Efficient Facade Segmentation using Auto-Context

Varun Jampani*
MPI for Intelligent Systems
Tübingen, Germany
varun.jampani@tuebingen.mpg.de

Raghudeep Gadde*
Université Paris-Est, LIGM
ENPC, ECP, Paris, France
gadder@imagine.enpc.fr

Peter V. Gehler
MPI for Intelligent Systems
Tübingen, Germany
pgehler@tuebingen.mpg.de

Abstract

In this paper we propose a system for the problem of facade segmentation. Building facades are highly structured images and consequently most methods that have been proposed for this problem, aim to make use of this strong prior information. We are describing a system that is almost domain independent and consists of standard segmentation methods. A sequence of boosted decision trees is stacked using auto-context features and learned using the stacked generalization technique. We find that this, albeit standard, technique performs better, or equals, all previous published empirical results on all available facade benchmark datasets. The proposed method is simple to implement, easy to extend, and very efficient at test time inference.

1. Introduction

In this paper we consider the problem of segmenting the pixels in images of building facades into different semantic classes. An example image from a common benchmark dataset for this problem is shown in Fig. 1 along with its provided manual annotation. This problem is a core component of several real world applications, including urban modeling, or automatic generation of virtual cities with specific building styles. Images of buildings exhibit a strong structural organization due to architectural design choices and construction constraints. For example, windows are usually not placed randomly, but on the same height, usually with a vertical ordering; a door can only be found on street-level, etc.

This problem is also an interesting test-bed for general purpose segmentation methods that allow including such strong prior knowledge. Therefore it is not surprising that most recent approaches to this problem focus on ways to include structural information in a principled way. Some examples are Conditional Random Field (CRF) models that use higher order potential functions [23, 14]. Another route is using grammar-based models that include generative rules [15, 20, 12] and try to infer the production rules from image evidence.

In contrast to these approaches, we largely ignore domain specific knowledge and describe a generic segmentation method that is easy to implement, has fast test time inference, and is easily adaptable to new datasets. The system is a sequence of boosted decision tree classifiers, that are stacked using auto-context [21] features and learned using stacked generalization [22]. The entire pipeline consists of established components and we consider it to be a baseline method for this task. Surprisingly, we find that this method outperforms or equals all published approaches on all available diverse benchmark datasets. Therefore, this approach defines a new state-of-the-art method in terms of empirical performance. Experiments suggest that more domain specific models would benefit from better unary predictions. Moreover our findings also suggest that previous methods need to be carefully re-evaluated in terms of a relative improvement compared to a method like the proposed one.

A pixel-wise facade classification might not be a desired output for some applications. For instance, high level structural information is needed to synthesize new facades in a virtual city. To show the usefulness of pixel-level segmentation result in obtaining accurate high-level information, we used our pixel-level predictions in a procedural modeling system [20] that aims to recover production rules of the facade image. This output is of interest in different applications and we show that an improved pixel-wise prediction directly translates into a better facade parsing result.

2. Related Work

Approaches to facade segmentation can be broadly classified into two categories: *bottom-up methods* [11, 14, 23, 1] that use pixel level classifiers in combination with CRF models and *top-down methods* [20, 15, 12] that use shape grammars. The shape grammar methods seek to parse a facade segmentation into a set of production rules and element attributes. The central idea is to represent the facade using a parse tree and compute the best possible derivation of a spe-

*The first two authors contribute equally to this work.

