmentation algorithms ($\sim$ 690 segments), we evaluate the maximum achievable accuracy, as well as classification accuracy achieved with each of the classifiers from Sect. 4.3. The results obtained are shown in Fig. 3.

Generally, using SEEDS as the segmentation algorithm results in the lowest classification accuracy. However, the difference between SEEDS and its competitors is relatively small (around 1 %), and one may opt to use SEEDS when speed is of the essence (as it may very well be when dealing with complete city modeling). Mean-shift and gPb performed similarly in each of the five classifier scenarios. Since Mean-shift segmentation is much faster to compute than gPb, we select it as our preferred segmentation algorithm.

Additionally, the data reveals that our method is quite robust with respect to the choice of classifier. As expected, there is a noticeable difference between the maximum achievable accuracy (MAA) and the results obtained with the five classifiers. The gap becomes even more apparent when considering class-wise accuracies. This is due to the unbalanced datasets, where the number of pixels of each class label varies significantly. By definition, class-wise accuracy disregards this variation.

We can see that the CRF model benefits from the addition of pairwise terms, compared to the LOG classifier. RNN outperforms the basic MLP model, but the results do not justify the extremely long training time of RNN (around 24 h). Unlike other methods, which classify each of the segments separately, RNN also creates a hierarchical parse tree of the image by recursively combining neighboring segments. However, existing RNN-based approaches (Socher et al. 2011; Martinović et al. 2012) do not exploit any knowledge from the tree during classification. Additionally, we achieved no improvement by using higher levels of the hierarchy, raising further questions about the usefulness of the tree. Finally, the SVM classifier emerges as the winner, as it achieves better results than its competitors both in terms of pixel-wise and class-wise accuracy.

One may argue that although the SVM classifier has the best performance in the first layer, some other classifier might provide better bottom-up information to the other layers of the system. We tested this hypothesis with the CRF and RNN classifiers, but obtained no improvement over SVM.

4.4.2 Number of Segments

The results in the previous section were obtained by setting the average number of segments per image to a fixed value. Now we evaluate the effect of changing the segmentation coarseness while fixing the best performing segmentation—classification pair, i.e. Mean-shift and SVM. By changing the minimum region size parameter in the Mean-shift implementation, we obtain seven different levels of coarseness, ranging from 1906 to 283 segments per image.

The classification results in Fig. 4 show that the maximum achievable accuracy steadily drops as we use coarser and coarser segmentations. The classifier performance follows a different trend, as its performance peaks around an optimal number of segments. While large segments introduce errors by combining neighboring objects into single regions, fine segmentations produce small image regions which are not discriminative enough for the classifier. However, this effect

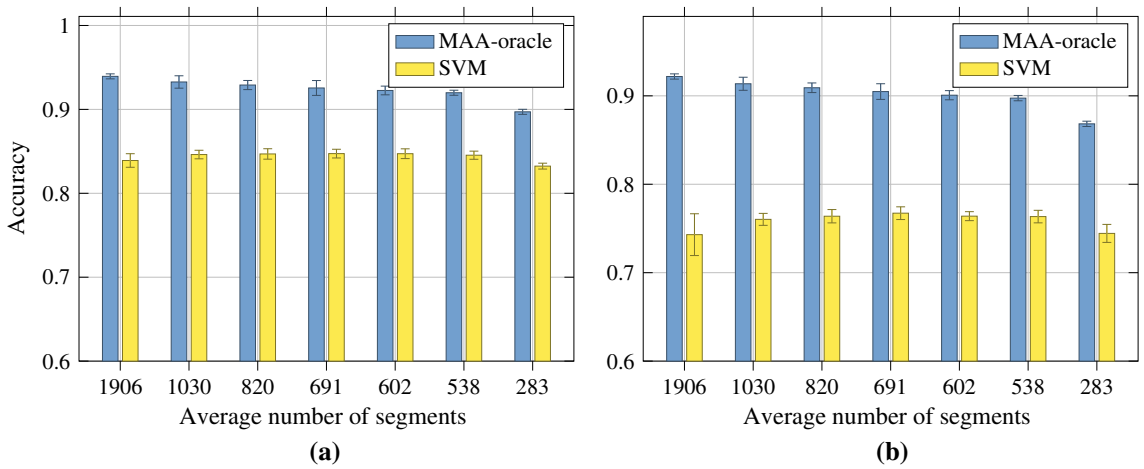

Fig. 4 The effect of segmentation coarseness on **a** pixel-wise and **b** class-wise accuracy of the oracle and SVM classifier on the ECP dataset

is prominent only when dealing with rather extreme numbers of segments, as we obtain similar results from 500 to 1000 segments per image. Therefore, we selected the middle level of coarseness in Fig. 4, amounting to 691 segments per image, on average.

5 Middle Layer: Introducing Objects Through Detectors

In the middle layer, we enrich our labeling pipeline by localizing facade elements directly through the usage of object detectors. Such detectors search for coherent structures that can span several of the previously segmented regions, thus allowing better discrimination. In this section, we demonstrate how detectors are integrated into our system and argue that their usage benefits the overall labeling quality.

Our bottom layer provides a probability distribution over labels for each region (segment) in the image. These regions are determined using fixed segmentation parameters for all input images. As shown in Fig. 3, even with the perfect *MAA* oracle, we can maximally reach 92 % pixel accuracy and 90 % class accuracy. By using object detectors in the second layer, we not only provide information from a second source, but also allow our final labeling not to be constrained by the initial segmentation boundaries. This is especially apparent for the case of window detection, where the initial object boundaries often do not coincide with image gradients.

From all classes present in the ECP and eTRIMS datasets, some are best discriminated by their texture and color (*sky*, *grass*, *road* ...). Other classes, such as *window*, *door* and *car*, are characterized by their distinctive shapes and sizes, and can therefore be discovered by classical object detectors. For these three object categories we trained object detectors, explained in more detail later, with training data coming from different sources. The total number of windows in the ECP dataset is large enough to train a detector which is **specific** to the Haussmannian style. Training a style-specific detector also benefits from the fact that Haussmannian windows and doors samples do not show much variance in appearance. In contrast, the eTRIMS dataset does not follow a fixed architectural style and shows a high variance of object appearances. At the same time, eTRIMS contains fewer samples per object class (1016 for windows, 85 for doors and 67 for cars). The higher variation combined with fewer examples makes training reasonable detector models by using only eTRIMS data infeasible. Therefore, we used data from an outside source to train style-agnostic window, door and car detectors. We call these detector models **generic** models (as opposed to **specific** models), as the data used for training (**generic** data) does not follow any specific style. As shown in Table 1, the window, door and car samples originate from various sources, e.g. from a dataset of Belgian facades, or from a general-purpose car datasets, such as Leibe et al. (2007).

5.1 Object Detectors

Selecting the appropriate detector is not a simple task, as object detection is still an area of very active research. In recent years, many top-performing object detection systems have been based on the well-known deformable parts models (DPM) from Felzenszwalb et al. (2010a). These detectors show excellent detection quality as demonstrated, for example, on the yearly Pascal VOC (Everingham et al. 2010) challenge. Using multiple components and parts gives these detectors an advantage when detecting object classes characterized by a considerable amount of variation in their spatial extent. Conversely, when the object class is characterized to contain roughly rigid elements, classifiers based on a single template seem to be more appropriate. Lately, approaches based on the integral channels classifier proposed by Dollar et al. (2009) have demonstrated excellent quality (Benenson

Table 1 Overview of the data used to train the generic detectors

	Trained on		Evaluated on
	Positives	Negatives	
Windows	3924 from Belgian facade images	8343 from pascalVOC	eTRIMS/ECP
Doors	447 from Belgian facade images	8343 from pascalVOC	eTRIMS
Cars (frontal)	516 front- and rear-view car images	4268 from pascalVOC	eTRIMS
Cars (side)	344 from (Leibe et al. 2007)	4268 from pascalVOC	eTRIMS

et al. 2013) and detection speed (Benenson et al. 2012). The latter detector, dubbed `Very Fast` by the authors, not only reaches 100 Hz on the task of pedestrian detection, but also generalizes well to other classes. For example, in the German traffic sign detection challenge (Houben et al. 2013), one of the winning approaches (Mathias et al. 2013) was based on this detector.

We decided to compare the `Very Fast` and the DPM detector for the window detection task, using the following setup. For both detectors we train one model in each of the fivefolds of the Haussmann-specific window training data, as described in Sect. 3. For each fold we use all available positive training samples, while patches not overlapping with windows are used as negative examples. Additionally, we augment the negative set with 8383 images not containing windows from the Pascal VOC dataset. Speed comparisons were performed on an Intel Core i7 870 CPU + Nvidia GeForce GTX 590.

Deformable part-based model detector (DPM) The DPMs are trained using the latest release (version 5) (Girshick et al. 2012) with default settings. The number of components is set to 1. The training took roughly 5 h, and the testing speed of 3.8 s/image can be sped up by a factor of 10–15 by using a cascade (Felzenszwalb et al. 2010b). We noticed that training this window detector with two or more components only reduces the overall quality while increasing the training time.

`Very Fast` *detector* We use the publicly available open source implementation of the `Very Fast` detector (Benenson et al. 2012). The training is initialized by using a feature pool size of 30,000 random features. We perform 4 rounds of training (2000 stage classifiers), where each round is followed by bootstrapping 5000 hard negative samples. With this setup, training lasts around 8 h. The testing time of 2.1 s/image can be sped up by a factor of around 40 by approximating nearby scales and using a soft cascade, as described in the original paper.

The performance of all detectors is evaluated using the Pascal VOC overlap criterion of 50 % overlap over union. Figure 5 compares the mean detection rates for the task of specific window detection on ECP, i.e. detectors trained with Haussmann windows. For each of the fivefolds of the ECP dataset we trained a DPM detector and a `Very Fast` detector. All detectors are evaluated on their appropriate testing sets and the results are then averaged over the 5 cross-validation folds.

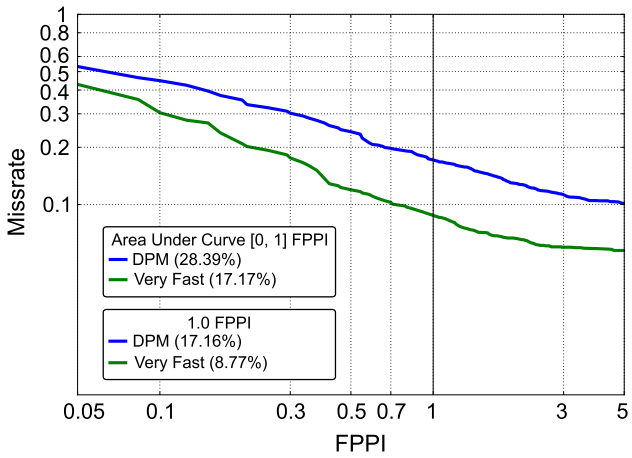

Fig. 5 Comparison of the dataset-specific DPM and `Very Fast` on the task of window detection. The *plot* shows the mean false positive per image (FPPI) versus miss-rate, averaged over fivefolds

The results reveal that the single template-based `Very Fast` detector performs better than DPMs on this task. This behavior may be explained with the fact that the window and door classes do not consist of independently moving parts. Furthermore, image rectification leads to axis-aligned window corners. Due to the better detection quality and speed we opted for the `Very Fast` detector in all following detection experiments. Figure 7 shows some example window detections of the `Very Fast` detector. For the task of car detection, DPMs might have a better performance due to the higher shape variability of the car class, but our experiments indicate that the `Very Fast` detector performs adequately on the few car samples in the eTRIMS dataset, where cars are usually shown either from the side, front or rear.

