Semantic Segmentation of Highly Structured and Weakly Structured Images

Raghu Deep Gadde

École des Ponts
ParisTech

UNIVERSITÉ
PARIS-EST

Semantic Segmentation of Highly Structured and Weakly Structured Images

Raghudeep GADDE

Submitted to the **École Doctorale Paris-Est Mathématiques et Sciences et Technologies de l'Information et de la Communication**.

Presented on the 30th of June 2017 to a committee consisting of:

President	Iasonas KOKKINOS	-	University College London
Rapporteur	Isabelle BLOCH	-	Telecom-ParisTech
	Frédéric JURIE	-	University of Caen
Examinateur	Karteek ALAHARI	-	Inria
Directeur de these	Renaud MARLET	-	École des Ponts ParisTech
	Nikos PARAGIOS	-	CentraleSupélec

École des Ponts ParisTech
CERTIS/IMAGINE
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77455 Marne-la-Vallée cedex 2
France

Université Paris-Est Marne-la-Vallée
École Doctorale Paris-Est MSTIC
Département Études Doctorales
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77454 Marne-la-Vallé cedex 2
France

Acknowledgments

I would first like to thank my advisors, Renaud Marlet and Nikos Paragios for letting me join IMAGINE and guiding me through the process of writing a thesis. It was a wonderful learning experience. I am very grateful for all the support I received during this journey. They gave me complete freedom and trust to follow my own academic interests while at the same time offering close supervision when I needed it. I would also like to thank Peter Gehler who gave me an internship opportunity at the Perceiving Systems, Max Planck Institute for Intelligent Systems. This thesis is the result of multiple research collaborations, innumerable in-depth discussions and the general support of so many people. Many have inspired me during the last few years, and I want to thank them for it. I would like to thank all my co-authors with whom I have worked together with very closely for the papers: Varun Jampani, Mateusz Kozinski, Sergey Zagoruyko, Martin Kiefel, Daniel Kappler.

At this point I would also like to thank Frederic Jurie, Isabelle Bloch, Iasonas Kokkinos and Karteek Alahari for taking time out to review the thesis and being part of the thesis committee. Another big thank you goes to Brigitte Mondou for all her help in all administrative matters as well as always going out of her way to help in general and to Dr. Pascal Monasse for his prompt support with all kinds of computer problems that I had during my PhD.

I would like to acknowledge the generous funding CSTB and ANR Semapolis project (ANR-13-CORD-0003) without which this thesis would not have been possible.

I also feel lucky to have been surrounded by so many supportive, smart, hard working and extremely entertaining group of peers, making the highs and lows of the PhD journey so much fun. Finally, I am very grateful for all the support of my friends and family. A special thank you goes to all of my friends and colleagues including Francisco Vitor Suzano Massa, Mateusz Kozinski, Marina Vinyes, Laura Fernandez Julia, Maria Vakalopoulou, Sergey Zagoruyko, Spyros Gidaris, Shell Hu, Puneet Dokania, Amine Bourki, Praveer Singh, Martin Simonovsky, Yohann Salaun, Alexandre Boulch, Martin de La Gorce, Pierre Moulon, Thomas Nestmeyer, Christoph Lassner. They have been a sort of family through all these years and I have very fond memories that I will remember for the rest of my life. I miss you all.

I owe all I am today to the hard work and sacrifices made by my parents, and their unconditional love and support on all my personal and professional endeavors. Above all, I would like to thank Mounika who has walked with me all the time. It is incredible that you have never stopped believing in me. I am so excited to see all the other adventures to come!

Dedicated to my family - for their endless love, support and encouragement

Semantic Segmentation of Highly Structured and Weakly Structured Images

Abstract: Semantic Segmentation is one of the fundamental and important problems of computer vision. The aim of this thesis is to develop techniques for segmenting strongly-structured scenes (e.g. building images) and weakly-structured scenes (e.g. natural images). Building images can naturally be expressed in terms of grammars and inference is performed using grammars to obtain the optimal segmentation. However, it is difficult and time consuming to write such grammars. To alleviate this problem, a novel method to automatically learn grammars from a given training set of image and ground-truth segmentation pairs is developed. Experiments suggested that such learned grammars help in better and faster inference. Next, the effect of using grammars for strongly structured scenes is explored. To this end, a very simple technique based on Auto-Context is used to segment building images. Surprisingly, even with out using any domain specific knowledge, we observed significant improvements in terms of performance on several benchmark datasets. Lastly, a novel technique based on convolutional neural networks is developed to segment images without any high-level structure. Image-adaptive filtering is performed within a CNN architecture to facilitate long-range connections. Experiments on different large scale benchmarks show significant improvements in terms of performance.

Keywords: Semantic Segmentation, Grammars, Auto-Context, Bilateral Filtering.

Resume

Cette thèse pour but de développer des méthodes de segmentation pour des scènes fortement structurées (ex. bâtiments et environnements urbains) ou faiblement structurées (ex. paysages ou objets naturels). En particulier, les images de bâtiments peuvent être décrites en termes d'une grammaire de formes, et une dérivation de cette grammaire peut être inférée pour obtenir une segmentation d'une image. Cependant, il est difficile et long d'écrire de telles grammaires. Pour répondre à ce problème, nous avons développé une nouvelle méthode qui permet d'apprendre automatiquement une grammaire à partir d'un ensemble d'images et de leur segmentation associée. Des expériences montrent que des grammaires ainsi apprises permettent une inférence plus rapide et produisent de meilleures segmentations. Nous avons également étudié une méthode basée sur les auto-contextes pour segmenter des scènes fortement structurées et notamment des images de bâtiments. De manière surprenante, même sans connaissance spécifique sur le type de scène particulier observé, nous obtenons des gains significatifs en qualité de segmentation sur plusieurs jeux de données. Enfin, nous avons développé une technique basée sur les réseaux de neurones convolutifs (CNN) pour segmenter des images de scènes faiblement structurées. Un filtrage adaptatif est effectué à l'intérieur même du réseau pour permettre des dépendances entre zones d'images distantes. Des expériences sur plusieurs jeux de données à grande échelle montrent là aussi un gain important sur la qualité de segmentation.

Contents

CHAPTER 1

Introduction

Understanding a given image, i.e, answering *what is where in a given image,* is one of the core problems of computer vision. The task here, called *semantic image segmentation* is to semantically understand each pixel of a given image, i.e, to know what object/region the pixel contributes in representing. More formally, for each pixel in a image, we have to assign the most appropriate and semantically meaningful label from a given set of labels. Some of the practical applications of semantic segmentation can be found in autonomous driving, robot grasping, image retargeting, aiding blind people, etc.

Figure 1.1: Sample natural and urban scenes with their corresponding human annotations from the PASCAL VOC2012 [Everingham 2010] (left two images), LabelMeFacade [Fröhlich 2012], eTRIMS [Korc 2009] datasets.

Semantics is essential and fundamental to understand both natural and urban/man-made scenes. Example images from different datasets are shown in Figure 1.1. Typically there are three steps in a standard semantic segmentation framework, namely (1) feature extraction, (2) classification, and (3) smoothing/regularization (see Figure 1.2). In the feature extraction stage, popular descriptors like SIFT [Lowe 2004], SURF [Bay 2006], HOG [Dalal 2005], etc., are extracted based on raw input image. In the classification stage, the extracted features are fed to classifiers like SVM [Cortes 1995] and Random Forests [Breiman 2001] to obtain predictions. Finally, the obtained predictions are smoothed using contextual information via graphical models. There have been numerous works in each stage of the framework. More complex multi-scale, region-based, local & global descriptors were proposed to better describe the given image. Boosted classifiers were in-

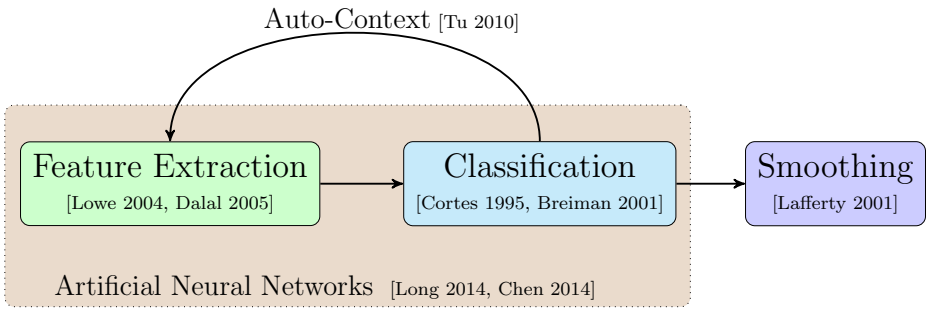


Figure 1.2: Standard semantic segmentation framework

troduced to improve the classification performance. Hierarchical, higher order and fully connected graphical models were proposed to better encode contextual information. While exact inference is intractable, several efficient and fast approximate techniques were introduced. Another type of graphical models use high-level prior information in terms of grammars, for example, to parse a given image. [Chen 2007, Zhu 2009, Tu 2005, Han 2009, Teboul 2011c] have explored top-down and bottom-up parsing to find the optimal labelling according to a given grammar.

Another line of research to encode contextual information is through the auto-context model [Tu 2008, Tu 2010]. Given a training set of labeled data, a classifier is learned using image features. The obtained predictions are then used as contextual information by appending them to the original image features to learn a new classifier and the algorithm continues iteratively until convergence. While feature extraction and classification are two seperate steps, artificial neural networks provide a common framework by learning hierarchical feature representations and classifying them. Recently due to availability of large datasets such as PASCAL VOC2012, Cityscapes and progress in the computational hardware industry, deep neural networks [Long 2014, Chen 2014a] showed significant improvements with low runtimes (typically few hundred milliseconds) over traditional methods. Deep neural networks combined with graphical models [Chen 2014a] as post-processing step are the current state-of-the-art techniques for semantic segmentation while some very recent techniques [Zheng 2015, Gadde 2015] combine deep neural networks and graphical models into a unified framework showing further improvement.

1.1 Aim of the thesis

The current thesis aims to address semantic segmentation from diverse perspectives.

Grammar-based: Man-made or urban scenes tend to follow certain layouts that can be regular, e.g., with respect to symmetry or repetition. Such patterns can often be expressed as prior knowledge in the form a shape grammar. Images are parsed with models encoding such prior knowledge to obtain the segmentation result. However it is hard to design such grammars manually and it requires expert knowledge. See Figure 1.3 for an example of

hand-written grammar and parsed result. To overcome the need for hand-written prior, we learn grammars automatically from given training data. To this end, we use the problem of segmenting building images as test bed for our approach. We show that the learned grammar (which typically represents an architecture style as we are dealing with building images) is compact in representation and helps in faster inference. Generally, the objective of a grammar-based or a prior-based method is to understand the underlying high-level structure/architecture of a given image. Learned grammars capture such information from the given training set. Such learned grammars can be used for a variety of other applications such as structure based retrieval, procedural modelling, robust 3D reconstruction.

Domain-independent: While domain-dependent approaches are common for parsing man-made scenes [Teboul 2010b,Teboul 2011c,Simon 2011,Teboul 2013,Martinovic 2012, Martinovic 2013a,Martinovic 2015] we follow a contrasting approach by investigating the contribution of domain knowledge for their success. For this we again use the problem of segmenting building images as a test bed. Interestingly we find that using a simple auto-context based [Tu 2010] model performs comparably or even better on several benchmarks for facade segmentation. We have also investigated whether this applies to segmenting 3D point clouds as well and found a similar performance.

Image adaptive filtering based: Very recently, deep convolutional neural networks have significantly improved the state of the art and drastically reduced the runtime to the order of few hundreds of milliseconds. However, the filters in standard CNNs have a fixed receptive field and interaction between far away pixels is not possible. To overcome this limitation we explore the usage of image-adaptive filters which facilitate long-range pixel interactions within a standard CNN. Using image adaptive filters also help in performing structured prediction that is trainable in an end-to-end setting, thereby combining all three steps of a standard semantic segmentation framework into a unified model.

1.2 Contributions and organization of the thesis

The thesis proposes new techniques for semantic segmentation of highly structured building images and weakly-structured natural scene images. First we propose an approach to learn high-level grammars from training data which we use to segment building images. Second, we investigate the need for domain-dependent approaches for segmenting 2D images and 3D point clouds of buildings and found that domain-independent and generic approaches perform comparably or even better for segmenting building images and point-clouds. Last, we exploit the structural information in a global fashion by allowing and encouraging connections between far away pixels and improve the state-of-the-art segmentation techniques for natural scenes. The chapter-wise organisation of the thesis is as follows

- **Chapter 2.** In this chapter, we introduce and provide sufficient background on hand-written priors and inference techniques for segmenting building images. This chapter

presents the first contribution of the thesis, a novel approach to learn architecture specific priors for segmenting building images.

- **Chapter 3.** The second contribution of the thesis, a novel technique that learns structural information from facade images using auto-context features is described in this chapter.

- **Chapter 4.** The third and final contribution of the current work, a technique to improve semantic segmentation of weakly-structured images by performing bilateral filtering which allow global connections between pixels is described in Chapter 4.

- **Chapter 5.** Finally, we conclude and provide possible extensions to this thesis.

1.2.1 Publications

The work in this thesis lead to the following publications and articles.

- Gadde, R., Marlet, R. and Paragios, N. Learning grammars for architecture-specific facade parsing. In 2016 International Journal of Computer Vision, 117(3), pp.290-316.

- Jampani, V., Gadde, R. and Gehler, P.V. Efficient facade segmentation using auto-context. In 2015 IEEE Winter Conference on Applications of Computer Vision (pp. 1038-1045).

- Gadde, R., Jampani, V., Kiefel, M., Kappler, D. and Gehler, P.V. Superpixel convolutional networks using bilateral inceptions. In 2016 European Conference on Computer Vision.

- Gadde, R., Jampani, V., Marlet, R. and Gehler, P.V. Efficient 2D and 3D facade segmentation using auto-context. In 2017 IEEE Transactions on Pattern Analysis and Machine Intelligence.

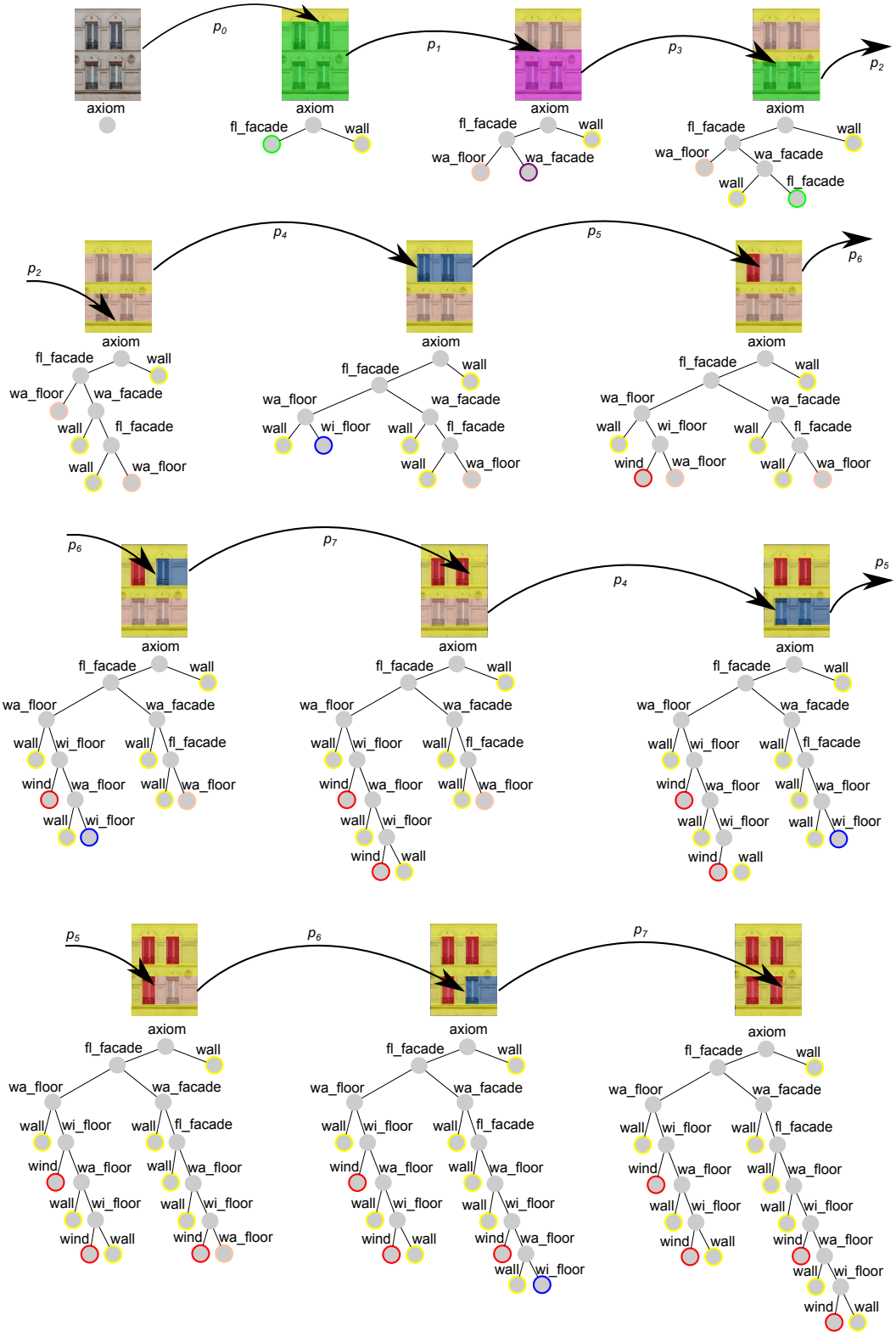


Figure 1.3: Parse tree derivation of an example image with a given grammar. Taken from Matuesz Kozinski's PhD thesis [Koziński 2015a].

CHAPTER 2

Learning Grammars for Architecture-Specific Facade Parsing

Parsing facade images requires grammars that are specific to a given class of buildings. Such grammars are typically designed manually by experts. In this chapter, we present a novel framework to learn a compact grammar from a set of images with ground-truth information.

2.1 Introduction

How building facades are segmented is of great interest in computer vision due to the number of applications and associated issues. Knowing the regularities in facade layout can be used in video games and movies to generate plausible urban landscapes with realistic rendering [Müller 2006]. It can also guide the analysis of building images to construct semantized models that can be used for urban planning and in simulation tasks (e.g., for thermal performance evaluation or shadow casting studies) as well as to compact data for virtual navigation in cities.

Existing approaches for facade analysis, i.e., the segmentation of facade images into semantic classes, use either conventional segmentation methods [Cohen 2014a, Dai 2012, Martinovic 2012] or rely on grammar-driven recognition methods [Teboul 2011b, Riemenschneider 2012, Martinovic 2013a]. Conventional segmentation methods treat the problem as a pixel labeling task, with the possible addition of local regularity constraints related to building elements, but ignoring the global structural information in the architecture. On the contrary, methods based on shape grammars impose strong structural consistencies by considering only segments that follow a hierarchical decomposition corresponding to a combination of grammar rules. However, these methods require carefully handcrafted grammars to reach good performance. Besides, as many grammars as different architecture styles are required, and it is not clear who will write and finely tune them, with what expertise and at which cost, when there exists so many building styles.

In this work, we focus on structural segmentation, i.e., with global regularities and hard constraints as opposed to just local pixel labeling. Our final goal is thus not to produce a state-of-the-art pixelwise classification but to provide a state-of-the-art, high-level, structured view of pictured objects. More precisely, we propose a method to automatically learn grammars from annotated images, which we illustrate on facade analysis. The grammars we learn are specific to the architecture style of the training samples. Using these

grammars, we reach state-of-the-art parsing results, competing with handcrafted grammars. Thanks to our method, the tedious grammar writing and tuning task is turned into the much simpler and basic task of annotating facade images.

2.1.1 Related Work

Conventional segmentation techniques rely on grouping together consistent visual characteristics while imposing piecewise smoothness. Popular methods are based on active contours [Osher 2003, Kass 1988], clustering techniques such as mean-shift [Comaniciu 2002] and SLIC [Achanta 2012], and graph cuts [Berg 2007, Kolmogorov 2004].

However, although they obtain very good pixelwise scores, these techniques are not appropriate for a number of applications because they frequently produce segments that are inconsistent with basic architectural rules, e.g., irregular window sizes or alignments, or balconies shifted from associated windows. While it may be enough, e.g., to get a rough estimate of the percentage of glass area for thermal performance evaluation, it is totally inappropriate to generate building models (BIM), with both geometric and semantic information, as used in the construction and renovation industry. Moreover, as they label only what is visible, ordinary segmentation methods are sensitive to occlusions, e.g., due to potted plants on windows and balconies, or to pervasive foreground objects in the street: trees, vehicles, pedestrians, street signs, lampposts, etc. As a result, important elements can be partially or totally missing from the produced segments, e.g., portions of wall or even complete windows. On the contrary, grammar-based methods can infer invisible or hardly visible objects thanks to architecture-level regularity. Conventional segmentation methods may also be sensitive to variations of illumination such as cast shadows, night lighting and glass reflection, although the sensitivity can be partly reduced with larger training sets. Here again, grammar-based priors arguably provide better segmentation in case of "illumination noise" thanks to more global constraints. Actually, grammar-based image parsing methods should not be thought of as alternative segmentation methods but as approaches that take a good pixel classification (a.k.a. unaries) as input and that further impose strong architectural constraints as high-level regularizers. The two kinds of approaches are thus complementary: a better low-level classification or segmentation naturally leads to a better parsing and better overall accuracy (assuming the observed facade follows the architecture style modeled in the grammar).