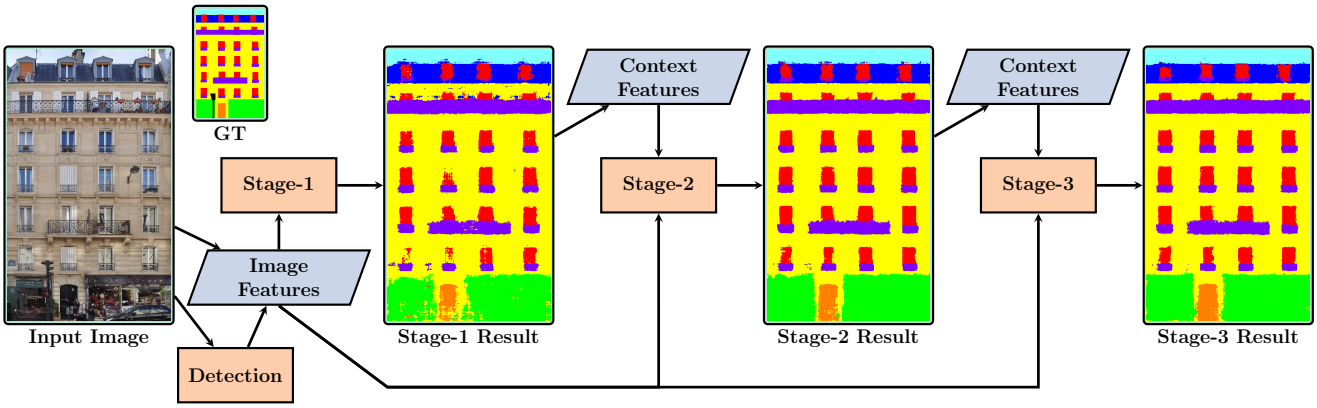


Figure 1. Schematic of different components in our facade segmentation pipeline with a sample facade from ECP dataset [19].

cific grammar towards optimal segmentation of the image. The high structural organization of facades due to architectural design choices make such a generative approach a natural model candidate. However they are not pixel accurate and not easily amendable to efficient inference which often leads to inefficient and sub-optimal segmentation results. As a consequence, the state-of-the-art methods in terms of pixel accuracy are dominated by the bottom-up methods.

In [11], a three-layered system is proposed. A first layer use a recursive neural network to obtain pixel label probabilities which are fed into a grid CRF model in a second layer along with object detections. The third layer enforces weak architectural principles in facades as post-processing. This setup combines high-level and low-level information into a single prediction. The runtime of this system is mentioned in [1] to be about 2 minutes for an image of size 500 by 300. The works [23, 14] incorporate architectural knowledge in a single CRF framework using higher order potential functions. The method of [23] proposes a hierarchical CRF framework to encode inter-class location information in facades. The work of [14] uses long range pairwise and ternary potentials to encode the repetitive nature of various class regions. Both methods require specific inference techniques that result in non-negligible runtimes. The approach of [1] is to use a sequence of dynamic programming runs that search for optimal placement of window and balcony rows, door location and others. Every single step is very fast and the overall system is mostly global optimal. The downside is that the sequence and type of classifications needs to match facade architecture type.

All the discussed methods build on top of pixel label probabilities which are obtained using pixel classifiers. It is only after those have been obtained, that architectural constraints are taken into account. This is different to the system we describe in this paper, where low-level pixel classifications are obtained directly from image, detection, and auto-context features.

The closest to our work are [7] and [8] which also proposed auto-context based methods for facade segmentation. The work of [7] incorporated auto-context features in random decision forests where the classification results from top-layers of trees are used to compute auto-context features and are then used in training the lower layers of the trees in forest. More recently, [8] proposed to use the local Taylor coefficients computed from the posterior at different scales as auto-context. Although conceptually similar, the proposed method uses different low-level features, auto-context features and learning techniques. Experimental comparisons suggest a superior performance of the proposed system.

3. Auto-Context Segmentation

The overall architecture that we use combines standard segmentation methods. Boosted decision trees are stacked with the use of auto-context [21] features from the second layer onward. Learning is implemented using stacked generalization [22]. We will describe the ingredients in the following, starting with the segmentation algorithm (Sec. 3.1), the feature representation (Sec. 3.2), auto-context features (Sec. 3.3), and the training architecture (Sec. 3.4).

Given an image I, the task of segmentation is to classify every pixel i into one of C classes $c_i \in \{1, \ldots, C\}$. For training we are given a set of N training examples $(I^j, Y^j), j = 1, \ldots, N$ of images along with a manual annotation Y^j. We will comment on the loss function in the experimental section and for now treat the problem as one that decomposes over the set of pixels. Two different feature sets are distinguished, image features $f_i \in \mathbb{R}^{D_f}$ that are derived from the RGB observations and auto-context features $a_i \in \mathbb{R}^{D_a}$ based on prediction results from previous stages.