5.2 Generic and Specific Object Detectors

The ECP dataset contains 3096 windows and 109 doors, exhibiting the style of typical Haussmannian facades. Hence, all windows and doors have similar appearances and are therefore well suited to train Haussmann-specific window and door detectors. On the other hand, eTRIMS provides

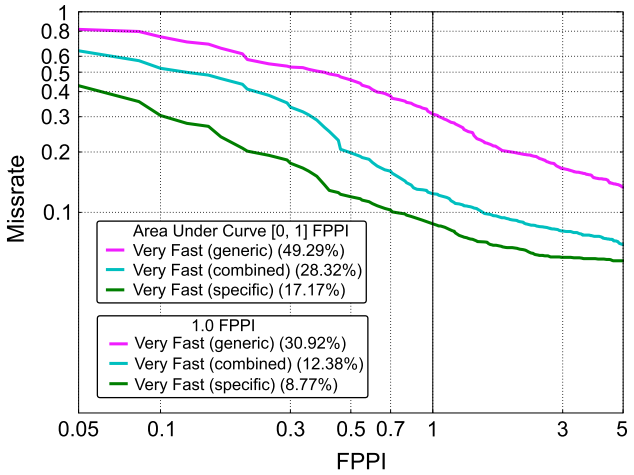

Fig. 6 Comparison of specific, generic and combined window detector on the ECP dataset. The combined detector is trained on the joint training set of style specific and generic window samples

Fig. 7 Example detections of the `Very Fast` specific window detector. The *color* encodes the confidence of the detection from high confidence (*red*) to low confidence (*black*) (Color figure online)

only 1016 window and 85 door instances from many different architectural styles, which leads to a high variance of e.g. appearance and aspect ratio. A common strategy to handle such diverse object classes consists in clustering the data into subsets (e.g. by their aspect ratio Felzenszwalb et al. 2010a) and independently training one detector for each subset. This would further reduce the number of samples used for training. We therefore did not train dataset-specific detectors on eTRIMS. Instead, we use the eTRIMS dataset as a proof of concept which shows that, even when using generic detectors trained with data coming from different sources, we can improve the labeling quality in our middle layer.

To recapitulate, we use style-specific detectors if there are enough samples in the training data. Otherwise, we train generic detectors. Still, we can gain some insight by evaluating the performance difference between detectors trained on style-specific and generic input data.

In Fig. 6 we compare the generic and specific window detectors on Haussmann. At 1 false positive per image (FPPI), the generic detector discovers around 70 % of the windows, while the specific detector finds more than 90 %.

Even though the generic detector could be used to improve window labeling in the middle layer, the specific detector has much better detection rates with fewer false positives. On the other hand, the advantage of using a generic detector lies in reduced training times when the system should be applied to many different styles. Instead of always retraining style-specific detectors—and having to know which style is relevant for any individual building—one might opt for collecting a large set of generic detectors for different facade elements and just select which detectors to use for specific styles. The detector obtained by **combining** specific and generic training data outperforms the generic detector, however it does not match the quality of the specific detector. By adding specific Haussmann windows to the training data, we get a better representation of Haussmann windows and therefore improve over the generic detector model. On the other hand, by adding generic window samples to the Haussmann samples, we add a much higher variability to an otherwise quite homogeneous training data. We believe that this variability cannot be exploited, as the more general window detector introduces new false positives rather than detecting additional windows that were missed before. In conclusion, style-specific detectors perform better than generic detectors, especially when the data variation is limited (e.g. for Haussmann windows). Generic detectors can still be used when it is infeasible to re-train detectors for every new style or when insufficient style-specific training data is available.

5.3 Learning Detector Label Distributions

In order to merge the information coming from the object detectors with a semantic labeling of an image, we need to transform the detector output (typically a set of bounding boxes with scores) into per-pixel label probabilities. The simplest approach would be to simply set the probability of each pixel within a detection to 1 for the class corre-

sponding to the detector (e.g. window), and zero to all other classes. However, window detections often cover other parts of the facade, such as a balcony or wall. The classification accuracy of balconies and walls would thus be negatively affected. To illustrate this, Fig. 7 shows an example output of the window detector, where the score of each detector is color-coded (brighter means higher detector confidence). Furthermore, not all detections ought to have the same influence: we want to significantly boost window probabilities in bounding boxes of high-scoring detections, but to be more conservative with low-scoring detections, since they might be false positives.

Let $\Delta = \{\delta_k\}_{1\leq k\leq K}$ be the set of K detectors. In the ECP dataset we use only a window and a door detector, so $K = 2$. We propose a novel way of learning the detector label distributions $\mathbf{P}^{\delta_k}(l|x_i)$, i.e. the probabilities that a given point x_i in a test image belongs to one of the semantic labels $l \in \Psi$ according to the detector δ_k. To achieve this, we investigate how detections of a certain score spatially overlap with the ground truth labeled images in the validation set.

Let us denote the set of N images in the validation set with $X^v = \{\mathbf{x}_n\}_{1\leq n\leq N}$, and their corresponding ground truth labeled images with $Y^v = \{\mathbf{y}_n\}_{1\leq n\leq N}$.

After running a detector δ_k on the validation set, we obtain a set of M_k detections

$$D_k^v = \{d_j \mid d_j = (\mathbf{b}_j, r_j, \mathbf{y}_j)\}_{1\leq j\leq M_k} \tag{1}$$

where each detection d_j is characterized by its bounding box $\mathbf{b}_j$, score (detector confidence) r_j and the labeled ground truth corresponding to the image where the detection was found: $\mathbf{y}_j \in Y$. The detections in the set D_k^v are sorted by their score in descending order.

Then, for each detection $d_j \in D_k^v$, we create a sub-image $\mathbf{S}_j$ by extracting the area of the corresponding ground truth label $\mathbf{y}_j$ covered by the bounding box $\mathbf{b}_j$, denoted as

$$\mathbf{S}_j = \mathbf{y}_j[\mathbf{b}_j] \tag{2}$$

The extracted sub-images are all subsequently rescaled to the same size using nearest-neighbor interpolation. For the normalized width u^{norm} and height v^{norm} we chose the value of 100 pixels, since most detection sizes in our dataset were on the same order of magnitude. The normalized sub-images are denoted as

$$\mathbf{S}_j^{norm} = NNResize(\mathbf{S}_j, v^{norm}, u^{norm}) \tag{3}$$

By construction, the normalized sub-images contain a subset of labels (classes) Ψ. Next, we create $|\Psi|$ binary label masks for each sub-image, defined as

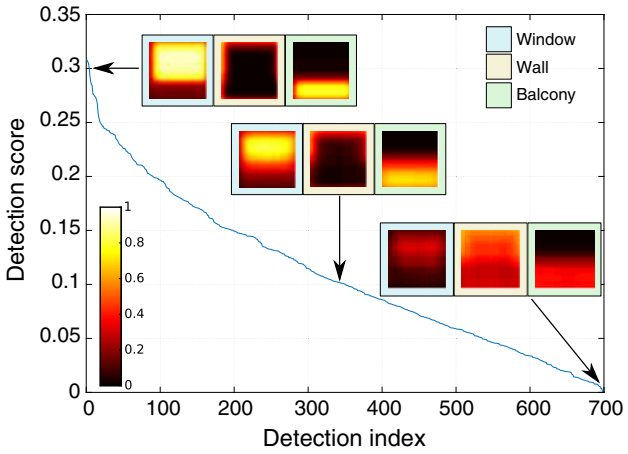

Fig. 8 (Best viewed in *color*) Learning label distributions for the window detector. Window detections on the ECP validation set are sorted by their score in descending order. High-scoring detections (*top-left*) provide a much stronger prior than the low-scoring detections (*bottom-right*) (Color figure online)

$$\mathbf{B}_j^l = \mathbf{1}_l(\mathbf{S}_j^{norm}), \quad \forall l \in \Psi \tag{4}$$

where $\mathbf{1}_l$ is the indicator function selecting only pixels with the label l. To obtain a smooth label distribution, we average the binary label masks of detections with a similar score. For each detection, we consider γ neighboring detections in the original sorted list of detections. Let $j_0 = max(1, j - \frac{\gamma}{2})$. We define

$$\mathbf{Q}_j^l = \frac{1}{\gamma}\sum_{i=j_0}^{j_0+\gamma} \mathbf{B}_i^l, \quad \forall l \in \Psi \tag{5}$$

as the per-pixel probability that a given pixel in the bounding box $\mathbf{b}_j$ is labeled with l. The obtained $\mathbf{Q}_j$ is a valid probability distribution, since

$$\sum_{l\in\Psi} \mathbf{Q}_j^l = \mathbf{J}_{v,u} \tag{6}$$

where $\mathbf{J}_{v,u}$ is a matrix of ones. In our experiments we set $\gamma = 200$, as there are on average 700 detections in the validation set. Very small values of γ result in distributions that are no longer smooth, while by using higher values of γ the detection score starts to lose its effect, as all $\mathbf{Q}_j$ become rather similar.

Examples of the resulting $\mathbf{Q}_j$ are visualized in Fig. 8 for $l \in \{window, wall, balcony\}$ (other labels are not shown for clarity). For high-scoring detections (top-left corner of the image), our approach learns that the upper part of a detection should be assigned to the *window* label, while the lower part often corresponds to the *balcony* label. On the other hand, for lower-scoring detections, the effect of false positives firing on *wall* areas is so prominent that the *wall* label probability

actually surpasses the *window* label. As we will see, the effect of false positives on the final labeling will be kept at bay, since our system hesitates to assign high label probabilities to low-scoring detections.

We can consider these learned label distributions as a look-up table during the testing phase. In a test image $\mathbf{x}^{test}$, we want to define the label distribution for each pixel, given the detections of a single detector δ_k. Initially, we assume no prior knowledge and assign a uniform label distribution to every pixel. Let

$$\mathbf{P}^{\delta_k}(l|x_i) = \frac{1}{|\Psi|}, \quad \forall x_i \in \mathbf{x}^{test}, l \in \Psi \tag{7}$$

be the initial probability distribution of labels in the image, where l is the predicted label for pixel x_i. After running the detector δ_k on the evaluation set, we obtain a set of M_k^e detections

$$D_k^e = \{d_j \mid d_j = (\mathbf{b}_j, r_j, \mathbf{y}_j)\}_{1 \le j \le M_k^e} \tag{8}$$

For every detection d_j in set D_k^e, we find the detection $d_{NN(j)}$ from the set D_k^v with the closest score to r_j. Its corresponding learned label distribution $\mathbf{Q}_{NN(j)}$ is resized to fit the bounding box $\mathbf{b}_j$, denoted as

$$\mathbf{Q}_{NN(j)}^{resized} = NNResize(\mathbf{Q}_{NN(j)}, v_j, u_j) \tag{9}$$

where u_j and v_j denote the width and height of the detection bounding box $\mathbf{b}_j$, respectively. Finally, the pixel probability distributions inside the bounding box are overwritten with the learned distribution, written as

$$\mathbf{P}^{\delta_k}(l|x_i) = Q_{NN(j)}^{resized}(x_i), \quad \forall x_i \in \mathbf{b}_j, l \in \Psi \tag{10}$$

The process is repeated for each detection d_j in the set D_k^e. Note that each position within $\mathbf{P}^{\delta_k}$ can be overwritten maximally once. There are no overlapping detections within one detector output due to non-maxima suppression. Detections of different object classes are handled by repeating the process for each detector δ_k, resulting in several learned priors $\mathbf{P}^{\delta_k}$.