More accurate segmentations have been obtained adding weak architectural constraints, that are either hard-coded [Martinovic 2012] or learned [Dai 2012], yielding improved pixel classifications, but still breaking fundamental architectural rules such as window alignments or balcony-window relationships. Extra structural constraints have been hard-coded into several dynamic programming problems that can be solved efficiently and accurately, again improving the state of the art [Cohen 2014a]. However, some structural rules are still not expressed in this approach, such as the vertical alignment of windows, which is a common constraint. It also is difficult to adapt to new structures and new architectural styles because the regularity is defined by hand, problem by problem.

On the contrary, segmentation methods based on shape grammars [Teboul 2011b, Riemenschneider 2012, Martinovic 2013a, Alegre 2004, Ripperda 2006a, Koutsourakis 2009, Si-

mon 2011, Simon 2012] make the constraints explicit and thus facilitate the parameterization and adaptation to new architecture styles. They impose strong structural consistencies by considering only segments that follow a hierarchical decomposition corresponding to a combination of the rules defined in the grammar. Analyzing an image consists here in producing a parse tree whose associated segments fit as well as possible with the observation. Mixed continuous-discrete inference is generally used to produce good parse trees. The inference of the structure of segments can also be separated from the optimization of their size and positions [Koziński 2014a], or be completely integrated into constraints not requiring inefficient rule sampling [Koziński 2014b]. With this kind of methods, partially or fully occluded scene elements such as wall and windows can be recovered thanks to structural consistency. These methods are also less sensitive to changes of illumination. However, one of their most important limitation is the dependency on the grammar design, that is generally written and tuned manually. It is thus natural to try to learn these grammars automatically.

Although grammatical inference is common in natural language processing (NLP), it is rare in computer vision. Recently, a couple of methods have been proposed to automatically learn shape grammars from ground-truth image annotations [Martinovic 2013a, Weissenberg 2013]. To the best of our knowledge, these two methods are the only ones that can tackle the complexity of multi-class facade segmentation over a substantial training set. Both operate on split grammars. Split grammars, in 2D, feature grammar rules where a rectangle image is recursively split vertically or horizontally into subrectangles. We detail both approaches.

Martinovic and Van Gool's method [Martinovic 2013a] does not operate directly on the image but on an irregular lattice space similar to the one used by Riemenschneider et al. [Riemenschneider 2012] for parsing. For each example in the training set, a specific split grammar is constructed based on the lattice representation, alternating horizontal and vertical split rules. Putting together all rules of all examples yields a large grammar describing exactly the training set. These rules are then merged iteratively by a generalization operation, following a Bayesian model-merging technique. Each step of this iteration is relatively expensive because it requires considering as merging candidates all pairs of non-terminals and evaluating the corresponding grammar. After iterating, the resulting merged grammar is both smaller, which leads to faster parsing, and more general, to treat examples that are not in the training set. It seems however this approach does not scale well as the authors have to reduce the size of the training set to keep the induction time practicable.

Weissenberg et al. [Weissenberg 2013] present an alternative technique to learn split grammars from images with ground-truth annotations. As in Martinovic and Van Gool's method, a parse tree is first constructed for each annotated image in the training set. However, the construction here operates directly in the image space, generating split rules iteratively based on an energy function expressing preference among split line candidates. Nested binary split rules in the same direction are then grouped together to form n-ary split rules. Finally, a compact grammar is generated by greedily merging grammar rules with identical structure (split direction and sub-components) but different parameters (split positions). The work is validated by a study of the performance of grammar compression,

an experiment in facade image retrieval and examples of virtual facade synthesis. But no experiment on using the generated grammars for parsing is reported.

Tu et al. [Tu 2013] propose a powerful and very general framework for the unsupervised learning of stochastic And-Or grammars which, like ours, is also based on some kind of factorization of similar subtrees. But it is not clear how this approach could be applied to the segmentation of facade images. In this framework, when applied to images, terminals are visual words that are to be connected via spatial relations and structured into a compact hierarchy of nonterminals. This hierarchy is inferred from the distribution of terminals in the training set, maximizing the posterior probability of the corresponding grammar. To apply this generic method, a specific work is required to select appropriate visual words and define relevant spatial relations that can carry across the factorization process. Besides, the learning process starts from a flat representation of all visual words in each image of the training set, along with their relations, whose number can grow quadratically with the number of visual words, and there is no indication in the general framework on a strategy for dropping or merging relations when performing generalization. In fact, examples in [Tu 2013] are only illustrated on objects with a small and fixed number of components that have well-defined relative positions (well centered animal faces with two ears, two eyes and one nose, among four species of mammals), which is quite different from the case of facades with an unknown number of floors and an unknown number of window columns, and where objects can cover a wide portion of the image area (whole extent of wall, roof, sky, running balconies).

Si and Zhu [Si 2013] have a similar approach to learn And-Or grammars. Rather than relying on specific and explicit relations between terminals, it is based on the direct encoding of object presence and position in an occupancy grid. However, the size of this encoding grows with the grid resolution (quadratically in the length of objects), which may raise scalability issues. As a matter of fact, it seems that experiments have been reported up to a 19x19 grid only, which is too coarse for the level of accuracy we target (about 70 to 90 % of pixel accuracy for images of size at least 0.2 Mpixels). Besides, in the case of facade images, similar windows that are just shifted a few squares horizontally or vertically would have a very different representation, leading either to an explosion of alternative cases if they are kept separate (large Or-nodes, i.e., overfitting), or to an excessive generalization if they are merged (large And-nodes containing small Or-nodes, i.e., independent probabilities for neighboring squares). On the contrary, split grammars separate presence (given by rules) and position (given by rule parameters), which greatly reduces the space of configurations to explore and allows an independent factorization of rules and parameters.

It seems that these approaches, based on And-Or grammars and visual words, are more suited for classification and detection tasks (as illustrated by presently reported experiments) than for accurate segmentation. To our knowledge, no experiment with these grammar learning methods has been reported on facade segmentation tasks, at least not on the standard datasets used to evaluate and compare competing methods.

Another interesting aspect of these two approaches, at least theoretically, is the use of stochastic grammars. We actually made experiments of facade parsing with the addition of probabilities to split grammar rules. As it resulted in a minor accuracy improvement, we choose not to burden our grammar learning method with probabilities, for such a small

margin. It is seems that fixed rule probabilities are less relevant as guides to explore the space of configurations (rule combinations) than the bottom-up cues specific to a given image [Ok 2012].

Grammar induction has been studied both in the formal language literature [De La Higuera 2005] (with applications, e.g., to pattern recognition and RNA structure modeling) and in the NLP community [D'Ulizia 2011]. The formal language literature mainly considers learning from strings based on positive examples, possibly complemented by negative data [De la Higuera 2010], whereas the NLP community focuses on learning distribution information from hand-annotated parse trees representing positive examples. As for the parsing images, where pixels are (at least) 4-connected, the 2D nature of the problem makes inappropriate most approaches based on learning from strings, as their working principle heavily relies on the 1D associativity of the binary concatenation operator [Nevill-Manning 1997, Sakakibara 1999, Clark 2010]. Learning sets for image parsing typically also consist of positive examples only. As a result, the most relevant literature concerning shape grammar learning lies in the NLP community. (Other approaches such as statistical relational learning and inductive logic programming that have some connections to grammar learning, but currently no obvious links to shape grammars.)

Learning from trees is also a way to escape some of the two-dimensional parsing issues. Parsing 2D data [Tomita 1991, Martinović 2013b] indeed has a much higher complexity than 1D parsing. The orders of magnitude also differ widely: an average English sentence, with about 21 words, whose part of speech (POS) can be determined with an accuracy of 97.3%, has a general accuracy of 56% [Manning 2011]; a small image with only 300,000 pixels, whose pixel accuracy is at best 92% [Jampani 2015], has an overall accuracy less than $10^{-10,000}$. Considering the noise in input data, image parsing actually is closer to speech processing than to plain text parsing. This situation probably explains why a number of proposed algorithms for image parsing consist of a partial, randomized exploration of an extremely large space, corresponding to derivation trees generated in a top-down manner [Teboul 2011b, Simon 2011, Simon 2012].

Now if the choices for splitting a region vertically or horizontally are already made in the trees of the training set, the grammar induction problem then becomes related to the problem of learning a tree automaton from tree-structured data [Carrasco 2001]. Indeed, previous approaches for shape grammar learning involve a first stage of tree hypothesis generation to produce ground-truth parse trees from the ground-truth segmentation, based on heuristics [Martinovic 2013a, Weissenberg 2013]; it is similar to the case of unsupervised data-oriented parsing [Bod 2006], that considers a subset of all possible binary trees that can be constructed over training strings. In our approach, we propose to generate these ground-truth parse trees differently, using a small generic handwritten grammar, which provides more similar trees in which patterns can be found, as well as interpretable parses (in terms of the generic grammar).

Two simple but useless solutions to grammatical inference are either to construct a flat grammar generating only the examples in the training set (one rule per training sample) or to construct a grammar that considers all strings or structures as parse-able. To prevent these trivial solutions and find a right balance between these two extreme cases, the grammar to infer is typically required to have a certain level of generality, thus allowing to also

parse unseen sentences or structures, but not too much not to over-generalize. This can be achieved by introducing mechanisms of rule inference that can generalize patterns in the training set, together with a compactness criterion such as a minimum message length (MML) or minimum description length (MDL) principle [Grünwald 1996].

In NLP, parsing can be ambiguous due uncertainties when determining the part of speech of words, and also because of possible spelling errors and unknown words. For this reason, statistical information is also learned from training data for the parser to produce most likely sentence analyses. The nature of this information is however strongly related to the nature of the targeted parser and grammar, e.g., whether it is statistical data for a probabilistic context-free grammar (PCFG) [Johnson 2007], a latent-variable PCFG (L-PCFG) [Cohen 2014c], or a data-driven dependency parser [Nivre 2007]. The same situation occurs for shape grammar learning. In our case, as we target Teboul et al.'s parser [Teboul 2011b], which does not exploit any data distribution knowledge when sampling production rules, probabilistic information makes little sense. This is consistent with the fact that, for practical shape grammars, the parser at any point only has a few structural choices, i.e., a small number of applicable rules if the split position parameters are ignored. Besides, even if many split positions are possible for the same "meta-rule" according to the grammar, position sampling actually depends on bottom-up cues extracted from the parsed image. What matters most is thus the occurrence or not of certain structural patterns and rule parameters in training data, not their frequency.

The work in NLP that is most closely related to our approach is grammar refinement, which operate on annotated trees to learn distribution information but also to generate specialized rules to represent patterns that could not be captured given strong independence assumptions of grammar rules. This may be achieved with symbol splitting [Petrov 2007], latent variable addition [Matsuzaki 2005], or grammar paradigms richer than plain context-free grammars (CFGs), such as tree substitution grammars (TSGs) [Bod 2003]. TSGs allows for arbitrarily large tree fragments as rules in the grammar and thereby better represent complex structures. The TSG induction scheme proposed by Cohn et al. [Cohn 2010] relies on a Bayesian non-parametric prior for regularizing the tree fragments to explore as rule candidates, giving a bias towards small grammars with small production rules. This method is different from our approach, where we find repeating subtrees in the data and then perform clustering of these complex subtrees. In our implementation, as our target parser only accepts plain binary split grammars (BSGs) [Teboul 2011b], we actually represent complex rules using a flat deterministic decomposition which is similar to what occurs when symbol splitting is performed [Petrov 2007]. The inference of latent variables to construct combined instances of specialized rules seems to be a promising alternative to the rule clustering algorithm that we propose, especially spectral methods [Cohen 2014c] that appear to scale well (more or less linearly) when the number of hidden states increases [Cohen 2013], compared to EM-based methods [Matsuzaki 2005], for a similar if not better accuracy. The order of magnitude of the reported number of hidden states (up to 32) [Cohen 2013] is comparable to the number of rule instances we generate. The level of recursion in natural language sentences is however much lower than in the kind of shapes and grammars we consider.

Another aspect is the difference of size of the training corpora. In NLP, the training

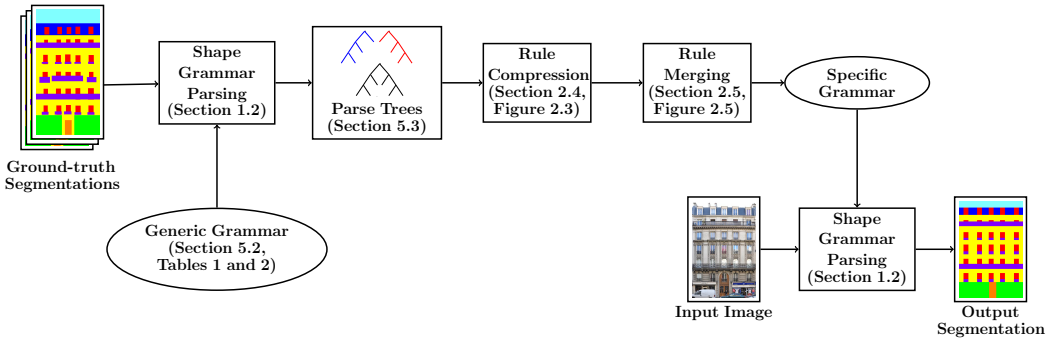


Figure 2.1: Overall pipeline of the framework.

sets used for grammar induction, such as the WSJ section of the Penn treebank, typically contain more than 30,000 trees (i.e., sentences). Although language constructs are arguably more complex than shape relationships, and thus require more training data, this is at least two orders of magnitude larger than the training sets used here for shape grammar learning, where the number of ground-truth segmentations for learning in our experiments is 40 to 300. This calls for different compromises.

The problem of grammatical inference is also studied in the data compression literature. The goal here is to find the smallest grammar that can generate a given string [Lehman 2002, Charikar 2002, Charikar 2005, Benz 2013]. However, as the information in this case is of symbolic nature (as opposed, e.g., to signals), compression is generally defined to be lossless. The grammar is thus designed to generate one and only one string. While some of these techniques can be accommodated to generate a given set of strings, they are not suited for generalization: the grammar is designed to reject any unknown strings, even if it is "similar" to a learned string. These techniques are thus not adapted to our problem, as we need to moderately generalize from the learning set.

2.1.2 Overview

Our method for automatically learning grammars from images with ground-truth annotations operates on split grammars. As the above two methods [Martinovic 2013a, Weissenberg 2013], it first generates a large set of rules from the training set, and then compresses and generalizes them. However, it is based on different principles and relies on more powerful grammatical transformations.

A graphical overview of our approach is pictured in Figure 2.1. We first consider a small, simple-to-write and generic grammar that can describe many kinds of segmentations but that is not constrained enough to impose actual structural regularities. Using these generic rules and a standard parser for split grammars, we parse the training image annotations. It generates a set of parse trees that fit, almost perfectly, the ground-truth annotations and that can thus be considered as ground-truth parse trees. The instantiated grammar rules occurring in these parse trees are representative of the architecture style of the training sample. However, this set of instantiated rules cannot practically be used as a

grammar within a parser because there are too many of them. Indeed, given the enormous combinatorial space to explore, current parsers require a moderate number of rules for inference to succeed. For these reasons, we perform two compression operations. First, we search for common subtrees in the parse trees and merge them into single rules. Second, we cluster rules using an appropriate similarity measure and factor each cluster around a single complex rule. This results in compact grammars that facilitate inference and generalize well the training samples.

In contrast with Weissenberg et al.'s method [Weissenberg 2013], our learned grammars can be used for efficient parsing. Our learned grammars reach the performance of handcrafted grammars in terms of accuracy of resulting segmentation with better parsing time. On the Haussmannian dataset [Teboul 2011b], it also outperforms the grammar generated by Martinovic and Van Gool's method [Martinovic 2013a] (for a different parser). Besides, our approach addresses the scalability issue of their method.

2.1.3 Contributions and Organization

The main contributions of our work are the following.

- We propose a new way to generate ground-truth parse trees based on simple, handwritten, generic grammars. Compared to current approaches, it provides less arbitrary and more systematic structures, from which patterns can better emerge, and that can be understood by a human.

- Contrary to other methods [Weissenberg 2013], the complex rule we generate may combine both horizontal and vertical splits, which captures richer patterns.

- Our rule generalization approach does not rely on a greedy iterative procedure, as in other methods. It is formulated as an unsupervised clustering problem, which is solved efficiently.

- Compared to the other approach for learning grammars that has been used for parsing, our method scales to training sets with several hundreds of annotated images.

- We provide and discuss experiments on four datasets featuring different architectures styles, including a new Art-deco dataset that we made available to the community. The other datasets are standard and well-known for evaluating facade segmentation. We show that our learned grammars have an equal or better performance than handcrafted grammars or other automatically generated grammars, almost reaching the state-of-the-art of hard-coded segmenters (that do not enforce all the hard architectural constraints that our generated grammars guarantee).

The rest of this chapter is organized as follows. Section 2.4 discusses the kind of grammars we want to learn. Section 2.5 presents a method to construct ground-truth parse trees and explains why these ground-truth parse-tree rules cannot be used directly as a learned grammar. Section 2.6 details how rules extracted from the ground-truth parse trees can be efficiently compressed. Section 2.7 explains how to merge rules, which both generalizes

and further compress them. Various experiments following this approach are reported and analyzed in Section 2.8. Section 2.9 concludes the chapter.

2.2 Split grammars in 2D

Split grammars were introduced by Wonka et al. [Wonka 2003] as a particular kind of shape grammars. The general idea of split grammars is to express the regularity of an object as a recursive decomposition where, at each level, a basic shape of a certain type is split into separate spatial regions that contain smaller basic shapes of some other types. A special case of split grammars in 2D considers a labeled image rectangle as the basic shape. This labeled rectangle is recursively split horizontally or vertically into labeled sub-rectangles according to rules of a grammar.

2.2.1 The grammatical formalism

More formally, a labeled rectangle of an image is denoted as $l(x, y, w, h)$, where l is the label of the rectangle, x, y are its coordinates and w, h are its width and height. A 2D binary split grammar (2D-BSG, or BSG for short) is a 4-tuple $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$ where $\mathcal{N}$ is a set of non-terminal symbols, $\mathcal{T}$ is a set of terminal symbols (disjoint from $\mathcal{N}$), $\mathcal{R}$ is a set of production rules, and $S \in \mathcal{N}$ is a start symbol, also called axiom. A *simple rule* in $\mathcal{R}$ has one of the two following forms:

$$A \quad \rightarrow \quad B \tag{2.1}$$

$$A \quad \overset{a}{\rightarrow}_p \quad BC \tag{2.2}$$

The left-hand side of the arrow is a single non-terminal $A \in \mathcal{N}$. The right-hand side consists of terminals or non-terminals $B, C \in \mathcal{N} \cup \mathcal{T}$. On the arrow, $a \in \{\mathsf{h}, \mathsf{v}\}$ indicates a split axis and $p > 0$ is the split position. The first kind of production rule (2.1) expresses a mere change of label of the rectangle. The second kind (2.2) expresses the fact that a rectangle $A(x, y, w, h)$ can be split along axis a at position p into two sub-rectangles of type B and C. The effect of the above rules on a rectangle scope (x, y, w, h) is as follows:

$$A(x, y, w, h) \quad \rightarrow \quad B(x, y, w, h) \tag{2.3}$$

$$A(x, y, w, h) \quad \overset{\mathsf{h}}{\rightarrow}_p \quad B(x, y, p, h)\, C(x + p, y, w - p, h) \tag{2.4}$$

$$A(x, y, w, h) \quad \overset{\mathsf{v}}{\rightarrow}_p \quad B(x, y, w, p)\, C(x, y + p, w, h - p) \tag{2.5}$$

If $a = \mathsf{h}$, the rectangle is split horizontally (with a vertical split line), which creates two adjacent sub-rectangles of the same height. If $a = \mathsf{v}$, the split is vertical (with a horizontal split line), which creates two sub-rectangles of the same width one on top of another. A rule of form (2.2) is only applicable if $p < h$ when $a = \mathsf{h}$, or $p < w$ when $a = \mathsf{v}$; it then creates two proper sub-rectangles (not with null height or width).

Terminals represent atomic elements, i.e., rectangles that contain only one type of object, e.g., a window, or part of a wall. By definition, a terminal never occurs on the left-hand side of a production rule. Non-terminals represent complex elements that can be broken

down recursively into simpler elements until all of them are terminals, e.g., the floor of a building, which can be broken down into wall parts, windows and balconies.