3.1. Architecture

Our system consists of an iterative sequence of classifiers as suggested in [21]. A schematic overview of the

pipeline is depicted in Fig. 1. At every stage t, the classifier has access to the image and to predictions of all earlier stages. Formally, at stage $t > 1$ a classifier F^t maps image and auto-context features to a probability distribution of the pixel class assignments

$$F^t\left(f_i(I), a_i(P^{t-1})\right) \mapsto P^t(c_i|I), \forall i. \quad (1)$$

For pixel classifier F^t, we use boosted decision trees that store conditional distributions at their leaf nodes, in general the output of F^t need not be a distribution. The first stage $t = 1$ depends only on the features $F^1(f_i(I))$ derived directly from the image.

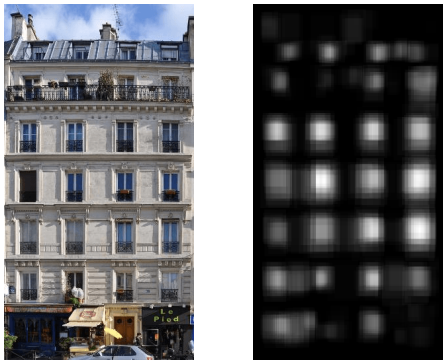
This architecture is a conceptually easy but efficient way to use contextual information in pixel level classification. Classifiers of later stages can correct errors that earlier stages made. An example sequence of predictions can be seen in Fig. 1. For example, an auto-context feature can encode the density of a predicted class around a pixel. The classifier can learn that certain classes only appear in clusters which then allows to remove spurious predictions. This has a similar smoothing effect as some pairwise CRF models have but with the benefit of a much faster inference.

For training and testing we used the DARWIN toolbox [9]. The maximum tree-depth of each classifier is set to 2 and we used a maximum of 200 boosting rounds. The runtime of the system is summarized in Table 1. These numbers are computed on an Intel Core(TM) i7-4770 CPU @ 3.40 GHz for a common image from the ECP dataset (about 500×400 pixels). The runtime for the features is the sequential computation for all features.

3.2. Image Features

As image features, we computed 17 TextonBoost filter responses [18], location information, RGB color information, dense Histogram of Oriented Gradients [2], Local Binary Pattern features, and all filter averages over image rows and columns at each pixel. These again are computed using the DARWIN [9] toolbox.

In addition to the above generic segmentation features we include detection scores for some specific objects. Following [11], we use detectors for windows and doors. Whereas [11] fused the detection scores into the output of the pixel classifiers, we turned the detection scores into image features at every single pixel. We use the integral channel features detector from [4] for which a toolbox is available [3]. For a given image, the detector outputs a number of bounding boxes along with a corresponding score for each bounding box. We sum up the scores to get a single detection score at each pixel. Object detection parameters are automatically estimated using the training data to get good recall. Fig. 2 shows an example window detection output for a sample facade image. The detection feature is of course a problem dependent one and based on the prior knowledge about the special classes door and windows. However it is still a generic feature in the sense that the prior information is very weak and a generic detection system has been used to obtain it. Moreover, door and window classes are common to any architecture of facades. 761 low-level image features coupled with 2 door and window detection features make a total of 763 feature values at each pixel.



(a) Facade (b) Detection

Figure 2. A facade and the corresponding window detection output. Bright and dark regions correspond to high and low detection scores respectively.

3.3. Auto-context Features

In addition to the image features, the classifiers from stage $t > 1$ can condition on statistics computed from previous predictions. We include the auto-context features a_i that are being computed from predictions of the previous classifier $P^{t-1}(\cdot|I)$ only. For every pixel i we compute the following auto-context features of length $14C + 1$.

Class probability The probability $P^{t-1}(c_i|I)$. (length C).