5.4 Learning Facade Label Maps

In the previous section, we learned the label distributions only for pixels covered by detection bounding boxes. For other pixels, we assumed a uniform label distribution. However, it is logical to assume that the probability of a certain label also depends on the relative position of the pixel in the image. For example, one would expect *sky* pixels to appear mostly in the upper parts of the image, while the *shop* or *road* classes normally appear near the bottom of the image.

We can learn such a spatial prior in the form of *facade label maps* by analyzing the ground truth labels in the training set. First, we resize each ground truth image $\mathbf{y}_n$ from the training set Y^t to a common size ($u^{normf} = v^{normf} = 500$). The normalized ground truth image is defined as

$$\mathbf{y}_n^{norm} = NNResize(\mathbf{y}_n, v^{normf}, u^{normf}) \tag{11}$$

Similar to the previous section, we create $|\Psi|$ binary label masks defined as

$$\mathbf{C}_n^l = \mathbf{1}_l(\mathbf{y}_n^{norm}), \quad \forall l \in \Psi \tag{12}$$

which are then averaged over the training set, obtaining $|\Psi|$ facade label maps

$$\mathbf{R}^l = \frac{1}{N^t} \sum_{n=1}^{N^t} \mathbf{C}_n^l, \quad \forall l \in \Psi \tag{13}$$

where N^t is the number of images in the training set Y^t. Figure 9 shows the learned label maps $\mathbf{R}^l$ for the ECP and eTRIMS datasets. The final distribution of labels in an evaluation image $\mathbf{x}^e$ with dimensions v^e and u^e is given by

$$\mathbf{P}^{\lambda}(l|x_i) = NNResize(\mathbf{R}_l, v^e, u^e), \quad \forall x_i \in \mathbf{x}^e, l \in \Psi \tag{14}$$

5.5 Incorporating Detector Knowledge into CRFs

In order to merge the labels coming from the bottom layer with those introduced by the detectors from the middle layer, we place a 2D Conditional Random Field (CRF) over the image pixels. We seek to minimize the CRF energy, defined as the weighted sum of unary potentials for each node and all pairwise potentials between neighboring nodes:

$$E(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \sum_{x_i} \Phi_s\,(y_i \mid x_i, \mathbf{w}) \tag{15}$$

$$+ \sum_{x_i} \sum_{x_j \sim x_i} \Phi_p\left(y_i, y_j \mid x_i, x_j, \mathbf{w}\right) \tag{16}$$

where x_i is an image pixel, $y_i \in \Psi$ represents the variable encoding the predicted label, $\mathbf{w} = \{w^{seg}, \mathbf{w}^{det}, \mathbf{w}^{lab}, w^{pair}\}$ is the set of CRF parameters, and the relation $\sim$ represents the 4-pixel neighborhood. We use the standard Potts model as the pairwise term, which encourages neighboring pixels to take on the same label. This has the effect of smoothing the output, with the degree of smoothing dependent on a parameter w^{pair}. The pairwise term is defined as

$$\Phi^p\left(y_i, y_j \mid x_i, x_j, \mathbf{w}\right) = \begin{cases} 0, & \text{if } y_i = y_j \\ w^{pair}, & \text{otherwise.} \end{cases} \tag{17}$$

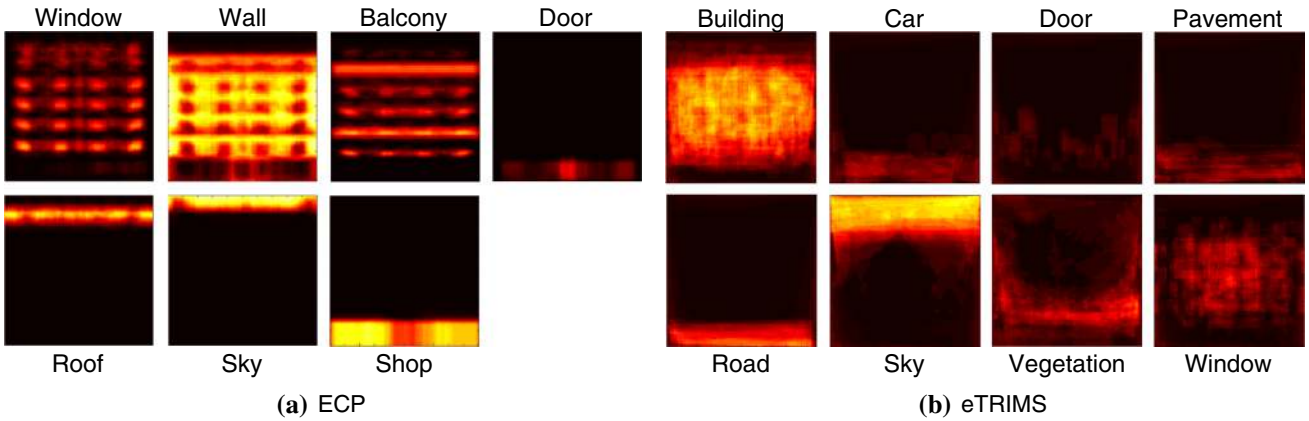

Fig. 9 Learned label maps from the training set in one cross-validation fold, by averaging over the different facades. *Brighter colors* denote higher label probability. Note the high level of regularity and alignment in the ECP dataset compared to the more washed-out probabilities for eTRIMS (Color figure online)

The unary term is a linear combination of the low-level information from the segment classification, the learned prior facade label distributions, and the detector outputs:

$$\begin{aligned}\Phi^s(y_i \mid x_i, \mathbf{w}) = &-w^{seg} \log P^{\sigma}(y_i \mid x_i) \\ &-\sum_{k=1}^{K} \mathbf{w}_k^{det} \log P^{\delta_k}(y_i \mid x_i) \\ &-\mathbf{w}_{y_i}^{lab} \log P^{\lambda}(y_i \mid x_i) \end{aligned} \tag{18}$$

Here, P^{σ} is the per-pixel probabilistic output of the bottom layer. Since the classification in the bottom layer operates at the level of segments, all pixels within the same segment share the same probability. The detector potentials P^{δ_k} and prior facade label map potentials P^{λ} were defined in Sects. 5.3 and 5.4, respectively. Parameters w^{seg},$\mathbf{w}^{det}$ and $\mathbf{w}^{lab}$ weigh the relative importance of segment classification, detector label maps, and facade label map priors. Note that $|\mathbf{w}^{lab}| = |\Psi|$, as we weigh each facade label map separately.

Applying the CRF model requires us to find the optimal labeling of a test image, given the set of parameters $\mathbf{w}$. The approximate solution to this problem can be found efficiently using graphcut-based methods (Boykov et al. 2001), since our CRF model contains only unary and submodular pairwise terms. Additionally, due to the usage of the Potts model, the α-expansion minimization guarantees a solution that is within a factor of two of the global minimum (Boykov et al. 2001). In the next section we describe how the parameter vector $\mathbf{w}$ can be learned from the validation set.

5.6 Learning CRF Parameters

There already exists a body of work on learning parameters in random field models. Most of these approaches use either a form of cross-validation or piecewise training. A good overview of parameter learning in CRFs can be found, for example, in Kumar et al. (2005) and Nowozin et al. (2010). We decided to follow the approach of Szummer et al. (2008), which is an efficient technique of max-margin learning in grid graphs, e.g. images, based on the structured support vector machine Tsochantaridis et al. (2005). This method represents parameter estimation as a maximum margin learning problem, formulated as

$$\begin{aligned} &\max_{\mathbf{w}:\|\mathbf{w}\|=1} \quad \gamma \quad s.t. \\ &E(\mathbf{y}, \mathbf{x}_n; \mathbf{w}) - E(\mathbf{y}_n, \mathbf{x}_n; \mathbf{w}) \geq \gamma \quad \forall \mathbf{y} \neq \mathbf{y}_n \quad \forall n \end{aligned} \tag{19}$$

where $\mathbf{x}_n$ is an image, $\mathbf{y}_n$ its corresponding ground truth, and n indexes all instances in the training set.

The learning algorithm constrains the energy of the ground truth labeling $\mathbf{y}_n$ to always be smaller than any other possible labeling $\mathbf{y}$ by a margin γ. Since there is an exponential number of possible image labelings, it is not feasible to solve the problem formulated in Eq. 19. The solution proposed in Szummer et al. (2008) works with a much smaller subset of labelings $\{\mathbf{S}_n\}$, i.e. a constrained set. For each image, a lowest-energy labeling is found using an efficient method, such as graph cuts. If this energy does not satisfy the margin, the labeling is added to the subset $\mathbf{S}^{(n)}$. After all images are processed, the parameters $\mathbf{w}$ are updated to satisfy the newly added constraints, and the process is repeated. Since there is only a finite number of labelings that can be added, the procedure is guaranteed to converge.

The above formulation is further improved by enforcing a larger margin when the labeling is far from the truth. This difference between desired and candidate labeling can be expressed in terms of a *loss function* $\Delta(\mathbf{y}_n, \mathbf{y})$. By adding slack variables ξ_n to account for constraint violations and rescaling the margin as proposed in Taskar et al. (2005), the following quadratic optimization problem is obtained:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{n=1}^{N}\xi_n \; s.t. \; \forall \mathbf{y} \in \mathbf{S}_n \quad \forall n \tag{20}$$

$$E(\mathbf{y}, \mathbf{x}_n; \mathbf{w}) - E(\mathbf{y}_n, \mathbf{x}_n; \mathbf{w}) \geq \Delta(\mathbf{y}_n, \mathbf{y}) - \xi_n \tag{21}$$

$$\xi_n \geq 0$$

where C is the regularization parameter and N is the number of training images. A common approach is to use Hamming loss, i.e. the number of mislabeled pixels in an image, as the loss function. However, our datasets do not have a balanced distribution of classes, since some classes only constitute a small percentage of total pixels (e.g. the *door* class). Mislabeling the small classes does not significantly change the overall pixel accuracy, however the class-wise accuracy is severely reduced. Therefore, we modify the loss function to take into account the frequencies of classes in each ground truth image, producing a greater loss when a low-frequency label is misclassified, resulting in a weighted Hamming loss:

$$\Delta(\mathbf{y}_n, \mathbf{y}) = \sum_{i=1}^{|\mathbf{y}|} f^{-1}(y_i^n)[y_i^n \neq y_i] \tag{22}$$

where [.] is the indicator function, and $f^{-1}(y_i^n)$ represents the inverse frequency of the label y_i^n in the ground-truth image $\mathbf{y}_n$.