The formalism of split grammars can be enriched with notations that facilitate the writing of grammars. Standard notations includes:

$$A_0 \xrightarrow{a}_{(p_1,\ldots,p_{n-1})} M_1 \ldots M_n \Leftrightarrow \begin{cases} A_0 \xrightarrow{a}_{p_1} M_1 X_1 \\ X_1 \xrightarrow{a}_{p_2} M_2 X_2 \\ \ldots \\ X_{n-2} \xrightarrow{a}_{p_{n-1}} M_{n-1} M_n \end{cases} \tag{2.6}$$

$$A_0 \xrightarrow{a} \ldots (M) \ldots \Leftrightarrow \begin{cases} A_0 \xrightarrow{a} \ldots X \ldots \\ X \xrightarrow{a} M \end{cases} \tag{2.7}$$

$$A_0 \xrightarrow{a} M_1 \,|\, \ldots \,|\, M_n \Leftrightarrow \begin{cases} A_0 \xrightarrow{a} M_1 \\ \ldots \\ A_0 \xrightarrow{a} M_n \end{cases} \tag{2.8}$$

$$A_0 \xrightarrow{a} \ldots M+ \ldots \Leftrightarrow \begin{cases} A_0 \xrightarrow{a} \ldots X \ldots \\ X \xrightarrow{a} M \\ X \xrightarrow{a} MX \end{cases} \tag{2.9}$$

$$A_0 \xrightarrow{a} \ldots M? \ldots \Leftrightarrow \begin{cases} A_0 \xrightarrow{a} \ldots \ldots \\ A_0 \xrightarrow{a} \ldots M \ldots \end{cases} \tag{2.10}$$

$$A_0 \xrightarrow{a}_{\{p_1,\ldots,p_m\}} M \Leftrightarrow \begin{cases} A_0 \xrightarrow{a}_{p_1} M \\ \ldots \\ A_0 \xrightarrow{a}_{p_m} M \end{cases} \tag{2.11}$$

where X, X_1, etc., are extra auxiliary non-terminals and where M, M_1, etc., is a concatenation of expressions built on non-terminal and terminal symbols. Note that these are only abbreviations, not a change of paradigm. In particular, a *parameterized rule* $A \xrightarrow{a}_P BC$ is just a factorization of the *meta-rule* $A \xrightarrow{a} BC$ for all the parameters $p \in P$ of *instantiated rules* $A \xrightarrow{a}_p BC$. Tables 2.1 and 2.2 provide examples of grammars in this formalism.

Split grammars can also be seen as tree substitution grammars (TSGs) by making explicit the split as a tree operator, with given split axis and position. Simple rewriting rule of the form $A \to B$ stay the same. Other kinds of rule are understood as follows:

$$A \xrightarrow{a}_p BC \qquad \Leftrightarrow \qquad A \to \mathsf{split}_{a,p}(B, C) \tag{2.12}$$

This allows the definition of *complex rules*, whose right-hand side is a tree with operators $\mathsf{split}_{a,p}$ as non-leaf nodes, and terminals or non-terminals as leafs, e.g.,

$$\mathsf{split}_{a_1,p_1}(\mathsf{split}_{a_2,p_2}(A_1,A_2),\mathsf{split}_{a_3,p_3}(A_3,\mathsf{split}_{a_4,p_4}(A_4,A_5))) \tag{2.13}$$

Note that in a right-hand side tree, the axes $a_1, \ldots, a_n$ of the split nodes are not required to be equal.

Simple generic grammar $\mathcal{G}_{\mathsf{sgen}}$		
Axiom	$\xrightarrow{\mathsf{v}}$	GroundFloor Floors RoofFloor sky
GroundFloor	$\xrightarrow{\mathsf{h}}$	shop door shop
Floors	$\xrightarrow{\mathsf{v}}$	wall (Floor wall)+
Floor	$\xrightarrow{\mathsf{h}}$	wall (BalcWins wall)+
Floor	$\xrightarrow{\mathsf{v}}$	balcony WinFloor
WinFloor	$\xrightarrow{\mathsf{h}}$	wall (windows wall)+
BalcWin	$\xrightarrow{\mathsf{v}}$	balcony window
RoofFloor	$\xrightarrow{\mathsf{v}}$	roof (window roof)+

Table 2.1: The meta-rules of a simple generic grammar that has the same structural expressive power as Teboul et al.'s Haussmannian grammar [Teboul 2011b].

This grammar formalism (BSG, or TSG with split nodes) defines context-free shapes: a non-terminal is transformed according to the grammar independently of its context. As such, this formalism cannot capture grid regularities, e.g., to model the alignment of windows both horizontally and vertically. For this reason, *repetition tags* were introduced by [Teboul 2011a], which can be put on any non-terminal of the grammar. This tag indicates that all derivations of a non-terminal (see Sect. 2.2.2) shall be identical w.r.t. its split direction and split position. This variant of the usual binary split grammars allows the expression of grid-like constraints. For instance, if the non-terminal for floors is tagged, all floors will have identical window splits, which will ensure that all windows are vertically aligned. This tag extends the BSG formalism to non context-free grammars (probably to something akin to Type-1 grammars in the Chomsky hierarchy, or possibly a subset).

2.2.2 Derivation trees

A derivation is a top-down view of the decomposition of an object via the grammar. It represents the process and result of recursively splitting a non-terminal into terminal elements. Unless otherwise specified a derivation originates from the start symbol S. Note that, in practice, a grammar generally contains several rules that have the same non-terminals A at the left-hand side of a rule. It introduces non-determinism as different rules can then be applied to split a given rectangle of type A.

More formally, given some rectangular image of size $W \times H$, the basic shape $S(0, 0, W, H)$ is recursively transformed or split into sub-rectangles as defined by production rules in the grammar. At any point of this process, the input image is tiled into rectangles that have a label in $\mathcal{T} \cup \mathcal{N}$, which provides a semantic interpretation in terms of labeled segments. In theory, this process may not terminate because of possible recursive rules; in practice, binary rules reduce the size of rectangles and lead to bounded derivations. A derivation is complete when no more rule can be applied. In theory, some non-terminal basic shapes $A(x, y, w, h)$ may remain as underlined because the productions rules with A on the left-

Generic grammar $\mathcal{G}_{\mathsf{gen}}$	
Axiom	$\xrightarrow{\mathsf{v}}$ GroundFloor Floors RoofFloor sky
GroundFloor	$\xrightarrow{\mathsf{h}}$ shop? wall DoorWall wall shop?
DoorWall	$\xrightarrow{\mathsf{v}}$ door wall
Floors	$\xrightarrow{\mathsf{v}}$ wall (Floor wall)+
Floor	$\xrightarrow{\mathsf{h}}$ wall (BalcWins wall)+
BalcWins	$\xrightarrow{\mathsf{v}}$ window | balcony Windows
Windows	$\xrightarrow{\mathsf{h}}$ window | wall (window wall)+
RoofFloor	$\xrightarrow{\mathsf{v}}$ roof? RoofWins roof?
RoofWins	$\xrightarrow{\mathsf{h}}$ roof (BalcWin roof)+
BalcWin	$\xrightarrow{\mathsf{v}}$ balcony? window

Table 2.2: The meta-rules of a generic grammar that can possibly express many facades, with the following segment types: door, shop, balcony, window, wall, roof and sky.

hand side cannot apply due to split positions p incompatible with the current rectangle. In practice, the grammar is generally designed such that the remaining basic shapes are all labeled in $\mathcal{T}$. The language generated by the grammar, i.e., the set of shapes represented by the grammar, is the set of all possible tilings with terminals only as labels, that can be obtained by a derivation process from S. Alternatives in the production rules generally create a combinatorial explosion of the possible tilings.

A derivation can be represented as a tree with a production rule at each node. The root node is a rule whose left-hand side is the start symbol S, and at any level of the tree, a non-leaf node holding rule $A \rightarrow B$ has one child whose rule has B as left-hand side, and a non-leaf node bearing rule $A \xrightarrow{a}_p B\,C$ has two children whose rule have B and C as left-hand sides. Such a derivation tree is also called a parse tree. It can be seen as a tree-shaped graphical model associated to the image, that is constructed dynamically rather than fixed. A complete subtree, a.k.a. bottom-up subtree, is a subtree whose leaf nodes contains rules that have only terminals in their right-hand side. This implies the subtree cannot be further derived, as there is no non-terminal whom to attach a corresponding rule as son.

For complex rules, we have to distinguish derivation trees from derived trees. A *derivation tree* in this case represents as above the successive application of grammar rules from an initial non-terminal: nodes of the derivation tree are grammar rules. A *derived tree* is the combination of trees occurring on the right-hand side of the rules, to form a single tree: here, non-leaf nodes of a derived tree are operators $\mathsf{split}_{a,p}$, and leaf-nodes are terminals or non-terminals. Whereas derivation trees and corresponding derived trees are isomorphic in the case of simple rules (putting aside the case of simple rewritings $A \rightarrow B$), they are not in the case of complex rules, as a single derived tree may originate from different derivation trees. Note also that a derivation tree can be seen as a complex rule that has as left-side the non-terminal of the root rule and as right-hand side the corresponding derived tree.

Parsing an image consists in looking for the best derivation that explains the image. It is generally based on low-level pixel classifiers and or detectors, that produce a set of probabilities, for each pixel, to be of given types $l \in \mathcal{T}$. A parser typically defines the score of a given parse tree based on a comparison between the "observed" pixel classification probabilities and the "expected", regularized pixel class as defined by the rectangular tiling associated to the parse tree. The goal of the parser is to find a parse tree that minimizes (or maximizes) this score. This search is extremely difficult due to the combinatorial explosion of the possible parse trees.

For this reason, existing inference methods require carefully handcrafted grammars that heavily reduce the search space while mostly preserving the applicability of the grammar to parse targeted images. This allows a parser to produce a good result within reasonable time limits.

2.3 Shape grammar parsing

Image parsing with grammars is a complex and challenging optimization task for two reasons. One, the number of unknown parameters to infer is not fixed and evolves during the optimization process, and two, the inference process involves both discrete (specific derivation rules) and continuous variables (derivation parameters). Prominent methods to tackle this problem are based on (i) reversible jump Markov chain Monte Carlo [Ripperda 2006a], (ii) evolutionary computation algorithms [Simon 2012] and (iii) Markov decision processes, in particular reinforcement learning [Teboul 2011b]. In this work, we use the latter for experiments because of the better performance it seems to provide compared to the other methods. Furthermore, the effectiveness of the learned grammar can then be evaluated compared to the handcrafted grammar used in [Teboul 2011b] under identical settings (see Section 2.8 for more details). It actually turns out that our learned grammar is more specific to the architecture style than this handwritten grammar (see also Section 2.4) and thus provides a better accuracy.

2.3.1 Principles of Reinforcement Learning

In reinforcement learning (RL) [Sutton 1998], an agent interacts with an unknown environment while choosing actions that maximize its cumulative reward. The unknown environment is modeled as a Markov Decision Process (MDP), described by a finite set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, transition probabilities P, and expected rewards R consecutive to actions. At time t, the agent in state s_t, takes action $a_t \in \mathcal{A}(s_t)$ leading the agent to a new state s_{t+1} with an immediate reward of r_{t+1}. The transition from state s to s' due to an agent action is subject to the probability $P^a_{ss'}$ and the reward received is an expectation $R^a_{ss'}$ on the distribution $P^a_{ss'}$. Formally, we have:

$$P^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a) \tag{2.14}$$

$$R^a_{ss'} = E[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'] \tag{2.15}$$

The goal of the reinforcement learning agent is to maximize its long term reward which is :

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.16}$$

The parameter γ is a discount factor and represents how much weight we give to the rewards that we will come across in the future. Such a behavior is governed by the agent's policy $\pi(s, a)$, the probability of choosing action a while in state s. This leads to the following state-value function $V^{\pi}(s)$ and action-value function $Q^{\pi}(s, a)$:

$$V^{\pi}(s) = \sum_{a} \pi(s, a) Q^{\pi}(s, a) \tag{2.17}$$

$$Q^{\pi}(s, a) = \sum_{s'} P^{a}_{ss'} \left(R^{a}_{ss'} + \gamma V^{\pi}(s') \right) \tag{2.18}$$

For the most optimal policy π^*, the above two equations lead to the following non-linear Bellman optimality equations:

$$V^{*}(s) = \max_{a} \sum_{s'} P^{a}_{ss'} \left(R^{a}_{ss'} + \gamma V^{*}(s') \right) \tag{2.19}$$

$$Q^{*}(s, a) = \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma \max_{a'} Q^{*}(s', a')] \tag{2.20}$$

These optimal value functions can be approximated using several algorithms, for example the Q-learning algorithm. For a further details please refer to [Sutton 1998].

2.3.2 Reinforcement learning for parsing

Reinforcement learning has been successfully applied to solve the shape parsing problem an as optimization of a top-down geometry (from binary split grammars) of the facade, to fit the bottom-up *merit* responses of a pixelwise classifier [Teboul 2011b]. The pixelwise merit $m(l, x, y)$ provides initial semantic information based on classifiers from the image-level features. It expresses the likelihood that the pixel at coordinates x, y is labeled l. The parsing engine is the agent which can be modeled as a MDP. The state s of the agent is (T, N) where T is a derivation tree and N refers to the non-terminal node that is currently being processed. The agent's action a at state s can be any of the grammar rule that is applicable to N, leading to the next state $s' = (T', N')$. The agent's actions are decided by the policy function reward $\pi(s, a) = P(a|s)$, the probability of choosing action a at state s. The agent's goal is to maximize the rewards that are being obtained from its actions. If multiple non-terminal nodes are generated, N refers to the leftmost non-terminal. Otherwise, N becomes the first unprocessed non-terminal encountered while backtracking in the tree. The different states are the several non-terminal shapes in the grammar for which the rewards are expressed as the sum of its descendant rewards. The goal is to choose a set grammar rules that maximize the reward for the axiom non-terminal. We refer the reader to [Teboul 2013, Teboul 2011a] for more details.

2.4 What grammars to learn?

Given a training set consisting of annotated images, we want to learn a grammar that is able to "parse well" similar unannotated images. Three aspects of such a "good parsing" can be considered, that depend both on the parser and on the grammar: accuracy of the resulting segmentation, parsing speed (and more generally resource consumption), and relative repeatability of the results if the same facade is parsed several times. Indeed, because the solution space is irregular and huge, most parsers only explore a small portion of this space and can be caught in local optimum, yielding sub-optimal results. Besides, most parsers also include randomized procedures and are thus non-deterministic. The convergence property of a RL parser can be studied, e.g., by observing the reward and its standard deviation over time [Teboul 2011a, Ok 2012]. In the following, we focus on accuracy and speed, varying the grammar for a fixed parser; repeatability seems to be a less relevant issue given the experimental data.

As explained below, the quantity and the nature of choices in a grammar are crucial regarding its performance. There are two sources of alternatives in a split grammar: structural choices (possible combinations of meta-rules) and parameter choices for each of these rules (split positions of instantiated rules).

Accuracy is bounded by the language generated by the grammar. If the grammar is too coarse, it will not be able to express some structural or parametric variations of the objects, leading by force to sub-optimal segmentations. For instance, Teboul et al.'s manual grammar for Haussmannian buildings [Teboul 2011b] does not allow wall areas between shop and door, imposes that roof windows are as high as the whole roof, and admits only two kinds of balconies: balcony running over the whole facade width, or balconies being attached to one single window and having exactly the same width. Conversely, if a grammar is too expressive, for instance to possibly cover rare or merely hypothesized cases, the solution space might be too large to search and inaccurate solutions can be produced, although better solutions could exist within the grammar. Speed and stability are also reduced in this case. A balance thus has to be found in the ability of the grammar to cover possible variations. This observation is not restricted to structural choices; it also applies to parameter variation. For example, Teboul et al.'s Haussmannian grammar discretizes split positions with a step of 3 pixels. Although it intrinsically implies sub-optimal splits, it actually results in a better overall accuracy thanks to the reduction of the search space (for a given bound on the number of episodes of the parse).

Besides, different grammars may generate exactly the same language. Yet, some of these grammars may lead to more efficient parses than others. In particular, grammars that impose derivation choices at a time where parsing cues are weak or missing necessitate more backtracking to recover from wrong early choices.

Conversely, a grammar may have different analyses for a given shape. Such a grammar is called ambiguous. As an ambiguous grammar for a given language uselessly increases the search space, we would like to learn unambiguous grammars, i.e., grammars for which any segmentation has at most one corresponding parse tree. However, the fact that a grammar is ambiguous is undecidable for context-free grammars. Thus, in practice, the property that a grammar is unambiguous can only be enforced by construction.

Note that these properties are not all intrinsic to grammars. They may also depend on the actual parser that is used. In the following, we consider the case of Teboul et al.'s parser based on reinforcement learning (see Section 2.3), which is available from the authors and which we have used in our experiments. However, we believe that the general reasoning as well as the qualitative results would be similar to another top-down parser based on a randomized search over the structure and parameter space.

As our goal is to prevent experts from having to manually write and tune grammars, our learned grammars should ideally have a similar or better performance than handwritten grammars, regarding accuracy, speed and stability. The difficulties when writing a grammar concerns the control of the expressive power, the specific encoding of complex patterns, and the tuning of parameters to express likely sizes. They have to be addressed automatically.

Finally, as we not only want to learn the structure of objects but also possible object sizes, we assume that all images (in both the training and the test sets) present the object more or less at the same scale, i.e., the same number of unit length per pixel. For instance, images in the Haussmannian dataset [Teboul 2011b] have been specifically designed to be scaled according to that principle. Images in other datasets have consistent sizes but do not enforce a strict rescaling.

2.5 Generation of ground-truth parse trees

As observed with formal languages and natural languages, a particularly appropriate data model to learn a grammar is the parse tree. However, training data in our case only consist of annotated images. Parse trees thus have, first, to be generated from these images.

An annotated image is a pair of images consisting of a real picture and a label image of the same size. In a label image, each pixel is assigned a label from $\mathcal{T}$ identifying the type of the underlying element at the same location in the real image. Label images express the ground-truth segmentation of the corresponding real images.

2.5.1 Arbitrary, prior-less splits

Different techniques have been proposed to build parse trees from label images [Martinovic 2013a, Weissenberg 2013]. As mentioned in the introduction, Martinovic and Van Gool [Martinovic 2013a] first tile the label image using the horizontal and vertical axis of segment boundaries, and then merge iteratively these tiles, constructing rules and parse trees on the fly. Weissenberg et al. [Weissenberg 2013] prefer to define an energy that gives a score to split line candidates. They use a greedy strategy to recursively split the image using optimal split positions.

These methods have one advantage, which can also be a drawback: they assume no specific knowledge. The problem is that a given label image can be compatible with several parse trees. For instance, a facade with a grid of windows can be analyzed as a set of floors containing rows of windows or as a set of columns of windows, or even as a combination of both. Imposing a minimum description length (MDL) [Martinovic 2013a] is not enough to single out one particular grammar. As a result, very different parse trees can be generated for very similar facades, resulting in suboptimal factorizations in the learned grammar. A

bias can also be imposed to choose specific types of parse trees, e.g., favoring horizontal splits over vertical splits or favoring split axis alternations [Weissenberg 2013]. But it is hard to control in order to guarantee similar analyses for similar images. To prevent arbitrary analyses of label images, we propose to generate ground-truth parse trees using a *generic grammar*.

2.5.2 The idea of a generic grammar

The idea is that the generic grammar should be very small and simple, to be written rapidly with no particular expertise and no tuning required: it should not defeat the purpose of automatically learning a full-fledged specific grammar with adapted parameters from annotated images. More than that, it should actually be able to explain a wide range of structures. The same generic grammar should thus make sense for different datasets, e.g., corresponding to different architecture styles.

Table 2.1 shows a simple generic grammar that has the same structural expressive power as Teboul et al.'s Haussmannian grammar [Teboul 2011a]. Table 2.2 shows another example of a generic grammar that can derive a wide range of facade images comprising the following elements: *wall*, *window*, *balcony*, *roof*, *shop*, *door* and *sky*. Note that these grammars are unambiguous: any resulting segmentation only has one single parse with the grammar. (They are unambiguous because, considering the ground-truth segmentation as the input and starting from the axiom, there is always only one single rule that can be applied to consume a part of the input, with the same label(s) as in the rule, and thus only one rule to grow the corresponding derivation tree. Moreover, this consumption must be maximum, i.e., with terminals of greatest extent, otherwise no further rule can be applied and the derivation is not complete.) Note also that only the meta-rules are shown here. The split parameters of the actual generic grammars are $P = \{1, \ldots, W-1\}$ for horizontal-split rules and $P = \{1, \ldots, H-1\}$ for vertical-split rules.

Note that such a generic grammar only makes sense for the learning task, to generate meaningful parse trees. It cannot be used practically to parse actual facade images, for two reasons. First, it is so general that the solution space would be too large to search, leading to time-consuming and suboptimal parses, and thus to inaccurate and unreliable segmentations. Second, the generic grammar would not be constrained to the specific structure of the learning set, i.e, to a particular architecture style. It thus could not regularize facade analysis with respect to noise (clutter), occlusions or variations of illuminations. However, as shown below, a generic grammar is appropriate to parse ground-truth annotations and generate corresponding ground-truth parse trees.

As an illustration of the inappropriateness of such grammars to directly parse real pictures, Figure 2.2 shows a few examples of parses using the simple generic grammar from Table 2.1 after 5000 episodes of Teboul et al.'s parser [Teboul 2011b]. Actually, 5000 episodes are not enough for convergence. This is in contrast with the handwritten compact grammar for Haussmannian facades, where convergence is typically observed within 2000 episodes of RL parsing [Teboul 2011b]. Moreover, with this generic grammar, some global structural consistency such as window alignment in columns are not modeled.