Entropy The entropy of $P^{t-1}(\cdot|I)$. This feature quantifies the ambiguity of the $t-1$ stage prediction (length 1) .

Row and column scores We compute the percentage of predicted classes in the row and column of pixel i. Along with this percentage we compute the average score of all pixels in the same row and column as i (length $4C$) .

Distance to the nearest class pixel Both Euclidean and Manhattan distances to the nearest class pixel are computed as features (length $2C$) .

Class color model For every class c we fit, with maximum likelihood, a Gaussian distribution to the RGB values of all those pixels that are being predicted to be class c. To be more robust, we fit the distribution only to those pixels with probabilities greater than the $3rd$ quartile. For every pixel we then calculate the log-likelihood for all classes (length C) .

Bounding box features For every class, we fit a rectangular bounding box to every connected component of MAP

predictions. For every pixel we compute a C vector with the c'th component being a 1 or 0 depending on whether it lies inside or outside of a box for class c. A variant of this feature is to compute the average class probability inside the box. This feature aims to improve the segmentation of rectangular objects such as doors and windows (length $2C$) .

Neighborhood statistics For every pixel, the average class probability is computed in a 10×5 region above and below the pixel; and also in a 5×10 region left and right to that pixel (length $4C$) .

3.4. Stacked Generalization

We train the sequence of classifiers using the method of stacked generalization [22]. The training data is split in four folds and four different models are trained using three folds, with one fold held out. The four models are used to obtain prediction on the held out fold, this results in a set of cross-validation predictions. It is from these predictions that the auto-context features for training are computed. The next stage classifier is trained subsequently, in the same manner. For every stage, one classifier is trained using the entire training data (all four folds) and used during test time inference.

During training, the auto-context features are thus computed using different classifiers, different also from the classifier that is being used at test time. The reason for this procedure is to obtain features that are not computed on training predictions and thus do not overfit to the data. This procedure is a standard strategy and found to be stable and well performing in many scenarios. In the experiments we use a total of three classification stages and, experiments indicated that the performance of a fourth stage levels out.

4. Experiments

We evaluate the auto-context pipeline on all five benchmark datasets that are available for the problem of facade segmentation. For all datasets except LabelMeFacade dataset, we report five fold cross-validation results, the standard protocol used in the literature. One fold cross-validation is done for LabelMeFacade dataset as the training and testing data are pre-specified for this dataset. We compare against all recent best performing methods.

As a performance measure we use the overall pixel-wise classification accuracy and the accuracy averaged over the classes. These are the two standard results that are reported in the literature. In addition we also report the intersection over union (IoU) score, popularized by the VOC segmentation challenges [5]. The IoU score is a higher-order loss function and Bayes optimal prediction requires dedicated inference techniques. For simplicity, we report MAP predictions for all pixels and evaluate all three measures on this prediction. The three measures in terms of false positives (FP), true positives (TP), and false negatives (FN) are defined as follows.

- *Pixel Accuracy*: "TP / (TP + FN)" computed over entire image pixels of all classes.
- *Average Class Accuracy*: Pixel accuracy computed for all classes separately and then averaged.
- *Intersection Over Union Score* (IoU): "TP / (TP + FN + FP)" computed on every class and then averaged.

The performance differences are tested for statistical significance. We used a paired t-test with one tail and $p < 0.01$.

4.1. Datasets

ECP Dataset The ECP dataset [19] consists of 104 rectified facade images of Hausmannian architectural buildings from Paris. For five-fold cross validation, we divided the training data into 4 sets of 20 images and 1 set of 24 images. There are seven semantic classes in this dataset.

Graz Dataset This dataset [15] has 50 facade images of various architectures (Classicims, Biedermeier, Historicism, Art Nouveau) from buildings in Graz. There are only four semantic classes, and the data is divided into 5 equal sets for cross-validation.

eTRIMS Dataset The eTRIMS dataset [10] consists of 60 non-rectified images. Facades in this dataset are more irregular and follow only weak architectural principles. Again, we split the data into 5 equal sets for cross-validation.