To calculate the most violated constraint in Eq. 21 we must find the labeling y which minimizes the refined energy function E', defined as

$$E'(\mathbf{y}, \mathbf{x}_n; \mathbf{w}) = E(\mathbf{y}, \mathbf{x}_n; \mathbf{w}) - \Delta(\mathbf{y}_n, \mathbf{y}) \tag{23}$$

As shown in Szummer et al. (2008), the loss function can be 'absorbed' into the energy function if it decomposes the same way as the energy. Since the weighted Hamming loss decomposes over image pixels (nodes in the CRF), we can transform it into an additional unary potential. This corresponds to augmenting the unary potentials in the CRF:

$$\Phi_s^{'}(y_i \mid x_i, \mathbf{w}) = \Phi_s(y_i \mid x_i, \mathbf{w}) - f^{-1}(y_i^n)[y_i^n \neq y_i] \tag{24}$$

where Φ_s is defined in Eq. 18. The resulting problem of minimizing E' can still be solved efficiently using α-expansion, as the energy remains submodular. Every labeling that violates the margin constraint in Eq. 21 is added to the constraint set $\mathbf{S}_n$, and the parameter vector $\mathbf{w}$ is updated by minimizing Eq. 20. The process is repeated until $\mathbf{w}$ remains unchanged. We have implemented this approach using the SVM^{struct} software from Tsochantaridis et al. (2005).

6 Top Layer: Using Weak Architectural Principles

The previous two layers propose a generic approach to semantic labeling, which is initially based on super-pixel classification and subsequently enriched by object detectors. Although the results of the first two layers are quantitatively convincing, the effect of the initial segmentation is still present in the output. This manifests itself in the jagged boundaries of some elements as well as the missing or misplaced facade elements. Hence it is difficult to use the output of these layers to derive convincing facade models with clearly defined boundaries and structures. Therefore, in the top layer we add meta-knowledge about buildings without defining a full facade grammar, in contrast to Teboul et al. (2013). This meta-knowledge is expressed through the concept of *weak architectural principles*.

An important advantage of these guidelines over procedural grammar rules is that the former are directly observable in the images, whereas the latter keep some concepts implicit. Even if the combined application of a number of grammar rules may lead to, for example, vertical alignment of windows, there might be no single rule explicitly prescribing such alignment. An issue with style grammars can therefore be the indirect coupling between what they specify and what can be easily verified in the images. Our approach also enables the modeling of irregular facades, as we use architectural concepts as guidelines, not as hard constraints. Some of the proposed principles are quite generic and can be re-used for many different facade styles, while others were intentionally designed with a certain style in mind, e.g. the Haussmannian style. Similar to object detectors, most principles are formulated for the objects in the facades (window, balcony, door), as these elements have a clearly defined boundary. In the end, the interplay between data evidence and various principles will influence the placement, modification or removal of facade elements.

6.1 Overview

Our first task is to define how the idea of weak architectural principles can be integrated into a generic system which allows for easy modification and addition of these principles. Therefore, we employ a modular design, where each principle has a well-defined interface and may be individually activated depending on the dataset at hand.

Figure 10 shows the overview of our proposed system. The first step is to generate proposals of facade elements (bounding boxes with corresponding labels) from the output of the middle layer (Sect. 6.1.1). Let us define a facade configuration $\mathbb{F}$ as a set of facade elements which constitute a valid facade (i.e. no overlapping windows). The most probable interpretation of the facade from the previous layer is selected as the initial facade configuration $\mathbb{F}_0$, while non-

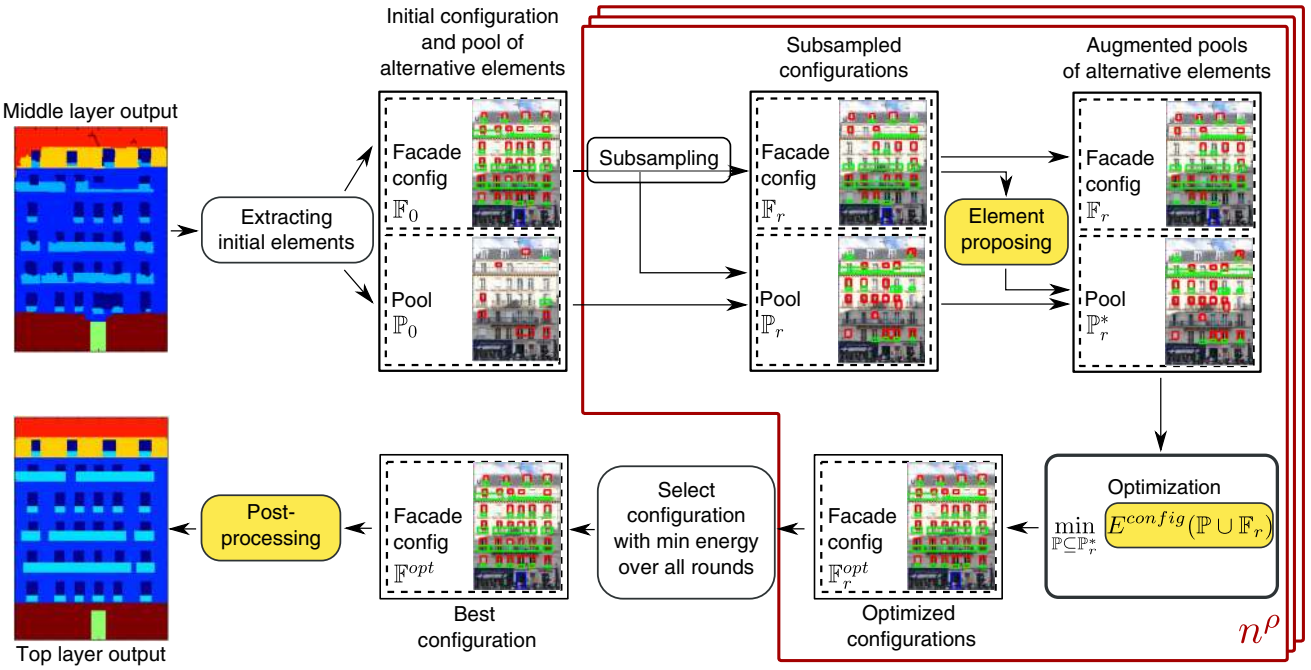

Fig. 10 A high-level overview of the top layer. The *blocks highlighted in yellow* depend on weak architectural principles, see text for description (Color figure online)

selected elements (such as overlapping windows) are placed in a set of alternative facade elements, or a *pool* $\mathbb{P}_0$.

The initial configuration is not necessarily the correct one, as it might contain false positives. To remove them, we perform random subsampling, retaining a subset of elements in the configuration, and moving the rest to the pool of alternative elements (Sect. 6.1.2). The subsampling is repeated in n^ρ rounds to increase the likelihood that, in at least one round r, the subsampled configuration $\mathbb{F}_r$ contains only true positives. Based on the subsampled configuration $\mathbb{F}_r$, the pool $\mathbb{P}_r$ is extended by new facade elements (Sect. 6.1.3). An optimization method is proposed to select the subset of elements in the augmented pool $\mathbb{P}_r^*$ which best complements the subsampled configuration $\mathbb{F}_r$ (Sect. 6.1.4), given an energy function E^{config}. The best facade configuration $\mathbb{F}^{opt}$ over all n^ρ is then fed into a post-processing step (Sect. 6.1.5).

The weak architectural principles are used for three different purposes in our system, see Table 2. First, they can *propose* new elements (Sect. 6.2.1). Second, some principles *grade* the proposal configurations through E^{config} (Sect. 6.2.2). Finally, certain principles are used to *modify* the facade element in the post-processing step (Sect. 6.2.3).

6.1.1 Extracting Initial Facade Elements

Starting from the pixel-wise classification output of the middle layer, the first step is to generate the initial configuration of facade elements $\mathbb{F}_0$. This configuration should contain facade elements such as windows, doors, or balconies. More precisely, each element is determined with the bounding box and its corresponding label. However, our middle layer produces a labeled image $\mathbf{y}^{L2}$, as opposed to discrete elements. To generate facade element proposals, we start by using connected components in the label map $\mathbf{y}^{L2}$ and define a minimal bounding rectangle R_z around the z-th connected component.

This minimal bounding rectangle is often too large compared to the actual facade element. Some initial super-pixels float over the object's real boundaries, which leads to oversized minimal bounding rectangles. To mitigate this problem, we adjust the edges of each rectangle R_z by maximizing the coincidence with the edges of the connected component. Each edge of the current rectangle is adjusted by shifting it pixel by pixel towards the center of the rectangle. Let D_z denote the number of pixels inside of R_z belonging to the connected component. We limit the search range with the constraint that the number of connected component pixels inside the new rectangle must not fall below τ^{init} percent of D_z. The threshold τ^{init} was set to 0.6 in our experiments. At each position of the element edge, we calculate the overlap between the edge and boundary pixels of the connected component, divided by the edge length.

We find at most two possible edge proposals per rectangle side. The first one results from the highest edge overlap with the connected component boundaries. The second proposal is added only if the ratio between its overlap and the highest edge overlap is above τ^{edge}, set to 0.75 in our experiments.

The rectangle with the best combination of edge proposals is added to $\mathbb{F}_0$, and all other combinations are added to the pool of alternative elements $\mathbb{P}_0$.

6.1.2 Sub-sampling

The initial facade configuration $\mathbb{F}_0$ obtained by the approach described in the previous section can potentially suffer from errors such as missing or misplaced elements, and false positives. As our proposed architectural principles are not designed to remove elements, dealing with false positives require separate consideration.

Our approach to dealing with incorrect facade elements is to repeatedly sub-sample the starting facade configuration with the goal of achieving at least one configuration containing only correct elements. Furthermore, we do not discard the elements that are not sampled, rather, we move them to the pool of alternative elements for later consideration.

The sub-sampling is repeated in n^ρ rounds. In each round we randomly split $\mathbb{F}_0$ into two disjoint subsets: the elements from the first subset are kept as the facade configuration of the r-th round $\mathbb{F}_r$, while the other elements are added to the pool $\mathbb{P}_0$, constructing the pool $\mathbb{P}_r$. The split is performed element-wise by adding an element to $\mathbb{P}_r$ with probability p^{rem}, or to $\mathbb{F}_r$ with probability $1 - p^{rem}$. We set $p^{rem} = 0.4$, which allows us to keep on average more than half of the initial elements while at the same allowing to remove different combinations of potentially incorrect candidates. In our experiments, the setting of $n^\rho = 20$ produced satisfactory results. Further increase in the number of rounds typically does not result in finding a better configuration. When reducing the number of rounds, the performance degrades gracefully, converging to the initial labeling defined by $\mathbb{F}_0$.

6.1.3 Element Proposing

Assuming that the configuration $\mathbb{F}_r$ contains only true positives (which should hold true for at least one round r), we have a strong cue for discovering facade elements which are not present in either $\mathbb{F}_r$ or $\mathbb{P}_r$. For example, we might search for elements similar to those in the current facade configuration.

At this point, we can plug in any weak architectural principle which has the property of proposing new facade elements. Depending on the configuration $\mathbb{F}_r$, different additional facade elements might be proposed in each round (see examples in Sect. 6.2.1). The facade elements proposed by these principles are then simply added to $\mathbb{P}_r$, resulting in the augmented pool $\mathbb{P}_r^*$.

6.1.4 Optimization

Starting from an incomplete facade configuration $\mathbb{F}_r$ and an augmented pool of elements $\mathbb{P}_r^*$ in the facade, our goal now is to find the optimal facade configuration $\mathbb{F}_r^{opt}$ with regard to a certain energy function. We assume that the elements in $\mathbb{F}_r$ are fixed, so the optimization amounts to the search for the optimal subset of elements in $\mathbb{P}_r^*$ which, combined with the entire set $\mathbb{F}_r$, minimizes the energy function:

$$\begin{aligned} \mathbb{F}_r^{opt} = & \mathbb{F}_r \cup \underset{\mathbb{P} \subseteq \mathbb{P}_r^*}{\operatorname{argmin}} \; E^{config}(\mathbb{P} \cup \mathbb{F}_r) \\ & s.t. \quad c_{\text{overlap}}(\mathbb{P}) \end{aligned} \tag{25}$$

The c_{overlap} constraints disallow any pair of overlapping elements in $\mathbb{P}$ to be selected at the same time, and can be expressed as a set of linear inequalities of the form $\mathbf{Ax} \leq \mathbf{1}$.