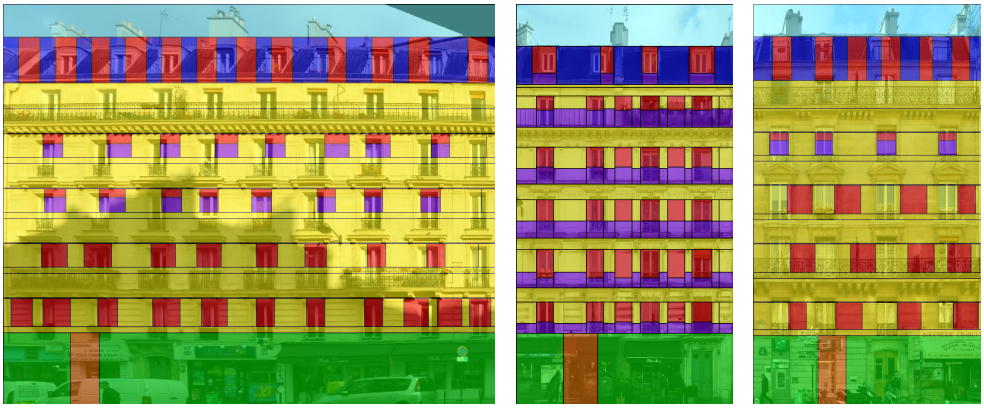


Figure 2.2: Segmentation maps after 5000 iterations of RL parsing [Teboul 2011b] with the generic grammar of Table 2.1.

2.5.3 Ground-truth parse trees from a generic grammar

To produce a ground-truth parse tree, we feed the parser described in Section 2.3 with the generic grammar and the ground-truth label image I_{gt}. Additionally, we replace the usual merit function based on a pixel classifier by the label image itself:

$$m(l,x,y) = \begin{cases} 1 & \text{if } I_{\text{gt}}(x,y) = l \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

With this definition, the merit of a parse tree is equal to the number of corresponding pixels that are assigned the same label as in the ground-truth annotation. The parser tries to maximize this merit, and thus to produce a parse tree whose associated label image matches as much as possible the ground-truth label image.

Although the parser, equipped with the generic grammar, cannot parse real images (in a reasonable time), it is able to successfully parse the ground-truth label images. The reason is that these label images are much more regular and much less noisy than the distribution of label probabilities given by the merit function for real images. It leads to sharper parsing scores and greatly contributes to pruning the search tree. Moreover, as the sampling distribution of split positions in the parser is based on image gradients, the sharp annotations also leads to a small number of sharp peaks in the gradients. There are less decisions to make and good choices are tried first. Some empirical data on parse tree generation using the generic grammar in Table 2.2 are given in Section 2.8.

Note that, at least in theory, any parser could actually be used with the same kind of input for generating ground-truth parse trees. For the same reasons as above, we believe that the convergence would be similarly good with other parsing schemes, e.g., based on rjMCMC [Martinovic 2013a]. Although we could experiment with only one parser (Teboul et al.'s parser [Teboul 2011b], that is publicly available), we believe our approach is not tied to a single parser but has general applicability.

Besides simplicity, one advantage of this approach is also that the generated ground-truth parse trees can be easily understood, as they reuse the same "concepts" and terms

as the generic grammar. This translates as well to the specialized grammars that we infer. For instance, a specific kind of floor in the learned specialized grammar can still be recognized as a floor, and even be given a name derived from the corresponding non-terminal in the generic grammar. (See Section 2.8.5 for a qualitative analysis of Art-deco facades, made easier with this property.) This is in contrast with the other approaches [Martinovic 2013a, Weissenberg 2013], that have to generate arbitrary names. More importantly, one could argue that the grammar we learn is strongly equivalent [Miller 1999] to grammars that would be written by hand for the targeted architecture style: the whole structure of the corresponding parse trees should be equivalent up to some kind of isomorphism, not just their leafs, i.e., the underlying segmentation. This is in contrast with grammars made from trees that are generated as heuristic groupings of segments in ground-truth images [Martinovic 2013a, Weissenberg 2013]. In this case, patterns could for instance be discovered for columns of windows rather than for rows; there would then be nothing like a floor in the corresponding parse trees.

2.5.4 Direct use of parse-tree rules

A grammar specific to the images of a ground-truth training set can be simply produced by just gathering all the rules (including their split parameters) present in the corresponding parse trees. Such a grammar is denoted by $\mathcal{G}_{\text{gt}}$.

While generating parse trees using a generic grammar $\mathcal{G}_{\text{gen}}$, the number of meta-rules present in the trees and thus in $\mathcal{G}_{\text{gt}}$ is bounded by the number of meta-rules in the generic grammar $\mathcal{G}_{\text{gen}}$. However, the number of actual rules (with specific split parameters) can be several orders of magnitude larger, as there can be $H-1$ or $W-1$ different instances of a single meta-rule. It grows initially more or less linearly with the number of training images, until most instances relevant for the training set have been encountered. (See also Section 2.8.)

Such a ground-truth grammar typically comprises most of the rules that are useful to parse an object similar to those in the training set. Even if a few optimal rules are missing because the corresponding split positions do not occur in the training set, close split positions are enough in practice to provide a reasonably good parse. Otherwise it means that the object is not similar to those in the training set.

However, as shown in Section 2.8, the ground-truth grammar $\mathcal{G}_{\text{gt}}$ cannot be used practically for parsing. It requires a large amount of time for convergence and often results in sub-optimal parses. The reason is that it is too large, which yields a huge space to search. Further, it is too general because it accepts any combination of parse fragments associated to different objects. For instance, for buildings, different floors may have different windows alignments and even different numbers of windows, even if, in the training set, all facades have perfect (but different) window grid alignments. The architecture style is thus not captured.

On the contrary, if we create new instances of non-terminals for the rules of each ground-truth parse tree, i.e., if we generate independent sets of rules for each tree, then the only possible parses are those in the ground-truth. An architecture style can somehow be captured in this way, but the grammar totally overfits the learning data: any new object

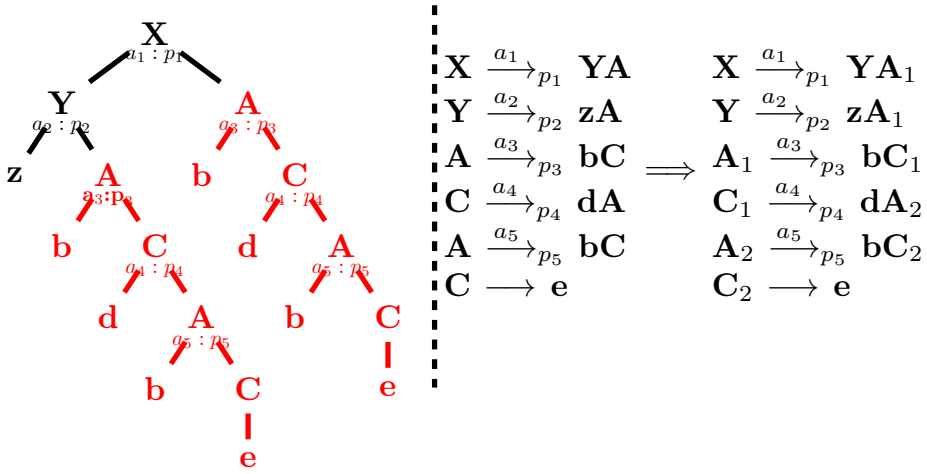


Figure 2.3: Example of rule compression.

can only be analyzed as an object of the ground truth. Consequently, previously unseen objects are parsed very inaccurately.

To produce a sensible grammar suitable for parsing objects similar to the ones in the training set, we need the grammar to be general enough not to overfit the data and specific enough to capture the structure of the objects. It should be large enough to cover some unseen cases but small enough to ensure efficient parsing.

Our approach consists first in identifying repetitions in each parse tree individually, and consider them as instances of the same pattern, specific to the considered facade. This lossless compression captures intra-object regularity in the learning set and improves convergence, but it is too restrictive to generalize to unseen objects. In a second step, we cluster these fixed patterns according to a similarity measure and merge them, introducing appropriate generalization. These operations are described in the following two sections.

2.6 Rule compression

We first consider repetition in a single derivation. More precisely, given a parse tree, we look for complete subtrees that repeat. It identifies patterns within a single object. Specific rules are then introduced to freeze these patterns. (Incomplete subtrees and inter-object patterns are treated in Section 2.7.)

2.6.1 Freezing repeated patterns

Many subtrees can repeat within a single parse tree, revealing different levels of structural and parametric regularity. For instance, in a building, there may be several identical instances of windows with balcony, or several repeated floors with the same layout. We are interested in the largest and most repeating patterns, which we hypothesize are the more likely to be characteristic of a more widespread regularity. More formally, we look for

complete subtrees that maximize the number of repeated nodes:

$$\underset{\substack{U \\ subtree(U,T) \\ nbocc(U,T) \geq 2}}{\arg\max} \quad nbocc(U,T)\, size(U) \tag{2.22}$$

where:

- $subtree(U,T)$ says that U is a complete subtree of T,
- $nbocc(U,T)$ is the number of occurrences of U in T,
- $size(U)$ is the number of nodes in U.

Repetition of subtrees here takes into account both structure and parameters. Two instantiated rules $A \xrightarrow{a}_{p} BC$ and $A' \xrightarrow{a'}_{p'} B'C'$ occurring in the parse tree are identical if $A = A'$, $B = B'$, $C = C'$, $a = a'$ and $p = p'$. However, noise, discretization discrepancies as well as inaccuracies when constructing the ground-truth annotations may result in identical meta-rules $A \xrightarrow{a} BC$ in the parse tree, but with slightly different parameters p, p'. For this reason, we actually consider two instantiated rules to be identical if they stem from the same meta-rule and their parameters p, p' differ only by a certain threshold (see Section 2.8).

Given a repeating subtree U, we then create new rules that represent the pattern only. For this, we duplicate all rules in the subtree, renaming all non-terminals to make sure they are only used once in a rule left-hand side. (In terms of derived trees rather than derivation trees, we rename all non-leaf nodes.) In the following, we note A_i a renamed non-terminal created from an original non-terminal A. The renaming creates as many instances $A_1, \ldots, A_n$ as there are occurrences of A in the subtree. An example of such a transformation is pictured on Figure 2.3.

This removes non-determinism, if any, wherever the pattern is used. As choices inside the pattern are frozen, the language generated by the resulting grammar rules, for a single parse tree, is smaller. It is as if we had introduced a new, complex n-ary rule representing the whole pattern. Note that this operation is much more general than the rule compression transformation of Weissenberg et al. [Weissenberg 2013], that only combines splits along one direction, horizontal or vertical. For instance, their transformation cannot handle a floor pattern (which requires horizontal splits) having identical windows with balcony (which requires a vertical split). Another advantage is that we capture rich patterns into complex rules without the need to change the underlying formalism (splits remain binary) nor the parsers that implement it. As a matter of fact, in all our experiments, we reuse Teboul et al.'s binary split parser as is [Teboul 2011b] (cf. Section 2.3).

2.6.2 Finding repetition via subtree isomorphism

The number of complete subtrees of a tree T is equal to the number of nodes in T. The simplest and most naive way to find the largest repeating complete subtrees in T is to compare each subtree with all the other subtrees, which is computationally expensive. Several efficient algorithms have been proposed to discover most frequent subtrees in ordered

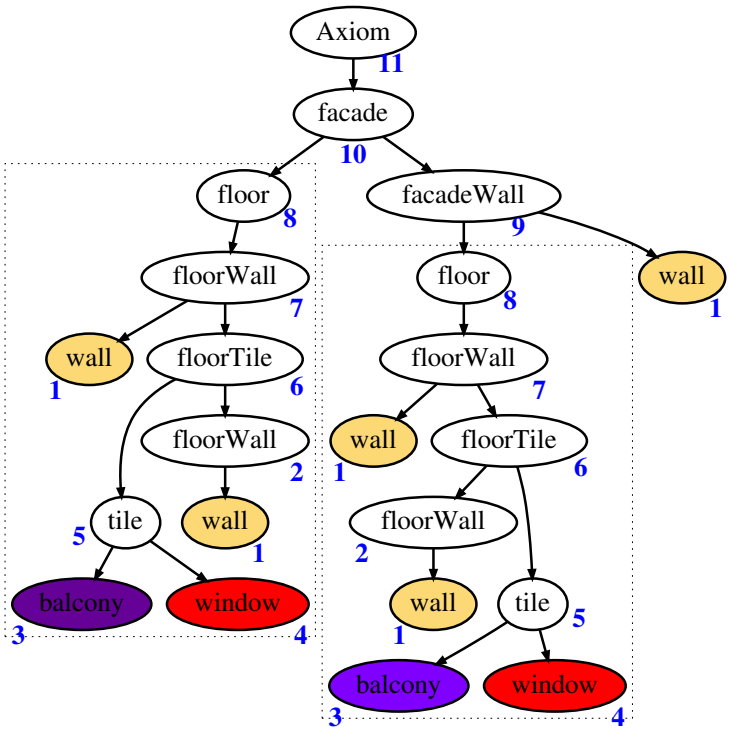


Figure 2.4: Certificates of a parse tree with 2 repeating subtrees.

trees [Chi 2005], making the search mostly linear in the size of the tree. A family of popular approaches turns the issue into a substring matching problem [Mäkinen 1989, Zaki 2002]. We prefer to rely on a proposition of Valiente [Valiente 2002] — actually a variant of a folklore method recalled by Flajolet et al. [Flajolet 1990] —, which is simple and can be adapted to approximate matching, as required to give some tolerance in rule parameter comparison.

Valiente's algorithm for subtree isomorphism computes a certificate for each subtree in a forest, which is a number between 1 and (at most) the number of nodes in T. The certificate is such that two subtrees have the same certificate iff they are identical. Certificates thus provide a partition of the set of subtrees into isomorphic equivalent classes. The assignment of certificates to subtrees is based on a bottom-up traversal of the tree (see Algorithm 1 and 2). When considering a new node, a signature is made from the label of the node and the certificates of its n sons, if any ($n \in \{0, 1, 2\}$). A hash table then maps this signature to the associated certificate. If the signature has not been encountered yet, a new certificate is created and used. For example, in Figure 2.4, both "floor" nodes have a certificate of 8, indicating that both have the same complete subtree starting from these nodes. The complexity is linear on average in the number of nodes in the tree.

Labels in a parse tree are grammar rules, of the form $A \rightarrow B$ or $A \stackrel{a}{\rightarrow}_p BC$. Subtree isomorphism between two nodes requires that labels to be equal, i.e., strict rule equality. To

Algorithm 1 : Subtree isomorphism

1: $H \leftarrow \emptyset$ // Hash table mapping signatures to certificates
2: $c_{\mathsf{new}} \leftarrow 0$ // Counter to make new fresh certificates
3: **for all** u node of T, in bottom-up order **do**
4: let $(c_1, \ldots, c_n)$ be the certificates of sons of u, if any
5: $l \leftarrow \text{label}(u)$ //Get label of subtree at u
6: $s \leftarrow (l, c_1, \ldots, c_n)$ //Make signature of subtree at u
7: $c \leftarrow \text{getCertificate(s)}$ //Make/get cert. for subtree at u

Ensure: identical subtrees $\Leftrightarrow$ identical certificates

Algorithm 2 : getCertificate(s)

1: **if** $s \notin Dom(H)$ **then** // *if signature is unknown yet*
2: $c_{\mathsf{new}} \leftarrow c_{\mathsf{new}} + 1$ // make new fresh certificate
3: $H[s] \leftarrow c_{\mathsf{new}}$ // associate it with signature
4: **return** $H[s]$ // return certificate associated to signature

where $Dom(H)$ is the domain of hash table H.

perform approximate matching, leaving some tolerance in split positions, only the meta-rule part $A \xrightarrow{a} BC$ is used as label in the signature; the parameter p is left out. The hash table now does not only contain a single certificate c for a given signature s; it contains an association between possibly several positions p_i and corresponding certificates c_i. This allows positions close to p_i to be considered as identical and to be given the same certificate c_i. Moreover, rather than use p_i when generating the pattern rules, we actually use the average of all positions assimilated to p_i. For this, we also store in the hash table, along with p_i and c_i, the sum m_i of all encountered positions similar to p_i as well as the number N_i of such positions. Later on, when the pattern is used to generate actual grammar rules, with corresponding parameters, this information can give access to the mean position $\frac{m_i}{N_i}$ of all positions similar to p_i. For this, a minor change is made to Algorithm 1: we replace line 7 by $c \leftarrow \text{getCertificate}(s, n, p)$, where n is the number of sons and p is the split parameter in case $n = 2$. Procedure getCertificate(s, n, p) is defined in Algorithm 3.

To find a complete subtree U in T that maximizes term (2.22), we actually also record the number of times the certificate of U is used. It counts the number of occurrences $nbocc(U, T)$ of subtree U in tree T. After such a U is found, new rules are generated as defined in Section 2.6.1. The hash table is updated accordingly, and the search for repeated subtrees is iterated.

At this stage a subtree-reduced grammar ($\mathcal{G}_{\text{st}}$) can be obtained and used for inference. Compared to the generic split grammar (with all possible parameters) or to the parse-tree grammar (set of rules occurring in ground-truth parse trees), the compressed grammar is much smaller in terms of complex rules (counting as one a whole rule pattern) and much more deterministic. Inference is thus much faster. (See Section 2.8.5 and Table 2.4 for figures on compression factor and convergence rate.) However, the size of the compressed grammar mostly grows linearly with the number of learning images. The reason is that there is no inter-object sharing and no sharing between *similar* patterns, as opposed to

Algorithm 3 : getCertificate(s, n, p)

if $n \neq 2$ **then** *// If rule is not a split rule*
  return getCertificate(s) // Return normal certificate
// If rule is a split rule at p
if $s \in Dom(H)$ **then** *// If signature is already known*
  $(p_i, c_i, m_i, N_i)_{1 \leq i \leq k} \leftarrow H[s]$ // Access remembered info.
  for all $1 \leq i \leq k$ **do** *// For all previously stored p_i*
    if $|p - p_i| \leq \theta$ **then** *// if $p \approx p_i$*
      $m_i \leftarrow m_i + p$ // Sum positions similar to p_i
      $N_i \leftarrow N_i + 1$ // Count positions similar to p_i
      return c_i // Yield same certificate as p_i
// If s unknown or p too different from the p_i's
$c_{\mathsf{new}} \leftarrow c_{\mathsf{new}} + 1$ // Make fresh certificate
$H[s] \leftarrow H[s] \cup \{(p, c_{\mathsf{new}}, p, 1)\}$ // Remember new p for s
return c_{new} // Return new certificate for s

identical ones. In fact, in the case of buildings, we would like to group all facades having the same architectural style independently of the number and values of corresponding attributes. For instance, a 4-window floor could be grouped with a 5-window floor given that the derivation of the former would be a similar subderivation of the latter. This is achieved by the rule merging stage.

2.7 Rule merging

The rule compression stage (cf. Section 2.6) freezes intra-object patterns, restricting rule usage. It also drastically reduces the size of the parse trees and of the corresponding grammar. This size reduction allows more complex transformations, which would otherwise be computationally expensive, to capture richer patterns. The rule merging stage described in this section, to be performed after rule compression, is such a transformation. It captures inter-object patterns and generalizes some of the patterns that have been frozen earlier at rule compression stage.

Given parse trees $T_1, \ldots, T_n$ covering all the learning set, we want to identify similar subtrees and group them. The similarity of subtrees here is looser than for rule compression: we still impose structural equality, i.e., equal meta-rules, but we give more freedom to parameters, allowing somewhat different split positions. More importantly, we allow two kinds of rule pattern matching: either a complete subtree U_i of T_i is fully included in a tree T_j (bottom-up matching), or two trees T_i, T_j share a common partial subtree at the root of both T_i and T_j (top-down matching). In both cases, matching is followed by a merging step that shares the pattern across the dataset and generalizes it where each occurrence of the pattern starts to differ.

In our current framework, we first cluster and merge all repeated subtrees identified at the rule compression stage, i.e. recurring subtrees in individual parse trees separately. For this, we use the bottom-up matching scheme. Then, we cluster and merge all parse trees,

at root level, with the top-down matching scheme.

2.7.1 Clustering rule patterns

Rather than use a greedy approach to enumerate groups of similar subtrees, we prefer to define the pattern search as a clustering problem, which is more principled. The idea is that each given tree or subtree is considered as an object to be grouped with other similar trees or subtrees into clusters. Each cluster then corresponds to a pattern. We require the cluster center to be one of the input tree or subtree. We actually define a distance between objects that favors the fact that the cluster center holds the most general part of the pattern. Other objects in the cluster define variations around this core.

This is a standard unsupervised learning problem and existing clustering algorithms can be used. Note however that centroid-based algorithms such as k-means cannot be used here as we require one of the samples to be the cluster center. Recent clustering techniques such as affinity propagation [Frey 2007] or LP-based clustering [Komodakis 2009] have the additional advantages of being insensitive to initialization and of inferring the optimal number of clusters k, around cluster centers $(C_j)_{1\leq j\leq k}$. In our experiments, we employ the LP-based clustering algorithm [Komodakis 2009] to minimize the following objective function, which is the sum of the distance of each object to its cluster center:

$$\min_{\substack{k\\(C_j)_{1\leq j\leq k}}} \sum_{i=1}^{n} \min_{1\leq j\leq k} d(T_i, C_j) + \alpha \sum_{j=1}^{k} \psi(C_j) \tag{2.23}$$

where

- $d(T, T')$ is the distance between trees T, T' (defined below), satisfying $d(T, T) = 0$,
- $\psi(T) = 1/depth(T)$ is a regularization penalty of choosing T as a cluster center, to avoid the trivial clusterization as a set of singletons, and which favors high trees as cluster centers,
- α is a parameter adjusted to balance the number of clusters, as explained in Section 2.7.3.