CMP Dataset This dataset, proposed in [14], has 378 rectified facades of diverse styles and 12 semantic classes in its base set. We divided the data into 4 sets of 75 images each and one set of 78 images for cross-validation.

LabelMeFacade Dataset Introduced in [6], this dataset has 100 training and 845 testing facade images taken from LabelMe segmentation dataset [17]. Facades in this dataset are highly irregular with lot of diversity across images.

4.2. Results

The empirical results on the ECP dataset are reported in Table 2, the results on the remaining four datasets are summarized in Table 3.

The first observation we make is that the use of a stacked auto-context pipeline improves the results on all the datasets. On the ECP dataset, the improvement is 1.95% in terms of overall pixel-accuracy for a three-stage classifier (ST3) compared to single stage classifier (ST1). The ordering in terms of statistically significant performance is on the ECP, CMP and LabelMeFacade datasets ST3>ST2>ST1 and on eTrims and Graz ST3=ST2>ST1. The auto-context

Method	Features	AC-Features	ST1	ST2	ST3	PW	[11]	[1]
Time (s)	3.0	0.6	+0.04	+0.64	+0.64	+24	110	2.8

Table 1. Average runtime for various methods. 'Features' correspond to low-level and object detection image features (computed once). 'AC' corresponds to Auto-Context features. The classifier runs at 0.04 seconds, every stage needs to additionally compute AC features. A Potts model using alpha expansion takes on average 24s. Inference times (excluding unary computation) of existing methods are also shown for comparison.

Class				Auto Context			AC + Potts Model			Grammar Parsing		
	[11]	[1]-1	[1]-2	ST1	ST2	ST3	PW1	PW2	PW3	[20]	[12]	ST3+[20]
Door	60	79	**82**	76.2	79.2	80.4	77.9	79.9	81.3	47	50	64.2
Shop	86	94	**96**	87.6	90.4	91.6	90.5	92.1	93.2	88	81	90.1
Balcony	71	91	**92**	85.8	89.4	89.4	86.4	89.3	89.3	58	49	71.4
Window	69	85	**87**	77.0	80.7	81.8	77.1	81.0	82.3	62	66	75.6
Wall	**93**	90	88	91.9	92.3	92.4	**93.0**	**93.0**	**92.9**	82	80	**92.5**
Sky	97	97	96	97.3	**97.9**	**98.0**	**97.8**	**98.1**	**98.2**	95	91	96.1
Roof	73	90	**93**	86.5	88.1	88.1	88.4	89.3	89.2	66	71	77.2
Average	78.4	89.4	**90.6**	86.04	88.28	88.79	87.31	88.94	89.49	71.1	69.7	81.0
Overall	85.06	**90.82**	90.34	88.86	**90.49**	**90.81**	90.02	**91.12**	**91.42**	74.71	74.82	85.2
IoU	-	-	-	75.25	78.62	79.32	77.57	79.88	**80.54**	-	-	72.4

Table 2. Segmentation results of various methods on ECP dataset. ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. PW1, PW2, and PW3 refer to a Potts model over the classification unaries. Published results are shown for comparisons. The parsing result of the reinforcement learning method [20] when using the output ST3 result are reported on the right.

features are frequently selected from the boosted decision trees. For the ECP dataset, about 30% of the features in stage 2 and 3 are auto-context features (CMP 46%, eTrims 31%, and Graz 11%). We didn't notice any significant differences or trends regarding the type of auto-context features picked by the boosted decision trees for different datasets.