Selecting a subset of elements can be viewed as a binary integer optimization problem, where each variable indicates whether the corresponding element is included in the subset. In general, binary integer programming is NP-complete (Karp 1972). There are of course certain subsets of energy functions for which this optimization can be performed efficiently. For example, Kolmogorov and Zabih (2004) shows that if the energy function can be written as a sum of functions of up to two binary variables at a time (unary and regular pairwise potentials), the optimization can be performed in polynomial time. However, we allow the energy function to depend on an arbitrarily complex set of weak architectural principles, see Sect. 6.2.2. For this reason, and to keep the optimization as general as possible, we assume no prior structure of the energy function. This rules out the use of deterministic optimization approaches, such as cutting plane methods (where the objective function need not be convex), branch-and-bound (no knowledge on lower and upper bounds), or dynamic programming (no optimal substructure property).

Therefore, our choice is limited to (meta)heuristic methods. The simplest approaches, e.g. hill climbing or coordinate descent, are prone to getting stuck in local minima (Russell et al. 1996). Monte Carlo methods such as simulated annealing (Kirkpatrick et al. 1983) or MCMC (Gilks et al. 1995) are more powerful, but typically require a large number of objective function evaluations. Genetic algorithms (Holland 1975) are another popular metaheuristic, which was adapted for efficient solving of integer optimization problems by Deep et al. (2009). Although there is no proof of convergence, the latter implementation was shown to compare favorably to random search or annealing-based algorithms on certain datasets. We use an existing implementation of this approach in MATLAB's Global Optimization Toolbox (Mathworks 2014), with default parameters, to solve the minimization problem in Eq. 25.

Table 2 Weak architectural principles used to complement the segmentation results of the first two layers

Principle	Propose	Grade	Post-process	ECP	eTRIMS
(Non-)alignment: vertical and horizontal	–	✓	✓	✓	✓
Similarity of different *windows* of the same *facade*	✓	–	–	✓	✓
Facade symmetry	✓	–	–	✓	✓
Co-occurrence of elements	–	✓	–	✓	–
Door hypothesis: first floor, touching ground	✓	✓	–	✓	–
Vertical region order: { *shop* *, *facade* $^+$, *roof* *, *sky* * }	–	–	✓	✓	–

A tick in the "Propose" column denotes that the principle is used to propose new facade elements. Some principles can be used to evaluate the fitness of the facade configuration, denoted with a tick in the "Grade" column. The "Post-process" principles can be used in the last step of the inference procedure to modify the existing facade elements. Last two columns indicate which principles are used for each of the datasets

6.1.5 Post-processing

After n^o rounds of sampling and optimization, the facade configuration with the lowest energy F^{opt} is selected as the best one. Note that the bounding boxes of facade elements boxes are fixed during optimization. Therefore, we employ post-processing principles on the best configuration, to clean up the final result by adjusting facade element boundaries.

6.2 Weak Architectural Principles

The weak architectural principles introduce meta-knowledge about facades into the labeling process. All principles take a configuration of existing facade elements as input, and can be divided into three main categories based on their output. The first category contains principles which propose new facade elements. These are used for generating new objects, which have not yet been discovered in the first two layers of our pipeline. Second, some principles can be used to grade proposal facade configurations, producing a single number, the 'energy' of the configuration as output. For example, the alignment principle should produce low energy for configurations with well-aligned elements. Third, some principles are used as a simple post-processing step, modifying existing elements in the facade configuration. Table 2 shows an overview of our proposed principles, sorted into the three main categories. The last two columns denote whether the principle was used while analysing a certain dataset. In the following sections, we describe in detail the aforementioned categories of principles.

6.2.1 Element-Proposing Principles

Based on a given facade configuration $\mathbb{F}_r$, element-proposing principles suggest new facade elements by exploiting meta-knowledge about (style-specific) facade structure.

As shown in Table 2, we identify three different principles for the proposal of new facade elements, namely the *similarity*, *symmetry* and *door hypothesis*. Each of these principles proposes a separate set of facade elements, which we denote $\mathbb{W}^{sim}$, $\mathbb{W}^{sym}$, and $\mathbb{W}^{door}$, respectively. Other principles can be added if necessary. We denote with $\mathbb{W}$ the set of all facade element proposals generated by the principles, i.e. $\mathbb{W} = \{\mathbb{W}^{sim} \cup \mathbb{W}^{sym} \cup \mathbb{W}^{door}\}$. **The similarity principle** is based on the observation that most facades contain visually similar objects. If some elements are missing in the current facade configuration, they can still be found through visual similarity to the existing elements, see Fig. 11. This principle is applied separately per object class and is parameterized by the median width u^{med} and height v^{med} of the object category. Our implementation is similar to that of Mathias et al. (2011a). Every object in the facade votes for similar elements using an ISM-like voting scheme (Leibe et al. 2006). As features we use self similarity descriptors (Shechtman and Irani 2007) calculated at Harris corner points.

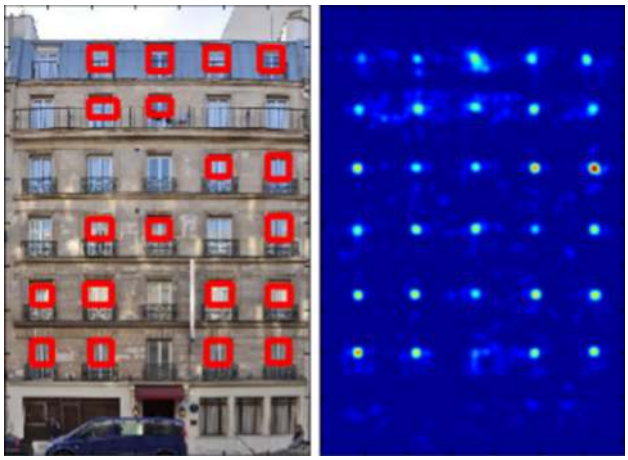

Fig. 11 Similarity principle: *Left* windows marked with *red rectangles* are the initially discovered windows. *Right* the similarity voting space contains strong peaks at previously undetected windows (Color figure online)

Let us consider a set of feature points that fall within the bounding box of a single element in the facade configuration. Each of these feature points is defined by its descriptor, a

vote vector to the center of the bounding box, and the size of the bounding box of the element. For each feature point, we search for 10 nearest neighbors among all feature points in the image based on its descriptor. The neighbors then cast votes into a global voting space using a Gaussian kernel of size $\min(u^{med}, v^{med})$. The process is repeated for all facade elements in the configuration. After all votes are collected, we perform greedy non-maximum suppression: each maximum defines an area of size $u^{med} \times v^{med}$ in which we keep the maximum and set the other values of the voting space in that area to 0. Most of the maxima in the voting space will be situated inside the bounding boxes of existing elements. Each remaining maximum defines a bounding box with the size defined as the median of bounding boxes sizes corresponding to votes which contribute to the maximum.

As the similarity voting is performed based on the subset $\mathbb{F}_r$ (Sect. 6.1.1), some maxima will correspond to facade elements already in $\mathbb{F}_r$. Only new elements build the set $\mathbb{W}^{sim}$. We limit the number of new proposals to $|\mathbb{F}_r|$, as we do not wish to add more proposals than the number of elements currently in the configuration.

Harris corners are also used as a simple measure in the principle of vertical **symmetry**. The interest points are mirrored about a symmetry axis (line) hypothesis. A match is defined by two interest point locations, which are mirrors of each other about the symmetry axis. Note that we only match interest point locations at this point, not their descriptors.

The maximum number of matches divided by the points under consideration defines a simple symmetry score for the corresponding symmetry axis. If symmetry is detected (symmetry score $> \tau^{sym}$), facade elements are mirrored about the similarity line with the maximum score and constitute the set of proposals $\mathbb{W}^{sym}$. For the value of τ^{sym} we select the lowest symmetry score from all symmetric facade examples in the training and validation sets. Figure 12 shows an example of a symmetric facade, with the symmetry axis denoted as a dashed blue line.

The **door hypothesis** principle creates a single door proposal $\mathbb{W}^{door}$, and it is only applied when $\mathbb{F}^r$ does not already contain a door bounding box. If there are no door objects in the pool of alternative elements either, we fall back to the probabilistic output of the bottom layer. We expect a door to be at least the size of a median window in the facade. Therefore, we first search for the maximum response by sliding a window of size $u^{med} \times v^{med}$ (median window size) over the bottom layer probabilistic output and averaging pixel probabilities corresponding to the *door* class inside that bounding box. From the position with the maximum support, we greedily grow the door bounding box until the average probability of the *door* class starts decreasing. Even if the real image contains several doors, this principle is limited to produce only one element, the one with higher support.

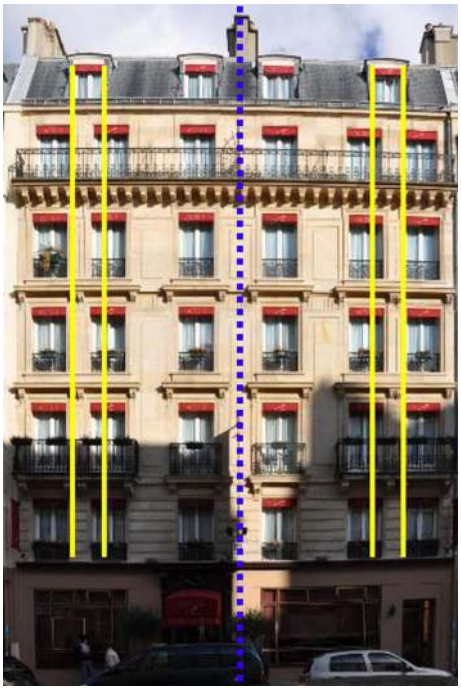

Fig. 12 The (non-)alignment principle states that facade elements should be either aligned or clearly off-center. In this image, windows exhibit a high degree of horizontal and vertical alignment. Two windows *bordered with yellow lines* are vertically off-center to other windows. This should not be penalized, as this is a often-observed window configuration. The *blue dashed line* depicts the symmetry axis of the facade (Color figure online)

6.2.2 Element-Grading Principles

Element-grading principles contribute to the energy function E^{config} which is used to judge a proposal facade configuration $\mathbb{F}$. We define this energy function as

$$E^{config}(\mathbb{F}; \mathbf{y}^{L2}) = E^{data}(\mathbb{F}; \mathbf{y}^{L2}) + \sum_{\pi \in weakPrinciples} \alpha^{\pi} E^{\pi}(\mathbb{F}; \mathbf{y}^{L2}) \quad (26)$$

where $\mathbf{y}^{L2}$ represents the middle layer output (CRF labeling). The data term E^{data} encourages the configuration to be as similar as possible to the prediction from the middle layer. It is independent from any principle and defined by:

$$E^{data}(\mathbb{F}; \mathbf{y}^{L2}) = \sum_{l \in \Psi_{obj}} E_l^{data}(\mathbb{F}; \mathbf{y}^{L2}) \quad (27)$$

$$E_l^{data}(\mathbb{F}; \mathbf{y}^{L2}) = -\sum_{y_i \in \mathbf{y}^{L2}} [y_i = l \wedge g(y_i, \mathbb{F}, l) = 1] / \sum_{y_i \in \mathbf{y}^{L2}} g(y_i, \mathbb{F}, l) \quad (28)$$

$$+ \sum_{y_i \in \mathbf{y}^{L2}} [y_i = l \wedge g(y_i, \mathbb{F}, l) \neq 1] / \sum_{y_i \in \mathbf{y}^{L2}} [y_i = l], \quad (29)$$

where $\Psi_{obj} \in \Psi$ denotes the subset of all object labels, as only facade objects are optimized in this step. Note that we define the data term separately for each object label $l \in \Psi_{obj}$. The function $g(y_i, \mathbb{F}, l)$ returns 1 when y_i is covered by a bounding box with the same label l from $\mathbb{F}$. Expression 28 reduces the energy when labeled pixel y_i is covered with a facade element with the same label from configuration $\mathbb{F}$, while expression 29 penalizes object pixels not covered by facade elements from $\mathbb{F}_r$.