Two different distance functions are used for the two different kinds of merging. The distance d_1 is used for bottom-up clustering and merging. It applies to subtrees identified as repeating in the rule compression stage, measuring the similarity of a subtree completely included in another one. The distance d_2 is used for top-down clustering and merging. It applies to rooted parse trees, measuring how similar the common rooted parts are. They are defined as follows:

$$d_1(T, T') = \begin{cases} \rho(U, T') & \text{if } \exists U\, subtree(U, T) \text{ s.t. } U \equiv T' \\ \rho(T, U') & \text{if } \exists U\, subtree(U', T') \text{ s.t. } U' \equiv T \\ \omega & \text{otherwise} \end{cases} \tag{2.24}$$

$$d_2(T, T') = \rho(U, U') \text{ where } (U, U') = T \sqcup T' \tag{2.25}$$

where

- $U \equiv U'$ indicates a structural equality between U and U', not taking into account rule parameters nor non-terminal renaming (cf. Section 2.6.1),
- $\rho(U, U')$ measures the similarity between structurally equivalent trees $U \equiv U'$ (as defined below),
- $subtree(U, T)$ expresses the occurrence of U as a complete subtree of T,
- $T \sqcup T'$ refers to the largest common part (a.k.a. least general generalization or anti-unification) of T and T', taken from the root, considered as a pair (U, U') of structurally equivalent partial subtrees of T and T', i.e., such that $U \equiv U'$,
- ω is a large value preventing the two trees to be part of the same cluster.

Function $\rho(U, U')$ is defined for structurally equivalent trees $U \equiv U'$, which implies $size(U) = size(U')$. It measures the similarity between the rule parameters $(p_u)_{1 \leq u \leq size(U)}$ of U and $(p'_u)_{1 \leq u \leq size(U')}$ of U':

$$\rho(U, U') = \frac{1 + \sum_{1 \leq u \leq size(U)} |p_u - p'_u|}{size(U)} \tag{2.26}$$

The value of ρ increases when parameters differ more or when the size of the common part reduces: this favors, in a same cluster, trees that have a large common part and whose parameters differ little. With this definition, d_1 and d_2 are symmetric, and $d_1(T, T) = d_2(T, T) = 0$.

2.7.2 Merging rule patterns

The merging of rules after clustering is performed as follows. For each cluster $\Gamma = \{T_1, \ldots, T_n\}$, we first consider each instance in each $(T_i)_{1 \leq i \leq n}$ of the largest common part $(U_i)_{1 \leq i \leq n} = \bigsqcup_{1 \leq i \leq n} T_i$ of all elements in the cluster.

Second, we generate a complex rule corresponding to the largest common part. To make sure this rule pattern is "frozen" and specific to the cluster, we rename all non-leaf non-terminals in the largest common part, as in Section 2.6.1, excluding the start symbol if present. We also group the parameters of instantiated rules into single parameterized rules. More formally, for each meta-rule $A \xrightarrow{a} BC$ in the largest common part, which has instances $A \xrightarrow{a}_{p_i} BC$ in each U_i and which is renamed $A_\lambda \xrightarrow{a} B_\mu C_\nu$, we generate a new rule $A_\lambda \xrightarrow{a}_P B_\mu C_\nu$ where $P = \{p_i\}_{1 \leq i \leq n}$. As each simple rule r_j accumulates its own set of specific parameters $P_j = \{p_{j,i}\}_{1 \leq i \leq n_j}$, the complex rule that combines them consequently gets a set of parameter vectors that corresponds to the product of the single-rule parameter sets, i.e., $\prod_{1 \leq j \leq k} P_j$. For meta-rules of the form $A \rightarrow B$ in the largest common part, we simply generate a new rule of the form $A_\lambda \rightarrow B_\mu$ according to the renaming of non-terminals defined by λ.

Last, we need to make sure that the non-terminal B_γ at the root of a newly renamed pattern can be derived from the rules that were deriving B before renaming. This only concerns bottom-up merging; for top-down merging the non-terminal at the root remains the start symbol. Formally, for each rule $A \xrightarrow{a}_P B_i\, C$ such that B_i is the root of the largest

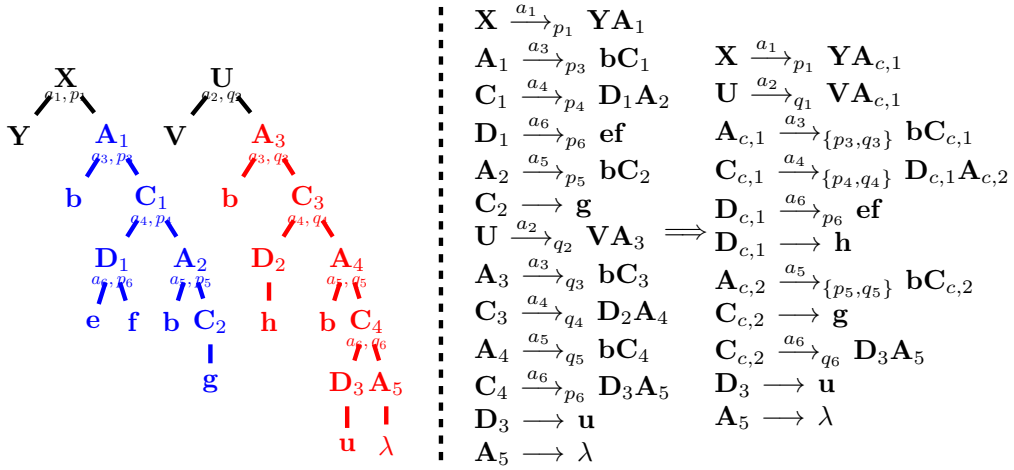


Figure 2.5: Example of rule merging.

common part U_i of Γ in T_i, we generate a new rule $A \xrightarrow{a}_P B_\gamma\, C$. The same applies to rules of the form $A \xrightarrow{a}_P C\, B_i$ and $A \to B_i$.

An example of such a rule merging (bottom-up case) is shown on Figure 2.5.

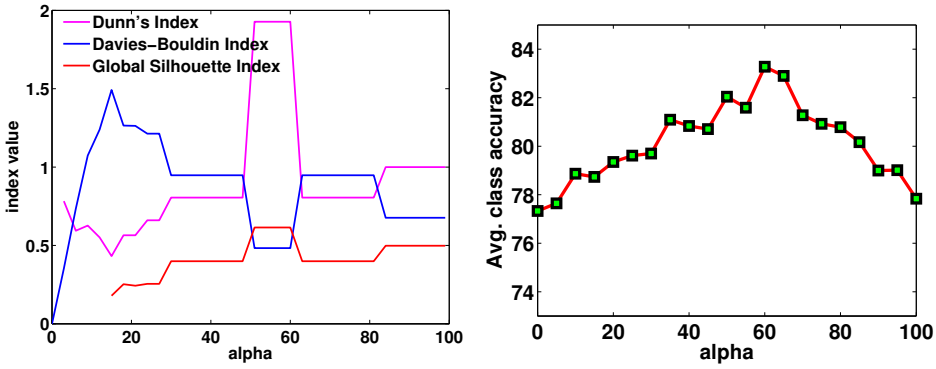
(We think that, if the generic grammar is unambiguous, then the specialized grammars that we generate are unambiguous too. However, we do not have a formal proof of it. In any case, parsing with our generated grammars experimentally has good convergence and accuracy properties, as can be seen from Section 2.8. Even if some specialized grammars contained a form of ambiguity, it does not prevent us from obtaining good results.)

2.7.3 Adjusting clustering parameters

The clustering result depends heavily on the value of α. A very high value of α results in very few cluster centers with large cluster radius, while a small α value could result in each data-point being a cluster center. In order to determine the optimal value of α, we consider three well-known indices, namely the Dunn's index [Dunn 1974], the Davies-Bouldin index [Davies 1979] and the Silhouette index [Rousseeuw 1987]. These indices are based on already clustered data. They combines measures of cluster compactness (distances between cluster members) and cluster separation (distances between clusters vs within clusters). Given a distance d, they are defined as follows given k clusters $(\Gamma_i)_{1\leq i\leq k}$ with respective centers $(C_i)_{1\leq i\leq k}$.

Dunn Index [Dunn 1974]: This metric is defined as the ratio between the minimal inter-cluster distance and the maximal intra-cluster distance:

$$D = \frac{\min\limits_{1\leq i<j\leq k} d(C_i, C_j)}{\max\limits_{1\leq i\leq k} \max\limits_{X,Y\in\Gamma_i} d(X,Y)} \tag{2.27}$$



(a) (b)

Figure 2.6: (a) Clustering index plots on the validation set for one fold (ECP2011 dataset). (b) Impact of α on average class accuracy on the test set of the fold from Figure-2.6(a) (ECP2011 dataset).

A higher Dunn index indicates better clustering.

Davies-Bouldin Index [Davies 1979]: As Dunn index, this metric measures cluster compactness vs cluster separation. It is defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k}\left\{\max_{\substack{1\leq j\leq k\\ j\neq i}}\left\{\frac{\bar{d}_i + \bar{d}_j}{d(C_i, C_j)}\right\}\right\} \tag{2.28}$$

$$\bar{d}_i = \frac{1}{|\Gamma_i|}\sum_{X\in\Gamma_i} d(X, C_i) \tag{2.29}$$

where is $\bar{d}_i$ is the average distance of members of Γ_i to the cluster center C_i. A lower *DB* value indicates a better separation of the clusters and a greater proximity among members of a cluster.

Global Silhouette Index [Rousseeuw 1987]: Contrary to the previous two indices, this metric takes into account the distance among all members in a cluster, not just with the cluster center. It is defined as:

$$GS = \frac{1}{k}\sum_{i=1}^{k}\left\{\frac{1}{|\Gamma_i|}\sum_{X\in\Gamma_i}\frac{b_i(X) - a_i(X)}{\max(a_i(X), b_i(X))}\right\} \tag{2.30}$$

$$a_i(X) = \frac{1}{|\Gamma_i| - 1}\sum_{Y\in\Gamma_i, Y\neq X} d(X, Y) \tag{2.31}$$

$$b_i(X) = \min_{1\leq j\leq k, j\neq i}\frac{1}{|\Gamma_j|}\sum_{Y\in\Gamma_j} d(X, Y) \tag{2.32}$$

where $a_i(X)$ is the average distance between X and the other elements in the same cluster Γ_i, and $b_i(X)$ is the lowest average distance of X to other clusters. A higher index indicates a better clustering.

Choice of parameter α. The above three indices are used in the experiment section to define α. The best value of α, to produce well-partitioned clusters, corresponds the maximum of Dunn and Global Silhouette indices and to the minimum of the Davies-Bouldin index. Although they differ in their formulation, these indices mostly agree on the kind of data we are clustering, as can be seen in Figure 2.6(a). Rather than select a single index, and as their computation cost is negligible, we choose the value of α such that the ratio $\frac{D \times DB}{GS}$ is maximum, which could add some robustness in case one of the indices would disagree with the others. Other authors [Parisot 2011, Parisot 2012] have used a similar treatment.

2.8 Results

In this section, we provide both quantitative and qualitative results using the proposed framework and compare with state-of-art. We experimented our method on three existing standard datasets of rectified annotated facade images: ECP2011 [Teboul 2011b], Graz2012 [Riemenschneider 2012] and CMP2013 [Tylecek 2012]. In addition, we evaluated our approach with ENPC2014, a new dataset with yet a different architecture style, that we have collected specifically to illustrate the applicability of our approach to a variety of structural constraints and to study the sensitivity of grammar learning to architecture styles.

Most facades pictured in these datasets represent buildings that contain a notable amount of regularity, both across the dataset and within the facade itself. For instance, they typically have at least three floors and at least three windows per floor, that are laid out according to one or two grid-like patterns, with possible variations in position and size though. This is an appropriate setting for segmenting with a grammar-based prior, and also for learning grammatical patterns from just a few tens of annotated samples. On the contrary, grammatical approaches are less suited for datasets that feature facades with little regularity, e.g., with few windows, highly uneven layouts and strong architectural inconsistencies, such as eTRIMS [Korc 2009]. For such datasets, grammatical priors have to be relaxed [Cohen 2014a, Koziński 2015b]. Naturally, trying to learn grammars from such datasets is inappropriate too, especially if the number of images is small, e.g., 60 annotated images in the eTRIMS dataset.

For all our experiments, we use the RL parser made available by Teboul et al. [Teboul 2011b], with default settings, on an Intel Xeon E3-1225 CPU 3.2GHz. Unless otherwise mentioned, we use the generic grammar of Table 2.2 ($\mathcal{G}^2_{\text{gen}}$) to generate ground-truth parse trees, and we use DARWIN [Gould 2012a] with default settings to generate specific pixel classifiers from annotated images, independently for each dataset. We first study the accuracy of parsing using the learned grammars: we report classwise accuracy, average class accuracy, overall pixel accuracy and average intersection-over-union score (IoU). We also evaluate

		Door	Shop	Balcony	Window	Wall	Sky	Roof	**Average**	**Overall**	**IoU**
	[Teboul 2011b]	47	88	58	62	82	95	66	71.1	74.7	-
	[Martinovic 2013a]	50	81	49	66	80	91	71	69.7	74.8	-
RF unaries +	**[Weissenberg 2013]**^{1}	20	84	30	24	74	**99**	33	51.9	62.9	36.5
Grammar induced	**[Weissenberg 2013]**^{2}	26	85	42	48	78	97	34	58.6	69.3	42.1
from $\mathcal{G}^1_{\mathbf{gen}}$	$\mathcal{G}_{\mathbf{gt}}$	19	79	24	26	71	95	29	49.1	59.9	34.3
	$\mathcal{G}_{\mathbf{st}}$	41	85	51	58	78	92	63	66.9	73.1	55.4
	$\mathcal{G}_{\mathbf{cl}}$	52	86	55	64	83	92	67	71.3	76.2	57.6
	[Weissenberg 2013]^1	49	87	58	52	79	99	52	67.9	74.2	54.8
DARWIN unaries+	**[Weissenberg 2013]**^{2}	54	89	69	59	83	96	58	72.6	78.6	57.3
Grammar induced	$\mathcal{G}_{\mathbf{gt}}$	48	88	66	56	76	96	54	66.5	71.8	52.3
from $\mathcal{G}^2_{\mathbf{gen}}$	$\mathcal{G}_{\mathbf{st}}$	57	90	78	67	85	96	73	78.1	82.6	67.7
	$\mathcal{G}_{\mathbf{cl}}$	**62**	**94**	**84**	**72**	**89**	98	**79**	**82.5**	**86.9**	**71.8**
State of art	**[Martinovic 2012]**	60	86	71	69	93	97	73	78.4	85.1	-
(no grammar)	**[Cohen 2014a]**	**79**	**94**	**91**	**85**	90	97	**90**	**89.4**	**90.8**	-

Table 2.3: Segmentation results on the ECP2011 dataset: [Teboul 2011b] uses a hand-crafted grammar; [Martinovic 2013a] infers a grammar but without strong constraints such as grid alignments; [Martinovic 2012] and [Cohen 2014a] are state-of-the-art methods with hard-coded constraints (that are soft or that do not cover all architectural constraints).

the grammars in terms of scalability, size and inference performance. In all our experiments, unless otherwise specified, we use a 5-fold cross-validation setup similar to [Martinovic 2012, Martinovic 2013a, Cohen 2014a], with 60% of the images for grammatical inference and pixel classifier generation, 20% for choosing the value of α, and the remaining 20% for testing. For each experiment with one of our grammar learning method, we thus actually generate 5 pixel classifiers, 5 specialized grammars, and average the resulting figures. Concerning rule compression, we set the similarity threshold mentioned in Section 2.6.1 to 10 pixels, except for the CMP2013 dataset for which it is set to 30 pixels because the images have a higher resolution.

To somehow compare with Weissenberg et al. [Weissenberg 2013], despite the fact that they do not evaluate their generated grammars for parsing, we replicated the part of their framework that deals with grammatical inference, namely transformation to n-ary split nodes [Weissenberg 2013, Sect. 4.1] and production rule inference by parameter merging [Weissenberg 2013, Sect. 4.2]. Note however that we did not replicate their method for generating ground-truth parse trees [Weissenberg 2013, Sect. 3.3]; in the following experiments, we always use as ground-truth parse trees the ones we obtain from the generic grammar approach (cf. Section 2.5.3). The parsing and size comparison with Weissenberg et al.'s method that we provide thus only concerns the grammar generation from *our* ground-truth parse trees.

For the rest of this section, we use the following notations to represent the induced grammars from different steps of different frameworks:

- [Weissenberg 2013]^1, [Weissenberg 2013]^2 represent the grammars induced by n-

		Convergence time (s)	# of episodes	Derivation length
Grammar induced from $\mathcal{G}^1_{\text{gen}}$	**[Teboul 2011b]**	22	1740	108
	[Weissenberg 2013][1]	13.4	1117	**26**
	[Weissenberg 2013][2]	7.1	695	28
	$\mathcal{G}_{\text{gt}}$	24	1956	103
	$\mathcal{G}_{\text{st}}$	7.5	489	42
	$\mathcal{G}_{\text{cl}}$	**6.6**	**306**	27
Grammar induced from $\mathcal{G}^2_{\text{gen}}$	**[Weissenberg 2013]**[1]	18.1	1421	31
	[Weissenberg 2013][2]	10.8	876	35
	$\mathcal{G}_{\text{gt}}$	32	2518	122
	$\mathcal{G}_{\text{st}}$	9.8	720	49
	$\mathcal{G}_{\text{cl}}$	8.9	580	37

Table 2.4: Performance comparison of handcrafted grammar [Teboul 2011b] w.r.t. learned grammar on ECP2011: average parsing time, median number of episodes for convergence and average derivation length.

ary composition [Weissenberg 2013, Sect. 4.1] and then parameter merging [Weissenberg 2013, Sect. 4.2], using our implementation of their method and our ground-truth parse-trees.

- $\mathcal{G}_{\text{gt}}$, $\mathcal{G}_{\text{st}}$, $\mathcal{G}_{\text{cl}}$ represent, respectively, the grammar inferred directly from the ground-truth parse trees (Section 2.5), after subtree reduction (rule compression, Section 2.6), and after clustering (rule merging, Section 2.7).

2.8.1 ECP2011 Haussmannian dataset [Teboul 2011b]

The ECP2011 dataset [Teboul 2011b] consists of 104 annotated images of Haussmannian buildings in Paris. For this set of images, we use the new, more accurate ground-truth annotations released by Martinovic [Martinovic 2012]. We consider two experimental settings. In the first one, we use for grammar inference the simple generic grammar ($\mathcal{G}^1_{\text{gen}}$, shown in Table 2.1), and for parsing a pixel classifier based on a random forest (RF) [Teboul 2010b]. This makes our results directly comparable to published results obtained in the same setting [Teboul 2010b, Teboul 2011b, Martinovic 2013a], i.e., with the same expressive power of the grammar (e.g., only single-window or whole-facade running balconies) and with the same pixel merits. In the second setting, we use for grammatical inference the richer generic grammar ($\mathcal{G}^2_{\text{gen}}$, shown in Table 2.2), which allows more architectural variation, and better pixel merits from DARWIN [Gould 2012a]. The feature vector used in DARWIN includes RGB color information, HoG descriptor, LBP texture descriptor and normalized pixel location.

We provide a detailed comparison of our approach with existing methods in terms of both accuracy (Table 2.3) and convergence time (Table 2.4). For the grammars that we generate, we run the RL parsing algorithm for a maximum of 10 seconds per image. For

Teboul et al.'s RL parser with a handwritten grammar [Teboul 2011b], we report the figures given by Martinovic and Van Gool [Martinovic 2013a] as the figures first provided by the authors were in a different setting, with a less accurate ground-truth [Teboul 2011b].

With weak pixel merits from a RF classifier, our method performs better than the handcrafted grammar from [Teboul 2011b] and better than the generated grammars from [Martinovic 2013a]. Comparing with the grammar induction framework from [Weissenberg 2013], we achieve better segmentation result with our learned grammar at a faster convergence rate. The manually-written grammar consists of 19 parametric rules, representing 281 instantiated rules. Comparing with the handcrafted grammar, the learned grammar ($\mathcal{G}_{\text{cl}}$) from $\mathcal{G}^1_{\text{gen}}$ is more efficient at least by a factor of five in terms of number of episodes required and by a factor of three in terms of wall clock time for convergence. One of the reasons might be that, thanks to such a compact grammar, the average length of the derivation sequence (counting complex rules as one) is reduced by a factor of three. Although we compare favorably to grammar-based methods, whether the grammar is written by hand or learned automatically, and even to some weakly-constrained segmentation methods [Martinovic 2012], our approach does not reach the accuracy of the state-of-the-art hard-coded segmentation method [Cohen 2014a]. It might be due to the fact that they use a very good pixel classifier and/or because they do not try to enforce as many hard constraints as we do.