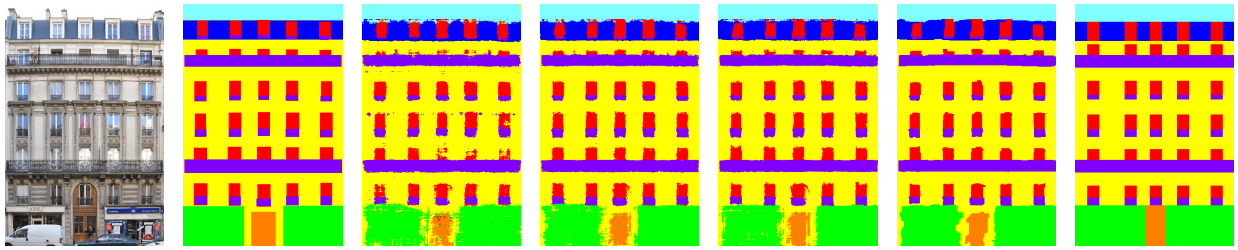
The next observation on ECP dataset is that the performance of ST3 with 90.81% is on par with the reported 90.82% from the current best performing method [1]. The results of the auto-context classifier are significantly higher on the other four datasets when compared to the methods of [15, 1, 11, 14, 7, 13, 8]. On all datasets, but ECP, even the first stage classifier produces better predictions than the previous approaches. The methods of [15, 1, 11, 14] all include domain knowledge in their architecture. For example, the system of [1] is a sequence of dynamic programs that is used to include specific domain knowledge, such as that balconies are below windows, only one door exists, or elements like windows are in fact rectangular segments. On this dataset, the authors of [1] observe an about 4% improvement over their unary classifiers accuracy*, we conjecture it also may improve the predictions of ST3.

The methods of [15, 1, 11, 14, 7, 13], use different unary predictions and therefore may profit from the output of the auto-context classifier. Unfortunately, the respective unary-only results are not reported, so at this point it is not possible to estimate the relative improvement gains of the methods. The fact that a conceptually simple auto-context pipeline outperforms, or equals, all methods on all published datasets suggests that a more careful evaluation of the relative improvements of [15, 1, 11, 14] is required.

In addition to the pixel-wise predictions of the auto-context classifiers we evaluated a CRF with an 8-connected neighbourhood and pairwise Potts potentials. The single parameter of the Potts model (weight for all classes set to equal) was optimized to yield the highest accuracy on the training set (thus possibly at the expense of losing a bit of performance compared to a cross-validation estimate). Inference is done using alpha expansion implemented in DARWIN [9]. The results of the Potts-CRF on top of the unary predictions of different staged auto-context classifiers are referred to as PW1, PW2, and PW3 in Table 2. For other datasets, only results for PW3 are shown in Table 3 with more results in supplementary†. First, we find that the Potts model is improving over the results from the auto-context stages ST1, ST2, and ST3 ($p < 0.01$). This suggests that some local statistics are not captured in the auto-context features; more local features may improve the auto-context classifiers. Second, the CRF-Potts model achieves higher accuracies than the method of [1], making it the (although only marginally) highest published result on

*personal communication

†http://fs.vjresearch.com



(a) Facade (b) GT (c) ST1 (d) ST2 (e) ST3 (f) PW3 (g) Parse

Figure 3. (a) Sample facade image from ECP dataset; (b) Ground truth segmentation; and (c,d,e) Result of various classification stages of our auto-context method. Observe that the method removes isolated predictions and recovers the second lowest line of windows. (f) Potts model on top of ST3 result, and (g) parsed result obtained by applying reinforcement learning [20] using ST3 result.

(a) eTRIMS Dataset

Class				**Auto Context**			
	[11]	**[1]**	**[8]**	**ST1**	**ST2**	**ST3**	**PW3**
Building	91	91	84	90.3	90.5	90.9	**92.5**
Car	69	70	51	63.3	74.8	72.4	**76.6**
Door	18	18	**73**	62.7	62.3	63.6	65.3
Pavement	33	33	55	43.0	46.5	47.1	**48.8**
Road	55	57	81	78.2	**82.3**	80.3	**82.1**
Sky	93	97	**99**	97.6	**98.5**	**98.6**	**98.9**
Vegetation	89	90	92	91.1	92.1	92.3	**92.9**
Window	**74**	71	78	65.9	67.1	68.4	68.2
Average	65.3	65.9	66.4	74.01	76.78	76.7	**78.14**
Overall	83.16	83.84	83.40	84.68	85.95	86.12	**87.29**
IoU	-	-	-	58.7	61.26	61.48	**63.54**