The energy of each grading principle is weighted by α^{π} and added to the total energy. We determine the values for α^{π} on the validation set. In the following, we will describe the principles that contribute to the energy function.

The (non-)alignment principle is based on the observation that many facade elements of the same type are either exactly aligned or clearly off-center (see the yellow lines in Fig. 12). The energy for an object class is defined as

$$E^{align}(\mathbb{F}) = \sum_{(e_1,e_2)} \Big(\beta(s_1^{(e_1)} - s_1^{(e_2)}; \tau^w) + \beta(s_2^{(e_1)} - s_2^{(e_2)}; \tau^w) + \beta(t_1^{(e_1)} - t_1^{(e_2)}; \tau^h) + \beta(t_2^{(e_1)} - t_2^{(e_2)}; \tau^h) \Big) \tag{30}$$

$$\beta(z;\tau) = \begin{cases} \frac{\tau^2}{6}(1 - [1 - z/\tau]^2)^3, & \text{if } |z| \le \tau \\ \frac{\tau^2}{6}, & \text{if } |z| > \tau \end{cases} \tag{31}$$

where e_1 and e_2 refer to each possible combination of same-class elements in $\mathbb{F}$, and (s_1, t_1) and (s_2, t_2) represent the coordinates of the top-left and bottom-right corners of an element. The capped influence function β rates the top, bottom, left and right alignment of a pair of facade elements. The function has a constant value as soon as the distance between element boundaries exceeds a certain threshold τ. Based on our initial observation that windows are either aligned or completely misaligned, we set τ^w and τ^h to half of the median object width and height respectively.

The principle of **co-occurring elements** reflects the observation that pairs of elements appear in certain fixed configurations. One particular case of this principle is window and balcony co-occurrence: a facade should not have a balcony without a corresponding window. Therefore, we first try to assign at least one window to each balcony. Balconies without a corresponding window are then penalized by adding a constant value τ^{occ} to the energy term. By setting $\tau^{occ} > 0$ we increase the energy of solutions containing one or more solitary balconies. The co-occurence principle might as well be used for other pairs of elements or even as a facade element proposing principle, but we leave this for future work.

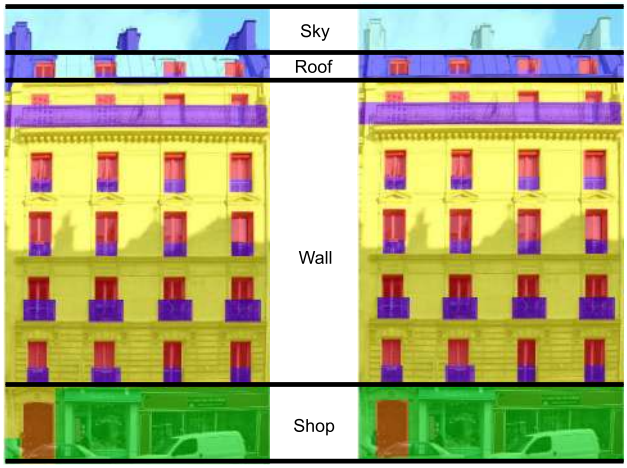

Fig. 13 The vertical region order principle determines the border between the sky, roof, wall and shop areas of the facade

6.2.3 Post-processing Principles

The **vertical region order principle** states the specific order of the *sky*, *roof*, *wall* and *shop* areas observed for Haussmannian facades. We enforce such an order in our output labeling (see Fig. 13). First, we find the initial split lines between the aforementioned areas. This is done by finding the connected components of the corresponding labels and placing a split line on the lower boundaries of the regions. Then, similar to Sect. 6.1.1, we test candidate split positions by moving the split lines pixel by pixel in the upward direction, seeking to maximize the overlap between the split line and region boundary pixels. After the splitting positions between the regions are found, we switch the labels of the aforementioned classes to be consistent with the region order.

The (non-)alignment principle is also used in the post-processing step. We use the second part of the energy function E^{align} (31) to align windows horizontally by adjusting the upper and lower borders of their bounding boxes. We find the local minimum of this energy with the iterative BFGS Quasi-Newton method Fletcher (1987). The result is that all windows aligned horizontally within a tolerance of τ^h will now be perfectly aligned with each other.

7 Results

We compare our approach to previous work on two datasets for facade parsing. Tables 3 and 5 show the performance of all approaches evaluated per class, as well as the average pixel and class accuracies. We show the results of our system for each layer of the pipeline together with the top layer performance of our previous work Martinović et al. (2012)

Table 3 Performance on the ECP dataset (in percent)

	Window	Wall	Balcony	Door	Roof	Sky	Shop	Pixel avg	Class avg
RF (Teboul et al. 2010)	33	67	32	82	52	92	20	53.46	53.73
RL (Teboul et al. 2013)	55	82	49	43	52	97	82	73.24	65.66
DKL (Dai et al. 2012)	72	87	70	66	80	93	91	83.50	79.80
SPT (Tyleček and Šára 2013)	75	86	73	66	85	95	95	84.20	82.14
3Layer (Martinović et al. 2012)	75	88	70	67	74	97	93	84.17	80.71
ATLAS (ours)									
Bottom layer	64	91	75	41	82	94	91	84.75	76.67
Middle layer	76	90	81	58	87	94	97	**88.07**	83.36
Top layer	78	89	87	71	79	96	95	88.02	**85.22**

Bold values refer to the best result
All experiments were performed with the same protocol (5-split cross-validation with 60 training, 20 validation and 20 testing images)

and the performance of other approaches. Example output of our system can be found in Figs. 14 and 15.

7.1 ECP Database

All methods were evaluated following the same fivefold cross validation, evaluated on the updated annotations as described in Martinović et al. (2012). In Table 3, we compare our results with the Random Forest (RF) pixel classifier of Teboul et al. (2010), the Reinforcement Learning (RL) grammar-based approach from Teboul et al. (2013), the domain knowledge learning (DKL) work of Dai et al. (2012), and the recent Spatial Pattern Templates (SPT) work by Tyleček and Šára (2013). We retrained and evaluated the RF and RL classifiers using the publicly available code, while DKL and SPT results were provided by the respective authors.

As expected, the simplest approach—Random Forest classifier based directly on image patches (Teboul et al. 2010)—exhibits the poorest performance. This can be partly attributed to weak features (raw pixel values), and partly to the lack of context, since every pixel is classified based only on its local patch of size 13×13. Compared to this approach, our bottom layer already achieves a better performance for all classes, due to the fact that we use a superpixel-based approach and more discriminant extracted features.

The state-of-the-art grammar-based RL approach (Teboul et al. 2013) requires a prior definition of a specific Haussmannian-style procedural grammar. The free parameters of the grammar are then optimized such that the agreement between the resulting labeling and the bottom-up merit function (RF labeling) is maximized. This approach greatly improves upon the results of the earlier RF approach, yet still performs worse than any layer output of our approach. One of the reasons for this behaviour is that the somewhat over-simplified grammar restricts the space of possible facade labelings, imposing certain structure even if it is not present in the image. For example, vertically misaligned roof windows are not supported with the existing grammar, and are thus mislabeled. Our approach does not suffer from these issues. One may argue that the lower performance of the RL method stems from their usage of less informative bottom-up cues. Therefore, we investigate how the RL approach performs when using much stronger merit functions, namely the output of our bottom and middle layers. The results in Table 4 (right) show that the RL method indeed benefits from stronger bottom-up information. However, when using our bottom and middle layer as the merit function, the RL method achieves lower performance than the merit function itself. This supports our claim that adding strong grammar constraints can actually decrease the overall performance.

Comparable results to our bottom layer were achieved by Dai et al. (2012), an approach which, like ours, forgoes the usage of style-specific grammars. Instead, it is designed to adapt to various building styles by learning weights for different architectural principles. However, as the approach was tested on only one building style (ECP dataset), it is difficult to assess the effectiveness of the learning algorithm. Furthermore, their initial image segmentation into rectangular regions is fixed and might pose a problem when dealing with more general facades containing irregular appearance (e.g. eTrims dataset). Our top layer utilizes a more flexible set of principles, which are not restricted to follow the initial segmentation. For example, we allow classes such as *car* or *vegetation* to keep their irregular boundaries.

Another region-based method, SPT (Tyleček and Šára 2013) achieves comparable results to our bottom layer, even outperforming it in class accuracy. This demonstrates that adding a region-based CRF with higher-order potentials on top of the initial segment classification boosts performance. We expect that our approach would benefit by integrating the SPT method in the first layer, which we leave for future work.

Fig. 14 Results on the eTRIMS dataset. (*Left*) The original image. (*Middle-left/center/right*) Outputs from the bottom, middle and top layers, respectively. (*Right*) Ground truth

When considering the added value of each layer in our approach, it is clear that the middle layer produces the biggest improvement in pixel accuracy for the *window* and *door* classes, as was expected for the usage of object detectors. Additionally, the accuracy of other classes goes up due to the usage of learned label maps (Sect. 5.4) and the smoothing property of the CRF (Sect. 5.5). By introducing high-level knowledge through the top layer, we further improve on most of the classes. The noticeable drop in the *roof* and *sky* class can be explained by the fact that the "region order" principle (Sect. 6.2.3) imposes a straight line to separate regions which does not always match the real, more complex, boundary. We nevertheless kept these strict horizontal split lines between image regions, as they facilitate the process of procedural facade modeling.

Compared to the top-layer output of our previous work (Martinović et al. 2012), we improve on almost all classes, boosting the average pixel accuracy to 88 %. The increase of nearly 4 % was achieved through several refinements of this work, namely: using SVM as the region classifier, using

Fig. 15 Results on the ECP dataset. (*Left*) The original image. (*Middle-left/center/right*) Outputs from the bottom, middle and top layers, respectively. (*Right*) Ground truth

stronger detectors, learning label priors, learning CRF parameters, and using the new top-layer sampling approach.