To show the role of α, we plot the value of α against the average class accuracy for one fold on the ECP2011 dataset (see Figure 2.6(b)). Intuitively, a high value of α implies fewer number of clusters with large cluster size. This induces a major generalization in the learned grammar, which enlarges the search space, potentially leading to suboptimal parse. And for a low value of α, there will be a large number of clusters, shrinking the generalization capability of the learned grammar; the learned grammar would overfit the training data and not be adapted to unseen images, leading to inaccurate parses. An appropriate value of α is thus one for which the generalization capacity of the learned grammar is balanced. This can be seen from Figure 2.6(b) by observing the average class accuracies for very high and very low values of α. Note that the best value for α in this case happens to be about the same as the one we compute automatically in a similar setting (see Figure 2.6(a)). Figure 2.13 shows some visuals results.

2.8.2 Graz2012 Dataset [Riemenschneider 2012]

The Graz2012 dataset [Riemenschneider 2012] consists of 50 images. A majority of them represent the Gruenderzeit architecture style which is common in Germany and Austria. As there are only 4 classes in this dataset, namely *door*, *window*, *wall* and *sky*, we had to downgrade the generic grammar to discard the other terminals, i.e., *shop*, *roof* and *sky*.

Classwise accuracies are shown in Table 2.5, with a comparison to Riemenschneider et al.'s method [Riemenschneider 2012]. Our learned grammar outperforms the other methods. The average number of episodes for convergence was observed to be 180 with the learned grammars, while the average derivation length was 22. The number of optimal clusters was found to be 21 for this dataset, and the average number of rules in the clustered grammar was 29. Figure 2.11 shows some visual results.

		DARWIN unaries				
	[Riemenschneider 2012]	[Weissenberg 2013][1]	[Weissenberg 2013][2]	$\mathcal{G}_{\text{gt}}$	$\mathcal{G}_{\text{st}}$	$\mathcal{G}_{\text{cl}}$
Door	41	29	33	31	39	**43**
Window	60	64	66	62	69	**76**
Wall	84	84	87	82	89	**91**
Sky	91	**95**	94	93	92	92
Average	69	68.2	70.1	66.9	72.3	**75.6**
Overall	78.0	79.3	81.9	77.4	83.9	**86.6**
IoU	58.0	61.6	63.2	59.4	63.1	**68.4**

Table 2.5: Segmentation results on the Graz2012 dataset.

2.8.3 CMP2013 Dataset [Tylecek 2012]

The CMP dataset [Tylecek 2012] contains a mixture of worldwide styles including a majority of Prague buildings. It consists of 378 images of diverse facades with ground-truth annotations initially provided for eleven classes *facade*, *molding*, *cornice*, *pillar*, *window*, *door*, *sill*, *blind*, *balcony*, *shop* and *deco*, plus one class for the *background*, corresponding to cropped areas after image rectification. To enable a comparison of our method across different datasets and different kinds of architecture, we did not try to extend the generic grammar $\mathcal{G}^2_{\text{gen}}$ to cover all the extra classes. We had to adapt it nonetheless because the dataset does not include classes *sky* and *roof*. We thus downgraded the generic grammar to the five classes *shop*, *door*, *balcony*, *window* and *wall*.

To compare the resulting accuracy with the figures reported by Tylecek [Tylecek 2012], we also had to merge or ignore some of his classes, based on the reported covariance matrix. While the *shop*, *door* and *balcony* classes are taken directly, the accuracy we give for the *window* class in [Tylecek 2012] is actually a combination of the figures for the original labels *window* and *blind*. Similarly, all the other labels are merged into a unique *wall* class, except the *background* class that is ignored by all methods. Classwise accuracy is shown in Table 2.6. The average number of episodes for convergence was observed to be 1200 with the learned grammars, while the average derivation length was 32. The average number of rules in the learned grammar was 78. Figure 2.12 shows few visual results.

2.8.4 ENPC2014 Art-deco Dataset

The Haussmannian style, as illustrated in the ECP2011 dataset, features facades with high regularity, not only regarding window layout but also concerning window sizes, which often have the same width across an entire facade. To demonstrate that architecture-specific grammars are required for a better parsing (see Section 2.8.7), a dataset with identical semantic classes but different architecture style is needed. For this reason, we have constructed a new dataset, called ENPC2014, with 79 images of Art-deco buildings in Paris. Although they have commonalities with Haussmannian facades, Art-deco facades actually differ, in particular in the typical sizes of windows (which can be wider) and in the number

		DARWIN unaries				
	[Tylecek 2012]	[Weissenberg 2013][1]	[Weissenberg 2013][2]	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$
Door	**54**	38	39	46	45	49
Shop	59	61	63	59	63	**66**
Balcony	**46**	26	27	24	25	32
Window	**59**	44	46	49	52	57
Wall	84	81	83	76	86	**89**
Average	**60.4**	50.2	51.6	50.9	54.2	58.8
Overall	78.3	70.4	72.34	67.7	75.6	**82.5**
IoU	-	35.5	37.8	34.5	39.7	**42.4**

Table 2.6: Segmentation results on CMP2013 dataset.

	DARWIN unaries				
	[Weissenberg 2013][1]	[Weissenberg 2013][2]	$\mathcal{G}_{\mathbf{gt}}$	$\mathcal{G}_{\mathbf{st}}$	$\mathcal{G}_{\mathbf{cl}}$
Door	49	53	41	56	**59**
Shop	78	84	78	85	**88**
Balcony	49	57	46	57	**63**
Window	51	59	46	58	**66**
Wall	72	79	78	77	**84**
Sky	**97**	96	95	95	92
Roof	52	54	49	56	**58**
Average	64.1	68.9	61.8	69.1	**72.9**
Overall	68.4	74.3	69.5	73.4	**78.8**
IoU	48.0	57.8	48.2	55.1	**59.4**

Table 2.7: Segmentation results on the ENPC2014 dataset.

of floors (which can be higher). The balcony layout may also be different. Besides, the dataset includes some layout inconsistencies due to image rectification as some windows and balconies are often protruding in the Art-deco style. It is similar to the case of roof windows, already present in the ECP2011 Haussmannian dataset, which often are not in the same plane as the other facade windows. Similar to ECP2011, images in ENPC2014 are segmented and annotated into seven classes, *door*, *shop*, *balcony*, *window*, *wall*, *sky* and *roof*. The segments in the ground-truth annotations we defined follow a rectangular regularity, but no alignment is artificially enforced. The dataset is publicly available[1].

Concerning our experiments, we use the same generic grammar $\mathcal{G}^2_{\text{gen}}$ as with the other datasets to generate our specialized grammars. Accuracy results are reported in Table 2.7. The average number of episodes for convergence was observed to be 670 with the learned Art-deco grammars, while the average derivation length was 30. The number of optimal clusters were found to be 18 for this dataset. Figure 2.14 shows few visual segmentations

[1] https://github.com/raghudeep/ParisArtDecoFacadesDataset/

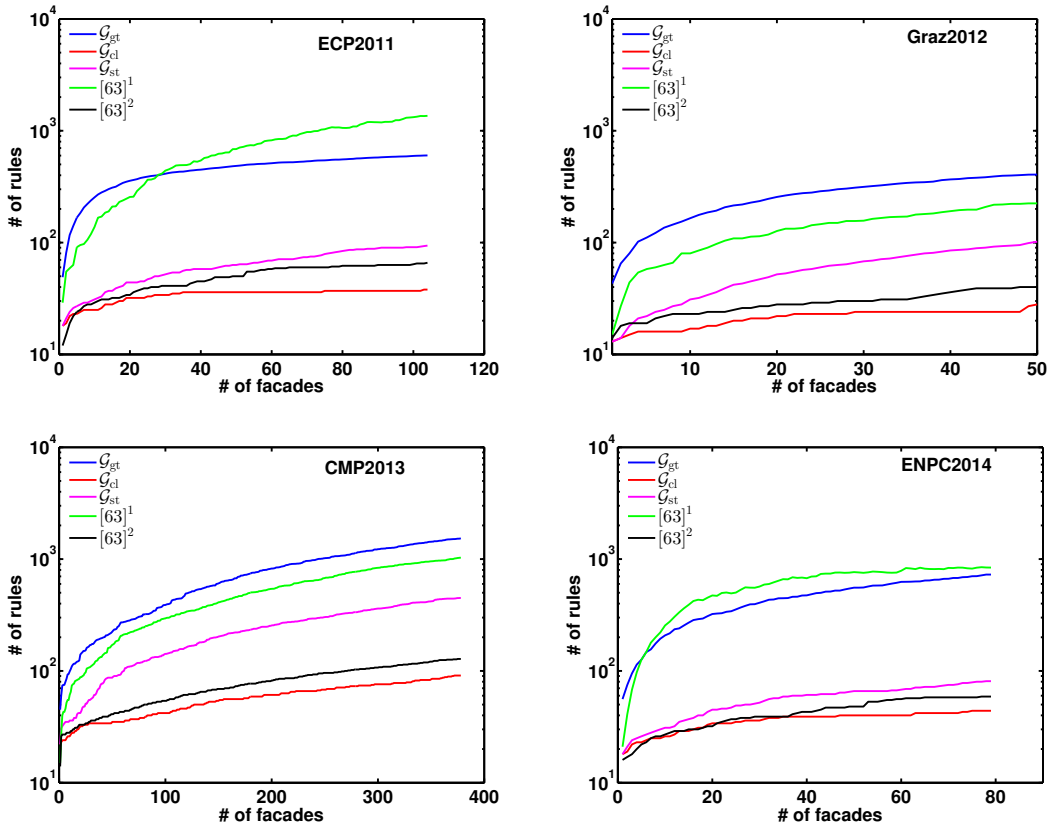


Figure 2.7: Number of rules in the learned grammar (Y-axis) w.r.t. the size of training set (X-axis). Notice the log scale of Y-axis.

on this dataset.

2.8.5 Scalability and qualitative analysis

To provide an insight on the scalability of our grammar learning method, we plot in Figure 2.7 the number of inferred rules against the size of the training set. For the ECP2011 dataset, the number of rules in the learned grammar is almost saturated after 25 samples, validating the claim of [Weissenberg 2013]. For the ENPC2014 dataset, the most common rules correspond to: (i) two large widely separated windows, on the first and fifth columns, (ii) large window in the middle (third) column, (iii) running balcony on the top floor. (Such an interpretation in made easier by the fact that our inferred grammars are generated from a generic grammar that already has an understandable semantics.) For the datasets CMP2013 and ENPC2014, the number of rules continues to grow with the training samples, indicating the diversity of the dataset and underlying architecture styles. Note that the numbers of rules provided here by our implementation of Weissenberg at al.'s method are a bit smaller than the values reported in the authors' paper [Weissenberg 2013]. This could be explained by the fact that we find more complex rule patterns, as we can combine both horizontal and

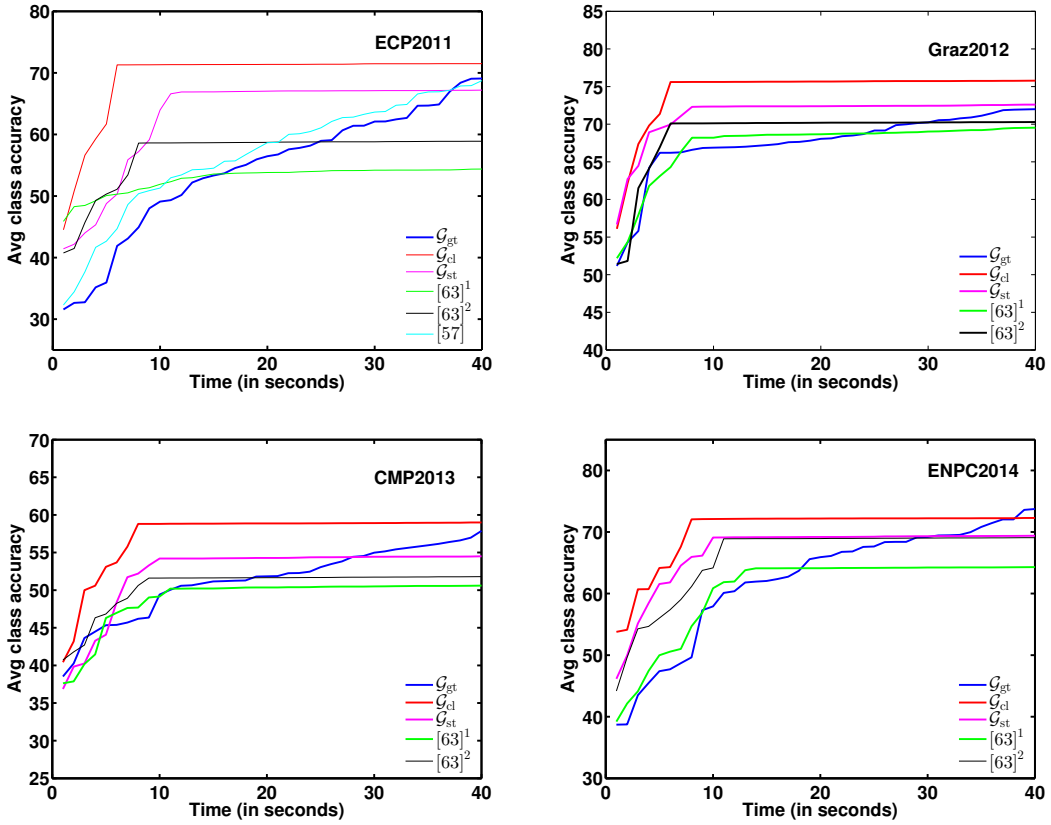


Figure 2.8: Average class accuracy (y-axis) w.r.t. time (x-axis).

vertical splits, and/or by the fact that our input ground-truth parse trees generated from a generic grammar display more regularity than the parse trees discovered by the heuristics in [Weissenberg 2013].

Computations for the rule compressing steps in our implementation took 5 ms per facade, on average, on the ECP2011 dataset. The clustering step took 15 ms on single core of Intel Xeon E3-1225 machine. As for extracting the ground-truth parse trees, we run the RL parser for a maximum of 15 s per annotated image. However, convergence was observed in 4.8 s on average. Please note that rule compression can be applied in parallel on all facades of the training set. Martinovic et al. do not report running times but it seems their approach does not scale well as, in their experiments, the authors limit the training sets to "30 images to keep the induction time within reasonable bounds" [Martinovic 2013a]. Weissenberg et al. [Weissenberg 2013] report that their inference algorithm takes about 32 ms per facade on an Intel Core i7 930. Their inference method is mostly linear and works online. As we are currently relying on the LP-based clustering of Komodakis et al. [Komodakis 2009], our implementation is not online, but it could be made so using an online clustering algorithm. In any case, learning time probably is not an issue given the current performance and the typical size of the training sets. Larger orders of magnitude for the number of images with handmade ground-truth annotations would defeat some of the interests of generating

	$\mathcal{G}_{\mathbf{AA}}$	$\mathcal{G}_{\mathbf{AH}}$	$\mathcal{G}_{\mathbf{HA}}$	$\mathcal{G}_{\mathbf{HH}}$
Door	59	56	57	62
Shop	88	86	83	94
Balcony	63	51	54	84
Window	66	56	48	72
Wall	84	71	76	89
Sky	92	82	92	98
Roof	58	68	51	79
Average	72.9	67.1	65.9	82.5
Overall	78.8	71.9	70.8	87.0
IoU	59.4	55.8	57.6	71.8

Table 2.8: Cross-dataset analysis using Art-deco (A) and Haussmannian (H) facades. $\mathcal{G}_{\mathrm{AH}}$ represents the grammar learned using annotated Art-deco facades and applied on Haussmannian facades images. Others follow similarly.

a grammar automatically to reduce the human burden on this task.

Figure 2.8 shows the performance of induced grammars from different stages of our framework and also a like-for-like comparison with different grammars obtained by our implementation of Weissenberg et al.'s framework. For these experiments, the RL parser is run for 40 seconds and the average class accuracy is plotted with respect to time.

2.8.6 Sensitivity to the accuracy of pixel classifiers

A question that arises is how much the underlying pixel classifier, that the parser uses to evaluate sampled layout configurations, impacts the performance of a given grammar. (Note that the goal here is not to reach the best pixelwise accuracy possible, but still to perform a structural segmentation that follows architectural constraints.) For this, we experimented with 4 different unaries: random forests (RF) [Teboul 2010b], DARWIN [Gould 2012a], Auto-Context (AC) [Jampani 2015], and the ground-truth (GT) labeling itself. We consider 6 different grammars (or more precisely, 6 families of grammars in the case of grammar generation, as we follow a 5-fold cross-validation in this case): the handwritten grammar of Teboul et al. [Teboul 2011b], grammars generated by the two variants of Weissenberg et al.'s method [Weissenberg 2013][1] and [Weissenberg 2013][2], and grammars generated by our 3 variants $\mathcal{G}_{\mathrm{gt}}$, $\mathcal{G}_{\mathrm{st}}$, $\mathcal{G}_{\mathrm{cl}}$. For all experiments, the RL-based parser is run under identical settings, with 2000 iterations. Figure 2.9 shows the overall pixel accuracy on the ECP2011 dataset for these 4 pixel classifications and 6 grammars. As can be seen, for a given grammar, the better the pixel classifier, the better the resulting pixel accuracy after parsing. Besides, the quality ranking of the grammars is preserved when the accuracy of the pixel classification increases. This shows that the quality of the grammars (or grammar generators) is relatively independent of the underlying pixel classifier used by the parser. In these experiments, our approach consistently performs better.

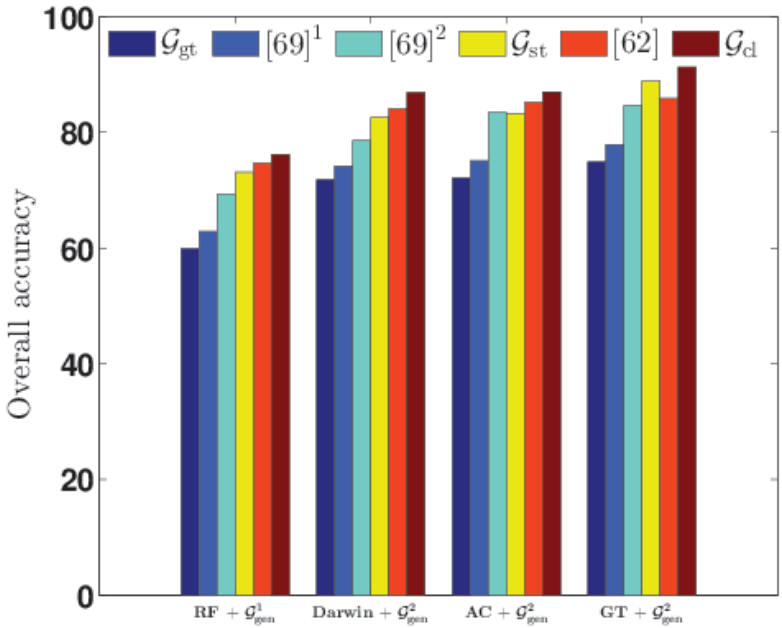


Figure 2.9: Performance of hand-crafted and learned grammars using different unaries on the ECP2011 dataset.

2.8.7 Cross-dataset analysis

To investigate commonalities and dissimilarities in grammar rules between different styles of architecture, we operate our learned Haussmannian grammar on Art-deco facades, and the other way around. Figure 2.10 shows such segmentation results on two images. The most common rule between these two styles corresponds to a running balcony on the top floor of a facade. And the most distinctive rules are (i) periodical large windows in the Art-deco style and uniformly-sized windows in the Haussmannian style, (ii) the number of floors: seven in Art-deco and five in Haussmannian. Not only this experiment provides an insight in understanding common rule patterns across different styles, but it also strengthens the need for style-specific grammars. Table 2.8 shows the performance of grammar learned using Art-deco and Haussmannian styles on Haussmannian and Art-deco facades.

2.9 Conclusion

In this chapter, whose content has been published in the International Journal of Computer Vision [Gadde 2016b], we have proposed a novel method for learning split grammars from annotated images, and we have used it to learn typologies of architectures. The method assumes a simple generic grammar which is used to parse the training set. Reasoning on the associated derivation trees, to first identify common subtrees and then merge similar trees, determines the set of meta-rules corresponding the observed typology of buildings. It leads to a compact (in terms of derivation trees) and simple (in terms of inference process) grammar. State-of-the-art results with respect to typology-specific handcrafted grammars

Figure 2.10: Cross-comparison of learned Art-deco and Haussmannian grammars on Haussmannian and Art-deco facades. Top, from left to right: Art-deco facade, analyzed with Art-deco grammar, analyzed with Haussmannian grammar, and disagreement map (white color for differences). Bottom: Haussmannian facade, analyzed with Haussmannian grammar, analyzed with Art-deco grammar, disagreement map.

or to grammars learned from data demonstrate the extreme potentials of our method.

Extending this to other typologies of architecture is an obvious work, such as applying the concept to modern architectures. Such a task will possibly benefit from improved likelihoods of image classes [Martinovic 2012, Jampani 2015]. Improving the process of establishing the set of meta-rules by reasoning simultaneously on the compact derivations of all training examples is a natural extension of our method. Considering more trees at the rule-merging stage should also lead to an improved performance. Furthermore, extending this approach to 3D grammars is an extremely promising task, and in particular when taking into account the difficulty of defining such a grammar manually.