(b) LabelMeFacade Dataset

Class			**Auto Context**			
	[7]	**[13]**	**ST1**	**ST2**	**ST3**	**PW3**
Building	-	-	87.7	88.1	88.2	**92.1**
Car	-	-	47.1	53.6	54.8	**58.2**
Door	-	-	**6.52**	6.03	5.12	1.71
Pavement	-	-	24	**25.3**	**24.6**	23.3
Road	-	-	80.3	82.1	84.5	**87.6**
Sky	-	-	86.2	87.2	87.4	**88.9**
Vegetation	-	-	53.3	**57.5**	**57.6**	**57.9**
Window	-	-	20.3	22.6	**25.4**	19.5
Various	-	-	19.9	**20.6**	**21.0**	12.1
Average	**56.61**	-	47.26	49.22	49.84	49.04
Overall	67.33	71.28	71.52	72.9	73.46	**75.23**
IoU	-	35.96	37.01	38.69	39.36	**39.57**

(c) CMP Dataset

Class		**Auto Context**			
	[14]	**ST1**	**ST2**	**ST3**	**PW3**
Background	58	67.1	71.8	**72.6**	73.1
Facade	73	74.6	75.3	75.2	**79.3**
Window	61	71.6	76.1	77.0	**78.1**
Door	**54**	37.9	45.5	47.0	48.7
Cornice	41	39.1	47.5	**49.6**	**50.1**
Sill	27	21.1	32.8	**36.2**	34.6
Balcony	46	31.6	44.1	46.7	**48.1**
Blind	**48**	22.7	35.8	40.1	39.9
Deco	**24**	10.4	13	13.8	11.4
Molding	54	63.2	65.4	66.5	**67.2**
Pillar	**25**	5.71	11.2	13.6	9.78
Shop	**59**	40.9	45.6	45.6	46.8
Average	47.5	40.50	47.00	**48.65**	**48.92**
Overall	60.3	61.83	65.47	66.24	**68.08**
IoU	-	29.26	34.46	35.86	**37.47**

(d) Graz Dataset

Class	**Auto Context**				
	ST1	**ST2**	**ST3**	**PW3**	**[15]**
Door	57.3	62.4	**62.7**	**63**	41
Window	78.2	81.2	**81.5**	80.9	60
Wall	94.9	94.7	94.9	**95.8**	84
Sky	87.4	**91.2**	**90.5**	**90.6**	**91**
Average	79.47	82.40	82.42	**82.56**	69
Overall	90.18	91.02	91.16	**91.68**	78
IoU	71.25	73.31	73.25	**74.39**	58

Table 3. Segmentation results on the eTRIMS, LabelMeFacade, CMP and Graz datasets. ST1, ST2 and ST3 correspond to various classification stages in our method. PW3 corresponds to Potts model on top of ST3 result. The method of [15] parses the image into a lattice representation and is not trying to maximize pixel accuracy results.

the ECP dataset. In practice, this performance gain has to be traded-off against the inference time of alpha-expansion which is on average an additional 24 seconds for an image from the ECP dataset. Some example visual results (ST3) are shown in Fig. 4 for various dataset images (more results in the supplementary material).

Figure 4. Qualitative results on various datasets (more results in the supplementary material)

5. Procedural Modeling

A pixel-wise classification of a building facade might not be the desired output for some applications. This motivated shape grammar methods [15, 16, 20, 12] that parse the facade into a different representation than pixel-wise labeling. The aim of these top-down approaches is to infer the architectural (structural) information in facades by fitting a set of grammar rules to a pixel classifier output. Such structural information can be used to generate new facade instances for virtual cities, retrieving structurally similar facades etc. We apply the method of [20], and compare against their result that is obtained using a random forest classifier that uses color information. All other settings and the grammar are the same. For space reasons we refer to [20] for more details about the approach. The results are shown in the last three columns of Table 2. These numbers are obtained by back-projecting the parsed representation into a pixel-wise prediction. We observe that better pixel predictions directly translates to better parsing results. A substantial improvement of 10.49% is achieved, closing the gap to pixel prediction methods. This shows the importance of good pixel predictions even for models that only make use of them as an intermediate step.