7.2 eTRIMS Database

As can be seen in Table 5, we outperform all previous results reported on the eTRIMS dataset in terms of overall pixel accuracy. It is important to note that even though the methods of Fröhlich et al. (2012), Tyleček and Šára (2013) achieve higher class average, their pixel accuracy is still lower than ours. This can be explained with the poor performance of the *pavement* class in our approach, especially after the smoothing effect of our CRF, which increases the confusion with the bordering segments (mainly *road*). One of the reasons for

Table 4 Comparison of our approach to the Reinforcement Learning (RL) approach of (Teboul et al. 2013) with different merit functions, on the ECP dataset

	Pixel avg	Class avg
ATLAS (ours)		
BL	84.75	76.67
ML	**88.07**	83.36
TL	88.02	**85.22**
RL (Teboul et al. 2013)		
RF merit	73.24	65.66
BL merit	82.41	72.58
ML merit	83.10	76.17

Bold values refer to the best result
RF: Random Forest; BL, ML, TL: Our bottom, middle and top layer output, respectively

this behaviour of the CRF is that its parameters are learned on a relatively small validation set (10 images), reducing the effect of unary potentials.

The difference between our bottom and middle layer is most apparent for the *window*, *car*, and *door* classes. These are the very classes for which we had trained object detectors. Additionally, the *road* class performance is significantly boosted due to the smoothing effect of the CRF (unfortunately, at the cost of the aforementioned *pavement* class). Finally, in the top layer, we only improve the performance of the *window* class, which is not surprising, as this is the only class in this dataset for which we use weak architectural principles. We also observe a 3 % performance drop for the *door* and *building* classes. By analyzing the output data, we can see that the door accuracy drops due to the rectangularization process (see Sect. 6.1.1). Since some doors were partially covered by *window* detections in the middle layer, they were re-labeled as windows when defining rectangular window regions. Furthermore, many of the buildings in eTRIMS contain window shutters, which are annotated with the *building* class in the ground truth. Our generic detector, on the other hand, is trained on data which includes the shutters in the window structure, therefore increasing the confusion between the *window* and *building* class. Even though the final pixel accuracy of the top layer is slightly lower than the accuracy of the middle layer, the resulting labeling is more visually pleasing, as can be observed in Fig. 14. Compared to our previous work (Martinović et al. 2012), we notice a significant increase in the performance of almost all classes.

7.3 Computing Times

We performed all of our experiments on an Intel Core i7 870 CPU with 8 cores. Table 6 shows the average computing times on the ECP dataset, differentiated with respect to the different layers. Please note that the training phase differs for each method. As said in Sect. 4, the SVM classifier training is performed on the training set, while detector label maps (Sects. 5.3, 5.4) and the CRF parameters are learned on the validation set. The training protocol for detectors is described in Sect. 5.1.

7.4 Application: Image-Based Procedural Modeling

We use the output of the top layer in a straightforward procedural modeling scenario, encoding the facade as a set of CityEngine CGA rules (Esri 2013). The 7 different classes from the ECP dataset correspond to the terminal symbols of the procedural grammar. However, we make a distinction between facade element classes (*window*, *balcony*, *door)* and region classes (*wall*, *roof*, *sky*, *shop*). Each element class is modeled in a separate layer and then overlayed on the vertical split of facade regions. For example, we create the *window* layer by extracting the binary mask of windows from the output of our system. This binary mask is subdivided into rows and columns of elements, and encoded as a set of splitting

Table 5 Performance on the eTRIMS dataset

	Building	Car	Door	Pavement	Road	Sky	Vegetation	Window	**Pixel avg**	**Class avg**
CRF (Yang and Förstner 2011b)	71	35	16	22	35	78	66	75	65.80	49.75
HCRF (Yang and Förstner 2011a)	67	36	14	85	53	80	78	80	69.00	61.63
ICFHGS- (Fröhlich et al. 2012)	–	–	–	–	–	–	–	–	77.22	**72.23**
SPT (Tyleček and Šára 2013)	89	70	37	64	68	81	84	68	82.10	70.13
3Layer (Martinović et al. 2012)	86	67	18	35	47	91	81	80	80.81	63.20
ATLAS (ours)										
Bottom layer	91	61	26	29	51	94	82	66	80.42	62.52
Middle layer	91	74	50	15	73	97	87	73	**83.39**	70.00
Top layer	89	73	49	15	73	97	87	75	82.90	69.81

Class accuracies are shown in percent. The per-class results from (Fröhlich et al. 2012) were obtained on only one cross-validation fold, thus we do not report them

Table 6 Computing times for our method on the ECP dataset

		Train	Test
Bottom layer	Segmentation	–	3.85 s
	Feature extraction	–	3.15 s
	Classifier	18 min	3.05 s
Middle layer	Detector	8 h	2.1 s
	Label maps	1 min	–
	CRF	70 m	3.5 s
Top layer	Subsampling and optimization	–	180 s

'Train': total time spent during training for each method. Items marked with '–' in the 'Train' row denote that the method has no training phase. 'Test': computing times for one test image. Note that label map learning is a learned prior, so it has no computing time during testing

CGA rules. The terminal symbols are then replaced with 3D models from a library of architectural elements. Finally, the texture of the original image is projected onto the *wall* and *shop* area, to create a more realistic visualization. Several rendered models can be seen in Fig. 16. Note that our approach is able to handle non-aligned roof windows correctly (first row), while it is not forced to hallucinate non-existing ones (second row).

8 Conclusion and Future Work

We proposed a new method for facade parsing which is divided into three layers. For the bottom layer we explored a variety of different segmentation and classifier combinations to get our initial bottom up facade labeling. In the middle layer we then introduced the usage of object detectors to improve over the initial labeling. The results from the bottom and the object detector responses are combined in a principled way by using a CRF formulation, where the weights of the different CRF terms are estimated automatically. Finally in the top layer we added facade specific information via *weak architectural principles*. We proposed a general framework in which principles can be removed or added. This facilitates the usage of this layer for other facade styles. The output of our top layer are architecturally plausible facade structures with clearly defined boundaries and structures. Our method was evaluated on two datasets and shows state of the art performance.

In a final step, we demonstrated how the output of our top layer can directly be used for the image-based procedural modeling of facades. Instead of building our system upon a previously defined grammar – as demonstrated in the previous chapter – we could actually infer procedural rules from the output of our system. These rules are instance specific and if extracted from a single building can in that case only be used to generate a model of that building. In a recently

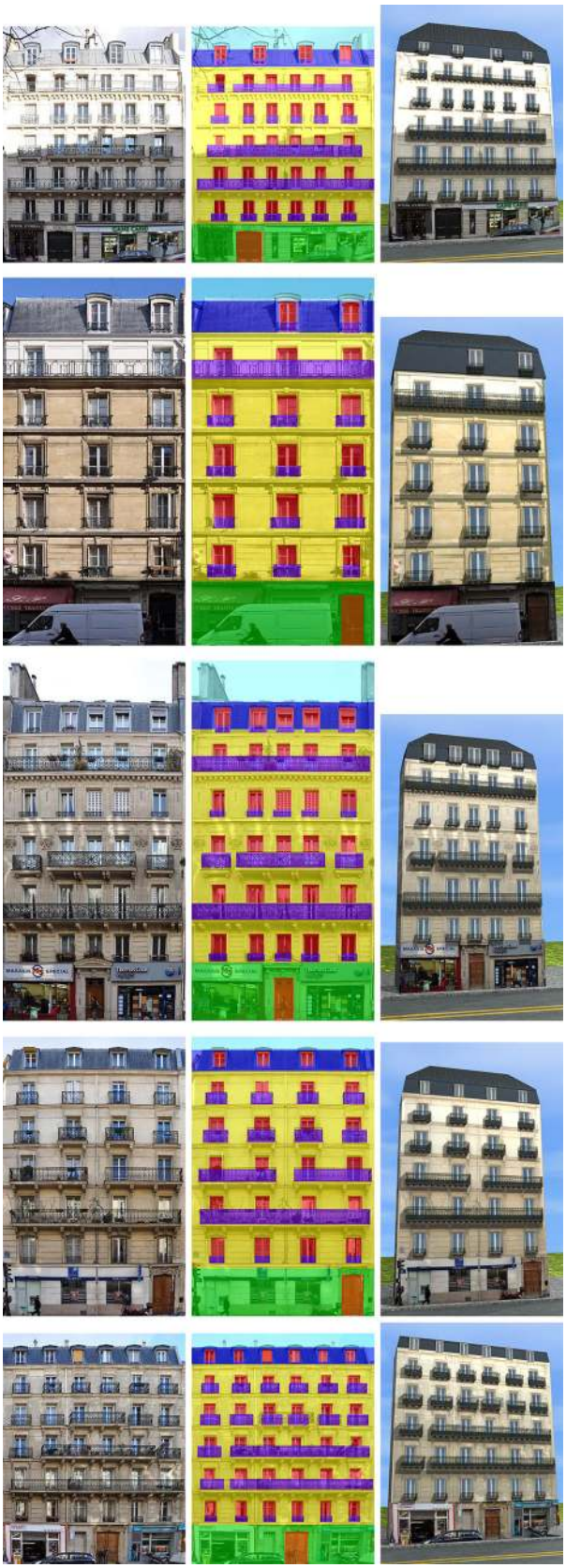

Fig. 16 Results on the ECP dataset. (*Left*) Input image. (*Middle*) Our top-layer labeling. (*Right*) Procedural building model

published work it is shown that a probabilistic Haussmann grammar can be learned automatically by using the ground truth image annotations of multiple buildings of the ECP dataset (Martinović and Van Gool 2013). Given this result, as a future work, we plan to combine the grammar learning with our approach. This means that the grammar has to be learned from noisy input data in contrast to learning it from ground truth annotations. The complete pipeline would bypass the tedious task of generating procedural facade grammars manually for many different facade styles and result in procedural grammars from which buildings of a certain style can be sampled.

Acknowledgments This paper is supported by EC FP7 Integrated Project 3D-COFORM, Grant No. 231809, and ERC Advanced Grant VarCity, Grant No. 273940.

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2010). *SLIC superpixels*. Technical Report, EPFL.

Alegre, O. & Dellaert, F. (2004). A probabilistic approach to the semantic interpretation of building facades. In *Workshop on vision techniques applied to the rehabilitation of city centres* (pp. 1–12).

Aliaga, D. G., Rosen, P. A., & Bekins, D. R. (2007). Style grammars for interactive visualization of architecture. *IEEE Transactions on Visualization and Computer Graphics*, *13*(4), 786–797.

Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(5), 898–916.

Benenson, R., Mathias, M., Timofte, R., & Van Gool, L. (2012). Pedestrian detection at 100 frames per second. In *IEEE conference on computer vision and pattern recognition*.

Benenson, R., Mathias, M., Tuytelaars, T., & Van Gool, L. (2013). Seeking the strongest rigid detector. In *IEEE conference on computer vision and pattern recognition*.

Bokeloh, M., Wand, M., & Seidel, H. P. (2010). A connection between partial symmetry and inverse procedural modeling. *SIGGRAPH*, *29*(4), 104:1–104:10.

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(11), 1222–1239.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*, 27:1–27:27.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 603–619.

Dai, D., Prasad, M., Schmitt, G., & Van Gool, L. (2012). Learning domain knowledge for facade labeling. In *European conference on computer vision*.

Deep, K., Singh, K. P., Kansal, M., & Mohan, C. (2009). A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Applied Mathematics and Computation*, *212*(2), 505–518.

Demuth, H., & Beale, M. (1993). Neural network toolbox for use with MATLAB: User guide. Math Works.

Dick, A. R., Torr, P. H. S., & Cipolla, R. (2004). Modelling and interpretation of architecture from several images. *International Journal of Computer Vision*, *60*, 111–134.