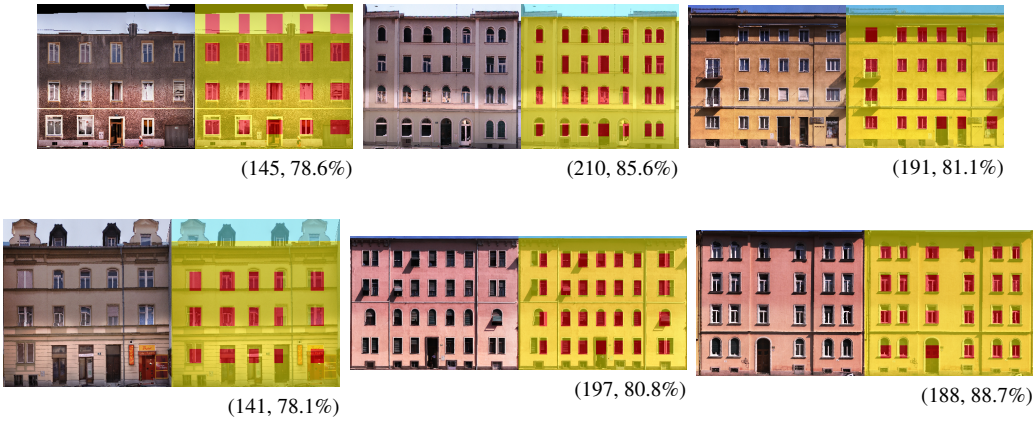


Figure 2.11: Qualitative results on Graz2012 dataset. Image (left) and segmentation using learned grammar $\mathcal{G}_{\text{cl}}$ (right) are shown here along with number of episodes for convergence and segmentation accuracy.

Figure 2.12: Qualitative results on CMP2013 dataset. Image (left) and segmentation using learned grammar $\mathcal{G}_{\text{cl}}$ (right) are shown here along with number of episodes for convergence and segmentation accuracy.

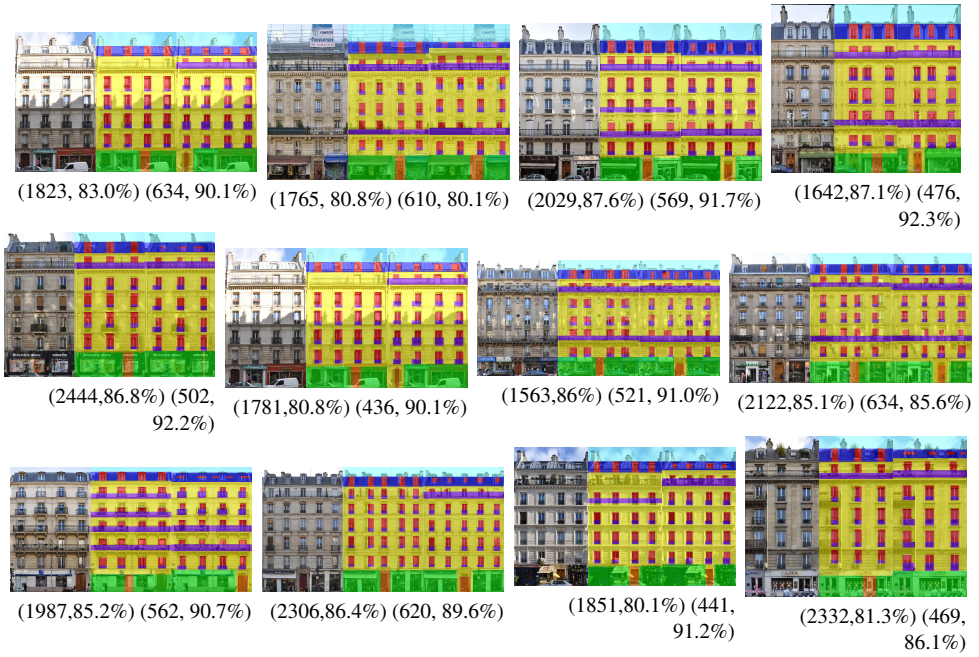


Figure 2.13: Qualitative results on ECP2011 dataset. Image (left) and segmentation using handwritten grammar (center) and learned grammar $\mathcal{G}_{\text{cl}}$ (right) are shown here along with number of episodes for convergence and segmentation accuracy.

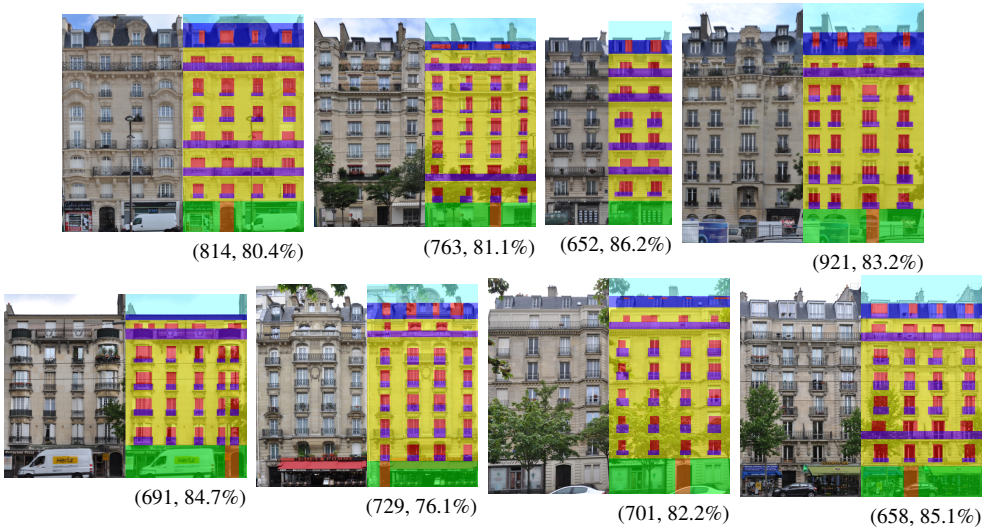


Figure 2.14: Qualitative results on ENPC2014 dataset. Image (left) and segmentation using learned grammar $\mathcal{G}_{\text{cl}}$ (right) are shown here along with number of episodes for convergence and segmentation accuracy.

CHAPTER 3

Efficient Facade Segmentation using Auto-Context

This chapter introduces a fast and efficient segmentation technique for 2D images and 3D point clouds of building facades. Facades of buildings are highly structured images and consequently most methods that have been proposed for this problem aim to make use of this strong prior information. Contrary to most prior work and chapter 2, we describe a system that is almost domain independent and consists of standard segmentation methods. Note that in the previous chapter the main goal was to learn shape priors and use them to infer the parse tree of unseen facade images. The goal of the technique presented in this chapter is to obtain accurate segmentation of building images with out using any such prior information.

3.1 Introduction

We consider the problem of segmenting building facades in an image resp. a point cloud, into different semantic classes. An example image from a common benchmark dataset for this problem is shown in Figure 3.1 along with a manual annotation. Being able to segment facades is a core component of several real world applications, including urban modeling and automatic generation of virtual cities with specific building styles. As evident from the example in Figure 3.1, 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; a door can only be found on the street-level etc.

This problem is also an interesting test-bed for general purpose segmentation methods that allow including such strong prior knowledge. As a result, it appears reasonable to assume that methods which incorporate high-level knowledge will perform well in doing automatic facade segmentation. Following this, existing facade segmentation methods use complex models and inference technique to incorporate high-level architectural knowledge for better pixel level segmentation. Some examples are Conditional Random Field (CRF) models that use higher order potential functions [Yang 2011, Tylecek 2013]. Another route are grammar-based models that include generative rules [Riemenschneider 2012, Teboul 2011c, Martinovic 2013a] and try to infer the production rules from the image evidence.

Contrary to the philosophy of existing methods, we largely ignore domain-specific knowledge. We describe a generic segmentation method that is easy to implement, has

fast test time inference, and is easily adaptable to new datasets. Our key observation is, that very good segmentation results can be achieved by pixel classifications methods that use basic image features in conjunction with auto-context features [Tu 2008]. In this work, we develop a simple and generic auto-context based framework for facade segmentation. The system is a sequence of boosted decision tree classifiers, that are stacked using auto-context [Tu 2008] features and learned using stacked generalization [Wolpert 1992]. We stack three pixel classifiers using auto-context features for images and two classifiers for 3D point clouds. Figure 3.1 shows an example segmentation result for various classification stages of our method. As can be seen in the visual result of Figure 3.1, the segmentation result is successively refined by the auto-context classifiers, from their respective previous stage result. Using pixel-level classifiers along with generic image features has the advantage of being versatile and fast compared to existing complex methods. The entire pipeline consists of established components and we consider it to be a baseline method for this task. Surprisingly, our auto-context based method, despite being simple and generic, consistently performs better or on par with existing complex methods on all the available diverse facade benchmark datasets in both 2D and 3D. Moreover, the presented approach has favourable runtime in comparison to existing approaches for facade segmentation. Therefore, this approach defines a new state-of-the-art method in terms of empirical performance. It is important to note that by proclaiming so, we are not invalidating the use of existing methods, that make of domain-knowledge, for facade segmentation. Experiments suggest that more domain-specific models would benefit from better unary predictions from our approach. 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 or point-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. We show how the pixel predictions we obtain can be used in a procedural modeling system [Teboul 2011c] that recovers production rules of the facade image. These rules are of interest in different applications and we show that an improved pixel-wise predictions directly translates into a better facade parsing result.

This chapter is organized as follows. Related work is discussed in Section 4.2, followed by a detailed description of the auto-context segmentation setup in Section 3.3. Section 3.4 contains the experimental results and in Section 3.5, an application of the system for procedural modeling is presented. We conclude in Section 3.6.

3.2 Related Work

Facade segmentation approaches can be broadly classified into two categories: *bottom-up methods* [Martinovic 2012, Tylecek 2013, Yang 2011, Cohen 2014b] that use pixel-level classifiers in combination with CRF models and *top-down methods* [Teboul 2011c, Riemenschneider 2012, Martinovic 2013a, Koziński 2014b, Koziński 2015b] that use shape grammars or a user defined shape prior. The shape grammar methods seek to parse a facade in terms of a set of production rules and element attributes thus segmenting the facade into semantic regions. The central idea is to represent the facade using a parse tree and

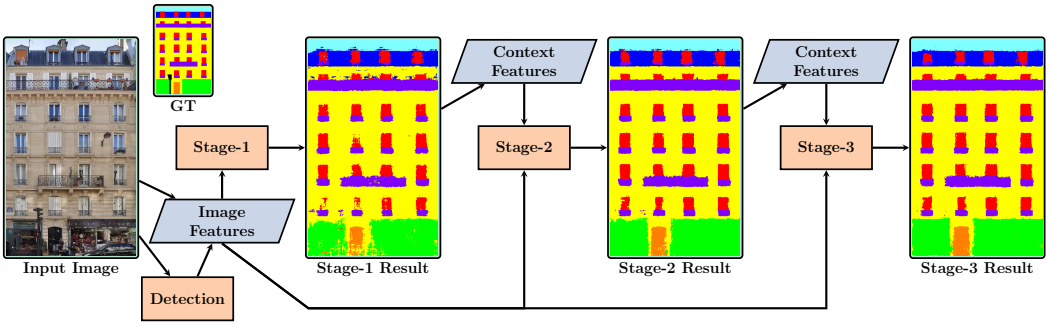


Figure 3.1: Schematic of different components in our facade segmentation pipeline with a sample facade from ECP dataset [Teboul 2010a]. 'Image' and 'Context' features correspond to features extracted on input image and previous-stage segmentation result respectively. 'Stage-n' refers to the n^{th} stage auto-context classifier. The segmentation result is successively refined by the auto-context classifiers from their respective previous stage result.

search for the derivation grammar that best matches a pixel-level classification of an 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 easily amendable to efficient inference, which often leads to inefficient and sub-optimal segmentation results. Furthermore, due to the strong prior that a grammar imposes, they are not necessarily pixel-wise accurate. As a consequence, the state-of-the-art methods in terms of pixel accuracy are dominated by the bottom-up methods, although they do not provide structured information as in a parse tree.

In [Martinovic 2012, Mathias 2015], a three-layered system is proposed. A first layer uses 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 [Cohen 2014b] to be about 2 minutes for an image of size 500 by 300. Other approaches [Yang 2011, Tylecek 2013] incorporate architectural knowledge in a single CRF framework using higher-order potential functions. The method of [Yang 2011] proposes a hierarchical CRF framework to encode inter-class location information in facades. The work of [Tylecek 2013] 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 [Cohen 2014b] 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. [Koziński 2015b] employ a user-defined shape prior (an adjacency pattern) for parsing rectified facade images and formulates parsing as a MAP-MRF problem over a pixel grid.

Recently, techniques have been introduced for facade understanding and modeling in 3D [Riemenschneider 2014, Martinovic 2015]. The 3D point cloud or meshes, that these methods operate on, are constructed using 2D images captured from multiple viewpoints. A standard way to label a 3D mesh or a point cloud, is to label all the overlapping images used for reconstructing the 3D model and then fuse the 2D predictions to obtain a consistently labeled 3D model [Ladickỳ 2010, Tighe 2010]. The work of [Riemenschneider 2014] proposed a fast technique to segment 3D facade meshes by exploiting the geometry of the reconstructed 3D model. To label a mesh face, their approach selects a single 2D image (from the set of images used for reconstruction) that best captures the semantics. The speed of this technique comes at the cost of performance. The method of [Martinovic 2015] implements a three stage approach to label point clouds of facades directly in 3D. First, features on 3D points are computed and are classified into various semantic classes. Next, facades belonging to different buildings are separated based on previously obtained semantics. Finally, weak architectural rules are applied to enforce structural priors, leading to marginal improvements in performance (0.78% IoU) compared to the inital classifier predictions.

All the discussed methods build on top of semantic label probabilities which are obtained using pixel/point classifiers. It is only after those have been obtained, that architectural constraints are taken into account. In the system we describe in this paper, 2D or 3D segmentations are obtained only using image or point cloud, and auto-context features without resorting to any domain specific architectural constraints. As a result, several above mentioned domain-specific approaches would benefit from using the segmentation label probabilities obtained with the proposed domain-independent approach.

The closest to our work are [Fröhlich 2012] and [Gatta 2014], which also proposed auto-context based methods for facade segmentation. [Fröhlich 2012] 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, [Gatta 2014] proposed to use the local Taylor coefficients computed from the posterior at different scales as auto-context. Although [Fröhlich 2012] and [Gatta 2014] are conceptually similar, the method we propose uses different low-level features, auto-context features and learning techniques acheiving better performance on several benchmark datasets.

3.3 Auto-Context Segmentation

We propose an architecture that combines standard segmentation methods into a single framework. Boosted decision trees are stacked with the use of auto-context [Tu 2008] features from the second layer onward. This system is then trained using stacked generalization [Wolpert 1992]. We will describe the ingredients in the following, starting with the segmentation algorithm (Sec. 3.3.1), the feature representation for images (Sec. 3.3.2) and for point-clouds (Sec. 3.3.3), auto-context features (Sec. 3.3.4), and the training procedure (Sec. 3.3.5).

Given a point cloud or an image I, the task of semantic segmentation is to classify every

point or pixel i into one of C classes $c_i \in \{1, \ldots, C\}$. During training, we have access to a set of N images each with a variable number of points/pixels: $(I_i^j, c_i^j), j = 1, \ldots, N$. 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, data dependent features $f_i \in \mathbb{R}^{D_f}$ that are derived from the spatial and color observations in a point cloud or image, and auto-context features $a_i \in \mathbb{R}^{D_a}$ based on the prediction results from previous stages.

3.3.1 Model Architecture

Our system consists of a sequence of classifiers as suggested in [Tu 2008]. A schematic overview of the pipeline is depicted in Figure 3.1. At every stage t, the classifier has access to the image and to predictions of all earlier stages. Formally, at stage $t > 1$ and at each pixel i, a classifier F^t maps image (I) and auto-context features (a_i) to a probability distribution P^t of the pixel class assignments

$$F^t\left(f_i(I), a_i(P^{t-1})\right) \mapsto P^t(c_i|I), \forall i. \tag{3.1}$$

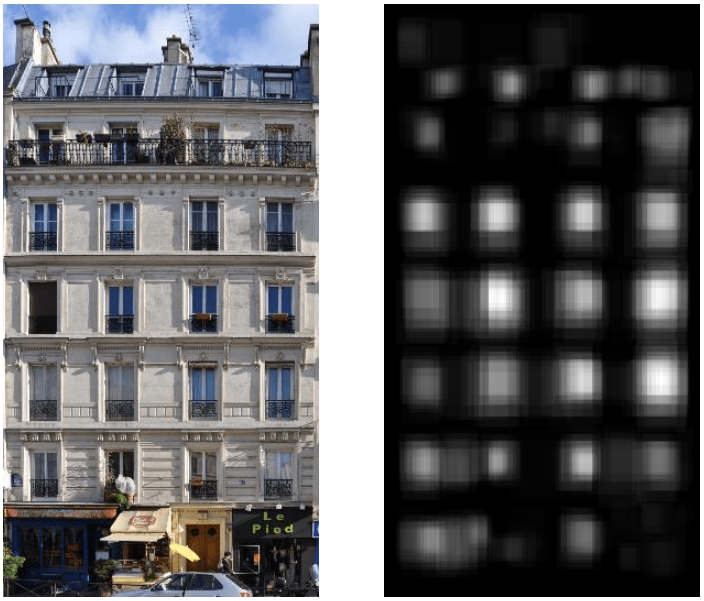
For pixel classifier F^t, we use boosted decision trees that store conditional distributions at their leaf nodes. In general the output of F^t need not be a distribution. The first stage $t = 1$ depends only on the features $F^1(f_i(I))$ derived directly from the image or the point cloud.

This architecture is a conceptually easy and 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 Figure 3.1. 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.

3.3.2 Image Features

As image features, we computed 17 TextonBoost filter responses [Shotton 2006], location information, RGB color information, dense Histogram of Oriented Gradients [Dalal 2005], Local Binary Pattern features [Ojala 2002], and all filter averages over image rows and columns at each pixel. These are computed using the DARWIN [Gould 2012b] toolbox.

In addition to the above generic segmentation features, we include detection scores for some specific objects. Following [Martinovic 2012], we use detectors for windows as well as doors. Whereas [Martinovic 2012] 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 [Dollár 2009] for which a toolbox is available [Dollár 2014]. 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 a good recall. Figure 3.2 shows an example window detection output for a sample facade image. The detection feature is of course a problem dependent



(a) Facade (b) Detection

Figure 3.2: (a) A facade and (b) its window detection scores. Bright and dark regions correspond to high and low detection scores respectively.

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 extremely weak and a generic detection system has been used to obtain it. Moreover, door and window classes are common to any architecture of facades. In the end, 761 low-level image features coupled with 2 door and window detection features make a total of 763 feature values at each pixel.

3.3.3 Point Cloud Features

We use the same set of features as [Martinovic 2015] to describe a point cloud. The features inlude the mean RGB color values, their corresponding LAB values, the estimated normal at the 3D point, the spin image descriptor [Johnson 1999], the height of a point above an estimated groud plane, the depth of the point from an estimated facade plane and the inverse height of the point which is the distance from the uppermost point of the facade in the direction of the gravity vector. The combination of all these features form a 132-dimensional vector for every point.

3.3.4 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 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 $14C + 1$, where C is the number of classes.

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 $4C$) .

In the case of point clouds, we used only the class probabilities and the entropy of the class probabilities as auto-context features. So, for point clouds the size of the auto-context features is $C + 1$.

3.3.5 Stacked Generalization

We train the sequence of classifiers using stacked generalization [Wolpert 1992]. The training data is split in M folds and at each stage, M different models are trained using data from $M-1$ folds, with one fold held out. The M 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 additional classifier is trained using the entire training data (all M folds), this one is used during test time inference. In our experiments, to segment 2D images we divide the training set into four folds ($M = 4$) and for 3D point clouds, we do not use the stacked generalization ($M = 1$) due to

the availability of fewer training points. Consequently, we used three classification stages for 2D images and only two classification stages for 3D point clouds as we observe that the performance levels out after that.

Thus, instead of using single classifier in each stage, the auto-context features are computed using predictions from different classifiers, different also from the classifier that will be used at test time. The reason for this procedure is to obtain features that are not computed on training predictions and thus avoid to overfit to the data. This procedure is a standard strategy and is found to be stable and well performing in many scenarios, e.g. [Gehler 2009].

For training and testing, we used the DARWIN toolbox [Gould 2012b]. The maximum tree-depth of each boosted decision tree classifier is set to two and we used a maximum of 200 boosting rounds.

3.4 Experiments

We evaluate the auto-context pipeline on all seven benchmark datasets that are available for the problem of facade segmentation. For all datasets except LabelMeFacade [Frohlich 2010] and RueMonge2014 [Riemenschneider 2014] datasets, we report five fold cross-validation results, the standard protocol used in the literature. One fold cross-validation is done for LabelMeFacade and RueMonge2014 datasets as the train and test data splits are pre-specified for these datasets. We compare against all recent best performing methods.

As performance measures, we use the overall pixel-wise classification accuracy, the accuracy averaged over the classes and the intersection over union (IoU) score, popularized by the VOC segmentation challenges [Everingham 2010]. 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 as done in the literature concerning these datasets. The three measures are defined as follows in terms of false positives (FP), true positives (TP), and false negatives (FN).