6. Discussion and Conclusion

The segmentation method that we described in this work is a collection of established components. It is easy to implement, fast at test-time, and outperforms all previous approaches on all published facade segmentation datasets. It is the fastest method amongst all those that we compared

against. The runtime is dominated by feature computation, which is amendable to massive speed improvements in case a high performing implementation is required. We observe on all datasets that adding stacked classifiers using auto-context features improves the performance. For the ECP dataset, a Potts-CRF improves about another 0.6% but this comes at the expense of a severe increase in runtime.

The auto-context classifier raises the bar when it comes to absolute performance. It largely ignores domain knowledge, but still the performance is equal or higher than all methods that include prior information in some form, for example the number of doors, relationship of balconies and windows, etc. We believe that it is important to evaluate methods in terms of a relative improvement over strong unary baselines. The system described in this paper can be considered as such a strong baseline. In order to facilitate a fair comparison of previous and future work, we release the code and predictions that have been used to obtain the reported results [‡].

Acknowledgements This work was partly carried out in IMAGINE, a joint research project between ENPC and CSTB, and partly supported by ANR-13-CORD-0003 and ECP.

References

[1] A. Cohen, A. G. Schwing, and M. Pollefeys. Efficient structured parsing of facades using dynamic programming. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 3206–3213. IEEE, 2014.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, volume 1, pages 886–893, 2005.

[3] P. Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

[4] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *British Machine Vision Conference*, volume 2, page 5, 2009.

[5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer vision*, 88(2):303–338, 2010.

[6] B. Frohlich, E. Rodner, and J. Denzler. A fast approach for pixelwise labeling of facade images. In *Pattern Recognition (ICPR), 20th International Conference on*, pages 3029–3032. IEEE, 2010.

[7] B. Fröhlich, E. Rodner, and J. Denzler. Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach. In *Asian Conference on Computer Vision (ACCV)*, pages 218–231. Springer, 2012.

[8] C. Gatta and F. Ciompi. Stacked sequential scale-space taylor context. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[9] S. Gould. DARWIN: A framework for machine learning and computer vision research and development. *Journal of Machine Learning Research*, 13:3533–3537, Dec 2012.

[10] F. Korč and W. Förstner. eTRIMS Image Database for interpreting images of man-made scenes. Technical Report TR-IGG-P-2009-01, April 2009.

[11] A. Martinovic, M. Mathias, J. Weissenberg, and L. Van Gool. A three-layered approach to facade parsing. In *European Conference on Computer Vision (ECCV)*, pages 416–429. Springer, 2012.

[12] A. Martinovic and L. Van Gool. Bayesian grammar learning for inverse procedural modeling. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 201–208, 2013.

[13] S. Nowozin. Optimal decisions from probabilistic models: the intersection-over-union case. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2014.

[14] R. Š. Radim Tyleček. Spatial pattern templates for recognition of objects with regular structure. In *Proc. German Conference on Pattern Recognition*, Saarbrücken, Germany, 2013.

[15] H. Riemenschneider, U. Krispel, W. Thaller, M. Donoser, S. Havemann, D. Fellner, and H. Bischof. Irregular lattices for complex shape grammar facade parsing. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1640–1647, 2012.

[16] N. Ripperda and C. Brenner. Reconstruction of façade structures using a formal grammar and RjMCMC. In *DAGM*, 2006.

[17] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer vision*, 77(1-3):157–173, 2008.

[18] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, pages 1–15. Springer, 2006.

[19] O. Teboul. Ecole centrale paris facades database, 2010.

[20] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 2273–2280. IEEE, 2011.

[21] Z. Tu. Auto-context and its application to high-level vision tasks. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1–8, 2008.

[22] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[23] M. Y. Yang and W. Forstner. A hierarchical conditional random field model for labeling and classifying images of man-made scenes. In *Computer Vision Workshops (ICCV Workshops), IEEE International Conference on*, pages 196–203, 2011.

[‡] `http://fs.vjresearch.com`