Dollar, P., Tu, Z., Perona, P., & Belongie, S. (2009). Integral channel features. In *British machine vision conference*.

Esri. (2013). CityEngine. http://www.esri.com/software/cityengine. Accessed 1 July 2015.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*(2), 303–338.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, *59*, 2004.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010a). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*, 1627–1645.

Felzenszwalb, P. F., Girshick, R. B., & McAllester, D. A. (2010b). Cascade object detection with deformable part models. In *IEEE conference on computer vision and pattern recognition* (pp. 2241–2248).

Fletcher, R. (1987). *Practical methods of optimization* (2nd ed.). New York, NY: Wiley.

Floros, G., Rematas, K., & Leibe, B. (2011). Multi-class image labeling with top-down segmentation and generalized robust P^N potentials. In *British machine vision conference* (pp. 1–11).

Fröhlich, B., Rodner, E., & Denzler, J. (2012). Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach. In *Asian conference on computer vision* (pp. 218–231).

Fröhlich, B., Rodner, E., Kemmler, M., & Denzler, J. (2013). Large-scale gaussian process multi-class classification for semantic segmentation and facade recognition. *Machine Vision and Applications*, *24*(5), 1043–1053.

Gilks, W., Richardson, S., & Spiegelhalter, D. (1995). *Markov Chain Monte Carlo in practice*. London: Chapman & Hall/CRC Interdisciplinary Statistics, Taylor & Francis.

Girshick, R. B., Felzenszwalb, P. F., & McAllester, D. (2012). Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/~rbg/latent-release5/. Accessed 1 June 2015.

Gould, S., Fulton, R., & Koller, D. (2009a). Decomposing a scene into geometric and semantically consistent regions. In *International conference on computer vision* (pp 1–8).

Gould, S., Russakovsky, O., Goodfellow, I., Baumstarck, P., Ng, A.Y., & Koller, D. (2009b). The stair vision library (v2.2). http://ai.stanford.edu/~sgould/svl. Accessed 1 Feb 2013.

Han, F., & Zhu, S. C. (2009). Bottom-up/top-down image parsing with attribute grammar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(1), 59–73.

Han, T., Liu, C., Tai, C. L., & Quan, L. (2012). Quasi-regular facade structure extraction. In *Asian conference on computer vision* (pp. 552–564).

Heitz, G., & Koller, D. (2008). Learning spatial context: Using stuff to find things. In *European conference on computer vision* (pp. 30–43).

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., & Igel, C. (2013). Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International joint conference on neural networks*

Karp, R. (1972). Reducibility among combinatorial problems. In R. Miller & J. Thatcher (Eds.), *Complexity of computer computations* (pp. 85–103). New York: Plenum Press.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Kolmogorov, V., & Zabih, R. (2004). What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 65–81.

Korah, T., & Rasmussen, C. (2008). Analysis of building textures for reconstructing partially occluded facades. In *European conference on computer vision* (pp. 359–372).

Korč, F., & Förstner, W. (2009). eTRIMS image database for interpreting images of man-made scenes. Technical Report TR-IGG-P-2009-01.

Kumar, M. P., & Koller, D. (2010). Efficiently selecting regions for scene understanding. In *IEEE conference on computer vision and pattern recognition* (pp. 3217–3224).

Kumar, S., August, J., & Hebert, M. (2005). Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. *Proceedings of the 5th international conference on energy minimization methods in computer vision and pattern recognition* (pp. 153–168). Berlin: Springer.

Ladicky, L., Sturgess, P., Alahari, K., Russell, C., & Torr, P. H. S. (2010). What, where and how many? Combining object detectors and crfs. In *European conference on computer vision* (pp. 424–437).

Leibe, B., Leonardis, A., & Schiele, B. (2006). An implicit shape model for combined object categorization and segmentation. In *Toward category-level object recognition* (pp. 508–524).

Leibe, B., Cornelis, N., Cornelis, K., & Van Gool, L. J. (2007). Dynamic 3d scene analysis from a moving vehicle. In *IEEE conference on computer vision and pattern recognition*.

Leibe, B., Cornelis, N., Cornelis, K., & Van Gool, L. J. (2007). Dynamic 3d scene analysis from a moving vehicle. In *IEEE conference on computer vision and pattern recognition*.

Liu, C., Yuen, J., & Torralba, A. (2011). Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(12), 2368–2382.

Martinović, A., & Van Gool, L. (2013). Bayesian grammar learning for inverse procedural modeling. In *IEEE conference on computer vision and pattern recognition*.

Martinović, A., Mathias, M., Weissenberg, J., & Van Gool, L. (2012). A three-layered approach to facade parsing. In *European conference on computer vision* (pp. 416–429).

Mathias, M., Martinović, A., Weissenberg, J., Haegler, S., & Van Gool, L. (2011a). Automatic architectural style recognition. In *ISPRS international workshop 3D-ARCH*.

Mathias, M., Martinović, A., Weissenberg, J., & Van Gool, L. (2011b). Procedural 3d building reconstruction using shape grammars and detectors. In *International conference on 3D imaging* (pp. 304–311). Processing, Visualization and Transmission: Modeling.

Mathias, M., Timofte, R., Benenson, R., & Van Gool, L. (2013). Traffic sign recognition—How far are we from the solution? In *International joint conference on neural networks*.

Mathworks. (2014). *Global optimization toolbox documentation*. Natick, Massachusetts.

Müller, P., Zeng, G., Wonka, P., & Van Gool, L. (2007). Image-based procedural modeling of facades. *SIGGRAPH*, *26*(3), 85.

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Gool, L. V., Purgathofer, W., et al. (2012). In *State of the Art Reports, Eurographics Association, EG STARs* (pp. 1–28).

Nowozin, S., Gehler, P. V., & Lampert, C. H. (2010). On parameter learning in crf-based approaches to object class image segmentation. In *European conference on computer vision* (pp. 98–111).

Ok, D., Kozinski, M., Marlet, R., & Paragios, N. (2012). High-level bottom-up cues for top-down parsing of facade images. In *International conference on 3D imaging, processing, visualization and transmission: Modeling* (pp. 128–135).

Recky, M., Wendel, A., & Leberl, F. (2011). Façade segmentation in a multi-view scenario. In *International conference on 3D imaging, processing, visualization and transmission: Modeling* (pp. 358–365).

Riemenschneider, H., Krispel, U., Thallerm W., Donoser, M., Havemann, S., Fellner, D. W. & Bischof, H. (2012). Irregular lattices for complex shape grammar facade parsing. In *IEEE conference on computer vision and pattern recognition* (pp. 1640–1647).

Ripperda, N., & Brenner, C. (2006). Reconstruction of façade structures using a formal grammar and rjmcmc. In *Pattern recognition—DAGM symposium* (pp. 750–759).

Russell, S. J., Norvig, P., Candy, J. F., Malik, J. M., & Edwards, D. D. (1996). *Artificial intelligence: A modern approach*. Upper Saddle River, NJ: Prentice-Hall Inc.

Shechtman, E., & Irani, M. (2007). Matching local self-similarities across images and videos. In *IEEE conference on computer vision and pattern recognition*.

Shen, C. H., Huang, S. S., Fu, H., & Hu, S. M. (2011). Adaptive partitioning of urban facades. *ACM Transactions on Graphics*, *30*(6), 184.

Shotton, J., Winn, J., Rother, C., & Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Internation Journal of Computer Vision*, *81*(1), 2–23.

Simon, L., Teboul, O., Koutsourakis, P., Gool, L. J. V., & Paragios, N. (2012). Parameter-free/pareto-driven procedural 3d reconstruction of buildings from ground-level sequences. In *IEEE conference on computer vision and pattern recognition* (pp. 518–525).

Socher, R., Lin, C. C., Ng, A. Y., & Manning, C. D. (2011). Parsing natural scenes and natural language with recursive neural networks. In *International conference on machine learning*.

Szummer, M., Kohli, P., & Hoiem, D. (2008). Learning crfs using graph cuts. In *European conference on computer vision* (pp. 582–595).

Taskar, B., Chatalbashev, V., Koller, D., & Guestrin, C. (2005). Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on machine learning, ICML '05* (pp. 896–903). New York, NY: ACM.

Teboul, O. (2010). Ecole centrale paris facades database. http://vision.mas.ecp.fr/Personnel/teboul/data.php. Accessed 2014-11-01.

Teboul, O., Simon, L., Koutsourakis, P., & Paragios, N. (2010). Segmentation of building facades using procedural shape priors. In *IEEE conference on computer vision and pattern recognition* (pp. 3105–3112).

Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., & Paragios, N. (2013). Parsing facades with shape grammars and reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(7), 1744–1756.

Tighe, J., & Lazebnik, S. (2013a). Finding things: Image parsing with regions and per-exemplar detectors. In *IEEE conference on computer vision and pattern recognition*

Tighe, J., & Lazebnik, S. (2013b). Superparsing—Scalable nonparametric image parsing with superpixels. *International Journal of Computer Vision*, *101*(2), 329–349.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, *6*, 1453–1484.

Tyleček, R., & Šára, R. (2013). Spatial pattern templates for recognition of objects with regular structure. In J. Weickert, M. Hein, & B. Schiele (Eds.), *Pattern recognition. Lecture Notes in Computer Science* (pp. 364–374). Berlin: Springer.

Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., & Van Gool, L. (2012). Seeds: Superpixels extracted via energy-driven sampling. In *European conference on computer vision*.

Weissenberg, J., Riemenschneider, H., Prasad, M., & Van Gool, L. (2013). Is there a procdural logic to architecture? In *IEEE conference on computer vision and pattern recognition*

Wendel, A., Donoser, M., & Bischof, H. (2010). Unsupervised facade segmentation using repetitive patterns. In *Pattern recognition—DAGM symposium* (pp. 51–60).

Wojek, C., & Schiele, B. (2008). A dynamic conditional random field model for joint labeling of object and scene classes. In *ECCV (4)* (pp. 733–747). Berlin: Springer.

Wu, F., Yan, D. M., Dong, W., Zhang, X., & Wonka, P. (2013). Inverse procedural modeling of facade layouts. *CoRR*. arXiv:1308.0419.

Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., & Quan, L. (2008). Image-based façade modeling. In *SIGGRAPH Asia*.

Xiao, J., Fang, T., Zhao, P., Lhuillier, M., & Quan, L. (2009). Image-based street-side city modeling. In *SIGGRAPH* (Vol. 28, No. 5).

Yang, C., Han, T., Quan, L., & Tai, C. L. (2012). Parsing façade with rank-one approximation. In *IEEE conference on computer vision and pattern recognition* (pp. 1720–1727).

Yang, M., & Förstner, W. (2011a). A hierarchical conditional random field model for labeling and classifying images of man-made scenes. In *2011 IEEE international conference on computer vision workshops (ICCV Workshops)* (pp. 196–203).

Yang, M. Y., & Förstner, W. (2011b). Regionwise classification of building facade images. In *Photogrammetric image analysis* (Vol. 6952, pp. 209–220). Springer, LNCS.

Zhang, H., Xu, K., Jiang, W., Lin, J., Cohen-Or, D., & Chen, B. (2013). Layered analysis of irregular facades via symmetry maximization. *ACM Transactions on Graphics*, *32*(4), 1–13.

Zhao, P., Fang, T., Xiao, J., Zhang, H., Zhao, Q., & Quan, L. (2010). Rectilinear parsing of architecture in urban environment. In *IEEE conference on computer vision and pattern recognition* (pp. 342–349).