- *Overall 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 $p < 0.01$.

3.4.1 Datasets

ECP Dataset. The ECP dataset [Teboul 2010a] consists of 104 rectified facade images of Hausmannian architectural buildings from Paris. For five-fold cross validation, we ran-

domly 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 [Riemenschneider 2012] has 50 facade images of various architectures (Classicism, 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 [Korc 2009] 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 [Tylecek 2013], 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 [Frohlich 2010], this dataset has 100 training and 845 testing facade images taken from LabelMe segmentation dataset [Russell 2008]. Facades in this dataset are highly irregular with a lot of diversity across images.

ENPC Art-deco dataset. This dataset, first used in [Gadde 2016b], contains 79 rectified and cropped facade images of the Art-deco style buildings from Paris. Similar to the ECP dataset, the images in this dataset are segmented into seven semantic classes.

RueMonge2014 Dataset. This dataset, introduced in [Riemenschneider 2014], is aimed towards providing a benchmark for 2D and 3D facade segmentation, and procedural modelling. It consists of 428 high-resolution and multi-view images of facades following the Haussmanian style architecture, a reconstructed point cloud, a reconstructed mesh and a framework to evaluate segmentation results. Three tasks were proposed on this dataset in [Riemenschneider 2014, Martinovic 2015] *. The first task is the standard image labeling task where each pixel has to be assigned a semantic label. The second task is the mesh labeling task where a semantic label has to be assigned to each face of a given mesh. And the third task is the point cloud labeling task where a semantic label has to be assigned to each point in the point cloud. For each of the tasks, fixed splits in training and testing sets are pre-defined. The ground-truth labeling consists of seven semantic classes, same as in the ECP dataset.

3.4.2 Results on Single-view Segmentation

The empirical results on different datasets are summarized in Tables 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, where ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. In addition to the pixel-wise predictions of the auto-context classifiers, we

*http://varcity.eu/3dchallenge/

	Door	Shop	Balcony	Window	Wall	Sky	Roof	**Average**	**Overall**	**IoU**
[Martinovic 2012]	60	86	71	69	**93**	97	73	78.4	85.06	-
[Cohen 2014b]-1	79	94	91	85	90	97	90	89.4	90.82	-
[Cohen 2014b]-2	**82**	96	**92**	**87**	88	96	**93**	**90.6**	90.34	-
[Mathias 2015]	58	**97**	81	76	90	94	87	83.4	88.1	-
[Koziński 2015b]	79	**97**	91	**87**	90	97	91	90.3	**91.3**	-
ST1	76.2	87.6	85.8	77.0	91.9	97.3	86.5	86.04	88.86	75.25
ST2	79.2	90.4	89.4	80.7	92.3	**97.9**	88.1	88.28	**90.49**	78.62
ST3	80.4	91.6	89.4	81.8	92.4	**98.0**	88.1	88.79	**90.81**	79.32
PW1	77.9	90.5	86.4	77.1	**93.0**	**97.8**	88.4	87.31	90.02	77.57
PW2	79.9	92.1	89.3	81.0	**93.0**	**98.1**	89.3	88.94	**91.12**	79.88
PW3	81.3	93.2	89.3	82.3	**92.9**	**98.2**	89.2	89.49	**91.42**	**80.54**
[Teboul 2011c]	47	88	58	62	82	95	66	71.1	74.71	-
[Martinovic 2013a]	50	81	49	66	80	91	71	69.7	74.82	-
ST3+ [Teboul 2011c]	64.2	90.1	71.4	75.6	92.5	96.1	77.2	81.0	85.2	72.4

Table 3.1: 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 [Teboul 2011c] when using the output ST3 result are reported on the last row.

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 [Gould 2012b]. 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.

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 1.9% 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 ST3>ST2>ST1 on the ECP, CMP, Art-deco and LabelMeFacade datasets and ST3=ST2>ST1 on eTrims and Graz datasets. The auto-context 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 the ECP dataset is that the overall accuracy of ST3, with 90.8%, is comparable with the reported 91.3% from the current best performing method [Koz-

Class	[Martinovic 2012][Cohen 2014b]		[Gatta 2014]	Auto Context (AC)			AC + Potts Model		
				ST1	ST2	ST3	PW1	PW2	PW3
Building	91	91	84	90.3	90.5	90.9	**92.7**	**92.5**	**92.5**
Car	69	70	51	63.3	74.8	72.4	69.4	**79.1**	76.6
Door	18	18	**73**	62.7	62.3	63.6	66.0	63.6	65.3
Pavement	33	33	55	43.0	46.5	47.1	43.1	**48.6**	**48.8**
Road	55	57	81	78.2	82.3	80.3	80.9	**84.7**	82.1
Sky	93	97	**99**	97.6	**98.5**	**98.6**	98.2	**98.8**	**98.9**
Vegetation	89	90	92	91.1	92.1	92.3	92.4	**92.8**	**92.9**
Window	**74**	71	78	65.9	67.1	68.4	65.6	66.5	68.2
Average	65.3	65.9	66.4	74.01	76.78	76.7	76.04	**78.32**	78.14
Overall	83.16	83.84	83.40	84.68	85.95	86.12	86.39	**87.29**	**87.29**
IoU	-	-	-	58.7	61.26	61.48	61.49	63.39	**63.54**

Table 3.2: Segmentation results of various methods on eTRIMS dataset. ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. PW1, PW2, and PW3 refer to a Potts model using ST1, ST2, and ST3, respectively, as unaries. Published results are shown for comparisons.

ński 2015b]. The CRF-Potts model (PW3) achieves higher accuracies than the method of [Koziński 2015b], making it the (although only marginally) highest published result on the ECP dataset. The results of the auto-context classifier are significantly higher on the other datasets, except in the Graz dataset, when compared to the methods of [Riemenschneider 2012,Cohen 2014b,Martinovic 2012,Tylecek 2013,Fröhlich 2012,Nowozin 2014, Gatta 2014, Koziński 2015b]. On the eTRIMS, CMP and LabelMeFacades datasets, even the first stage classifier produces better predictions than the previous approaches. The methods of [Riemenschneider 2012, Cohen 2014b, Martinovic 2012, Tylecek 2013, Koziński 2015b] all include domain knowledge in their design. For example, the system of [Cohen 2014b] is a sequence of dynamic programs that is used to include specific domain knowledge, such as that balconies are below windows, that only one door exists, or that elements like windows are rectangular segments. [Koziński 2015b] uses hand-written adjacency patterns to limit the possible transitions between different states based on the semantic classes. On the ECP dataset, the authors of [Cohen 2014b] and [Koziński 2015b] observe respectively, an improvement of about 4% (personal communication) and 1.3% over their unary classifiers accuracy, we conjecture they also may improve the predictions of ST3.

The methods of [Riemenschneider 2012,Cohen 2014b,Martinovic 2012,Tylecek 2013, Fröhlich 2012,Nowozin 2014] 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 [Riemenschneider 2012, Cohen 2014b, Martinovic 2012,

Class	[Tylecek 2013]	Auto Context			AC + Potts Model		
		ST1	ST2	ST3	PW1	PW2	PW3
Background	58	67.1	71.8	**72.6**	68.0	**72.6**	73.1
Facade	73	74.6	75.3	75.2	**80.5**	79.9	79.3
Window	61	71.6	76.1	77.0	74.1	77.4	**78.1**
Door	**54**	37.9	45.5	47.0	39.6	46.4	48.7
Cornice	41	39.1	47.5	**49.6**	40.0	48.3	**50.1**
Sill	27	21.1	32.8	**36.2**	16.9	30.3	34.6
Balcony	46	31.6	44.1	46.7	31.6	45.2	**48.1**
Blind	**48**	22.7	35.8	40.1	19.5	34.7	39.9
Deco	**24**	10.4	13	13.8	6.1	10.0	11.4
Molding	54	63.2	65.4	66.5	64.2	66.0	**67.2**
Pillar	**25**	5.71	11.2	13.6	1.33	7.72	9.78
Shop	**59**	40.9	45.6	45.6	42.8	46.7	46.8
Average	47.5	40.50	47.00	**48.65**	40.38	47.1	**48.92**
Overall	60.3	61.83	65.47	66.24	64.46	67.48	**68.08**
IoU	-	29.26	34.46	35.86	30.67	36.02	**37.47**

Table 3.3: Segmentation results of various methods on CMP dataset. ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. PW1, PW2, and PW3 refer to a Potts model using ST1, ST2, and ST3, respectively, as unaries. Published results are shown for comparisons.

Class	Auto Context (AC)			AC + Potts Model			[Riemenschneider 2012]
	ST1	ST2	ST3	PW1	PW2	PW3	
Door	57.3	62.4	**62.7**	57.3	**62.8**	**63**	41
Window	78.2	81.2	**81.5**	77.8	80.6	80.9	60
Wall	94.9	94.7	94.9	**95.8**	**95.6**	**95.8**	84
Sky	87.4	**91.2**	**90.5**	87.7	**91.4**	**90.6**	**91**
Average	79.47	82.40	82.42	79.65	**82.61**	**82.56**	69
Overall	90.18	91.02	91.16	90.78	**91.53**	**91.68**	78
IoU	71.25	73.31	73.25	72.49	**74.45**	**74.39**	58

Table 3.4: Segmentation results of various methods on Graz dataset. ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. PW1, PW2, and PW3 refer to a Potts model using ST1, ST2, and ST3, respectively, as unaries. Published results are shown for comparisons. The method of [Riemenschneider 2012] parses the image into a lattice representation and is not trying to maximize pixel accuracy results.

Tylecek 2013] is required.

On all the datasets, we observe 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

Class	[Fröhlich 2012]	[Nowozin 2014]	Auto Context(AC)			AC + Potts Model		
			ST1	ST2	ST3	PW1	PW2	PW3
Building	-	-	87.7	88.1	88.2	**92.7**	91.8	92.1
Car	-	-	47.1	53.6	54.8	51.1	57.0	**58.2**
Door	-	-	**6.52**	6.03	5.12	2.61	3.22	1.71
Pavement	-	-	24	**25.3**	**24.6**	22.0	24.2	23.3
Road	-	-	80.3	82.1	84.5	85.3	85.1	**87.6**
Sky	-	-	86.2	87.2	87.4	88.3	**88.6**	**88.9**
Vegetation	-	-	53.3	**57.5**	**57.6**	53.4	**58.1**	**57.9**
Window	-	-	20.3	22.6	**25.4**	13.0	16.9	19.5
Various	-	-	19.9	**20.6**	**21.0**	11.6	12.2	12.1
Average	**56.61**	-	47.26	49.22	49.84	46.68	48.56	49.04
Overall	67.33	71.28	71.52	72.9	73.46	74.1	74.62	**75.23**
IoU	-	35.96	37.01	38.69	39.36	37.74	38.96	**39.57**

Table 3.5: Segmentation results of various methods on labelmeFacades dataset. ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. PW1, PW2, and PW3 refer to a Potts model using ST1, ST2, and ST3, respectively, as unaries. Published results are shown for comparisons.

Class	Auto Context (AC)			AC + Potts Model			[Gadde 2016b]
	ST1	ST2	ST3	PW1	PW2	PW3	
Door	64.9	69.4	**69.7**	65.0	69.6	**69.7**	59
Shop	93.9	95.3	95.4	94.5	**95.9**	**95.9**	88
Balcony	70.2	76.8	77.7	70.7	77.2	**78.0**	63
Window	75.1	79.9	81.2	75.2	79.8	**81.4**	66
Wall	90.7	91.3	91.2	**92.5**	92.4	92.3	84
Sky	96.5	97.1	97.5	96.9	97.4	**97.7**	92
Roof	74.4	77.9	77.3	76.9	**79.4**	78.7	58
Average	80.83	83.97	84.28	81.67	84.54	**84.83**	72.9
Overall	85.88	88.08	88.29	86.96	88.79	**89.03**	78
IoU	68.32	72.03	72.39	69.85	73.16	**73.51**	58

Table 3.6: Segmentation results of various methods on Art-deco dataset. ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. PW1, PW2, and PW3 refer to a Potts model using ST1, ST2, and ST3, respectively, as unaries. Published results are shown for comparisons.

the auto-context classifiers. 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 Figure 3.3 for different datasets and in Figure 3.5 for different classification stages. All the results for

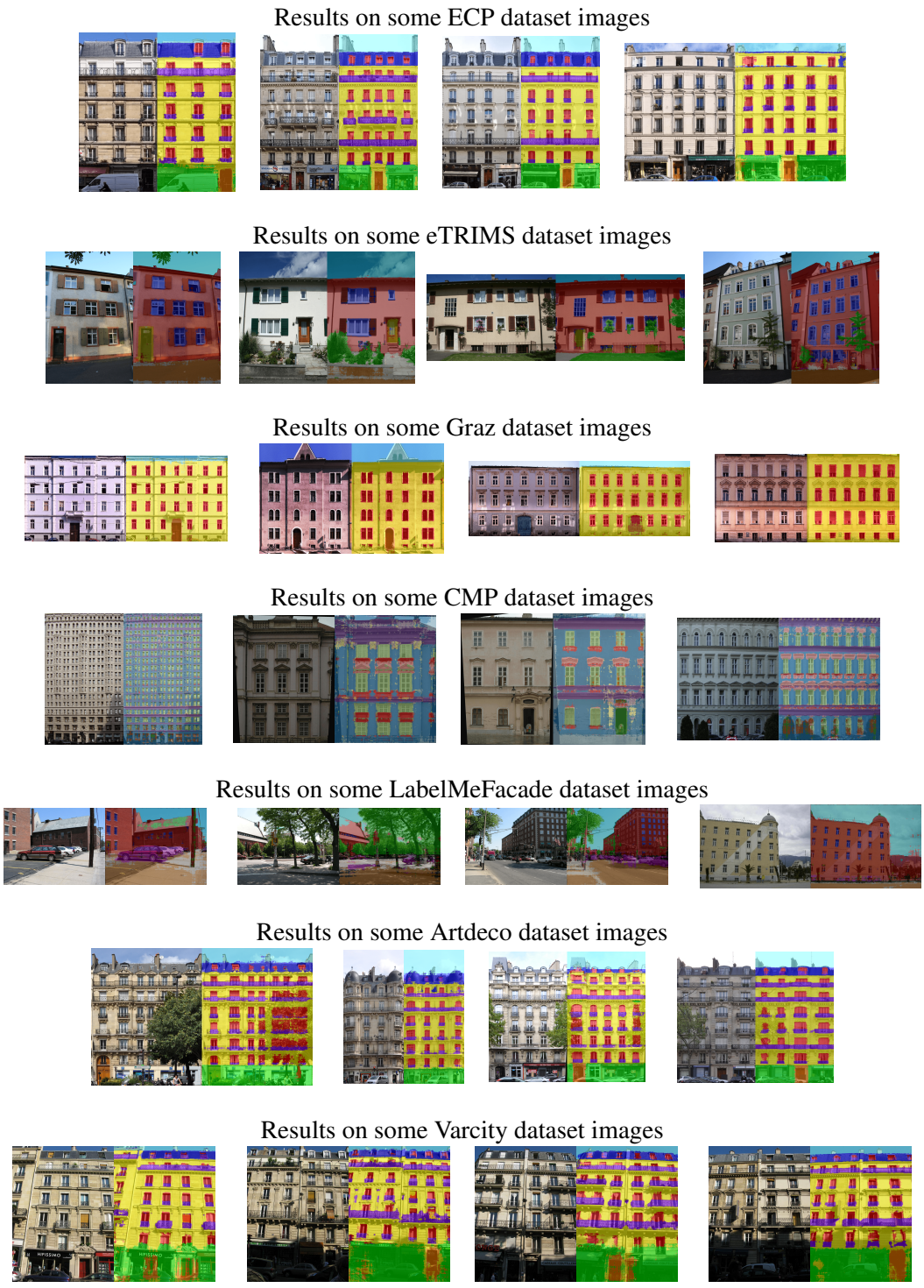


Figure 3.3: Sample qualitative results on various datasets (more results at the end of the chapter)

Method	Features	AC-Features	ST1	ST2	ST3	PW	[Martinovic 2012]	[Cohen 2014b]
Time (s)	3.0	0.6	+0.04	+0.64	+0.64	+24	110	2.8

Table 3.7: 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.

comparison purpose are publicly available at http://fs.vjresearch.com.

The average runtime of the system on 2D images is summarized in Table. 3.7. 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). All the extracted features are computed sequentailly one after the other. Table. 3.7 also indicates that the proposed appraoch has favourable runtime in comparison to existing prominent methods.

3.4.3 Results on Multi-view Segmentation

In this section, we present semantic segmentation results of multi-view scenes using 2D images and 3D point clouds from the RueMonge2014 dataset [Riemenschneider 2014]. The dataset details are already presented in Sec. 3.4.1. Similar to [Martinovic 2015], we show results using only 2D images, only 3D point cloud and combined 2D+3D approaches for the tasks of image labeling and point cloud labeling. Additionally, we present results for the mesh labeling task by projecting the image segmentation results and point cloud segmentation results on to the mesh faces.

Here, we first apply the proposed auto-context technique, separately on 2D images and 3D point cloud to measure the relative gains. We find improvements in both 2D and 3D, similar to the results on the single view datasets. For 2D images, the performance levels out after three stages, while for 3D point clouds, the performance levels out after two stages. This can be due to fewer autocontext features and less training data for 3D point clouds. The publicly available 3D point cloud in the RueMonge2014 dataset has only 290196 training points and 276529 test points.

Next, we compare our results with the best performing techniques on these tasks. All the results are summarized in Tables 3.8 and 3.9. For all our experiments in 2D, we used the specified training set of 119 images along with ground-truth, and evaluated on 202 test images. For the image labeling task, using only 2D images, we perform better (by +2.93% IoU) and faster (by at least a factor of 3) compared to [Martinovic 2012]. Note that our runtimes shown in Tables 3.8 and3.9 are computed by applying the proposed technique sequentially on all the 202 test images. This can be easily parallelized by performing the segmentation on all the test images in parallel. Similar improvements in performance are observed in the image labelling task when using only 3D point cloud data as well. Here, the image labeling is performed by back-projecting the semantically labelled 3D point cloud onto 2D images.

	Image Labeling Task	Door	Shop	Balcony	Window	Wall	Sky	Roof	IoU	Runtime (min)
2D	[Martinovic 2012]	-	-	-	-	-	-	-	57.53	379
	ST1	17.8	57.6	67.0	53.0	73.7	78.4	59.5	58.13	25
	ST2	17.3	56.7	68.5	53.6	74.5	79.2	58.7	58.37	51
	ST3	16.4	58.5	68.9	54.0	74.9	79.8	59.7	58.89	77
	PW1	18.9	58.9	69.0	55.5	75.2	79.1	60.8	59.63	57
	PW2	18.5	58.5	70.8	56.3	76.2	79.9	61.1	60.18	83
	PW3	17.3	59.7	71.4	56.6	76.4	80.5	61.5	**60.46**	**109**
3D	[Riemenschneider 2014]	-	-	-	-	-	-	-	41.34	15
	[Martinovic 2015]	-	-	-	-	-	-	-	**53.22**	**21**
	ST1	0	90.1	60.1	33.5	64.4	80.0	51.8	54.29	15
	ST2	0	89.3	57.7	39.9	71.6	79.6	56.0	56.30	15
	PW1	0	90.1	61.8	34.4	65.9	80.7	53.0	55.13	15
	PW2	0	89.0	57.8	39.1	72.2	81.5	59.8	57.04	16
2D+3D	[Martinovic 2015]	-	-	-	-	-	-	-	**61.95**	**404**
	ST3(2D) + ST1(3D)	25.2	85.8	78.9	69.2	87.4	94.8	78.2	61.95	85
	ST4	60.1	74.5	84.9	71.7	84.3	95.6	74.0	61.16	114
	ST3(2D) + ST1(3D) + PW	23.7	89.0	76.1	66.7	88.7	95.3	81.4	**62.68**	117
	PW4	61.3	76.6	85.1	72.5	85.2	96.4	76.0	**62.64**	146

Table 3.8: Segmentation results of various methods for the image labeling task on Varcity dataset. ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. PW1, PW2, and PW3 refer to a Potts model using ST1, ST2, and ST3, respectively, as unaries. Published results are shown for comparisons. The runtimes shown here, in minutes, are the times taken to segment the entire dataset and includes the feature extraction and classification. Note: Just for this dataset, we report the class-wise IoU scores.

For the point cloud labeling task, using either only 2D or 3D data, we observe better segmentation results compared to [Martinovic 2015]. We note that all improvements are obtained without explicitly modeling structural priors or any other type of domain knowledge. [Martinovic 2015] proposed to use weak architectural principles in 3D on top of initial segmentations that come from a simple classifier. Such weak architectural principles have shown only a marginal improvement of +0.15% in IoU. In contrast, when using only 3D data, the ST2 result of the proposed auto-context technique performs better by +1.5% in IoU, and improves even further by applying a pairwise Potts model. Note that, in this case, we use exactly the same 3D features as [Martinovic 2015] for the first stage classification. This renders the results of weak architectural principles in [Martinovic 2015] and the proposed auto-context features directly comparable. We also obtain favorable runtime. It takes 8 minutes to enforce the weak achitectural principles but less than a minute to apply another stage of auto-context.

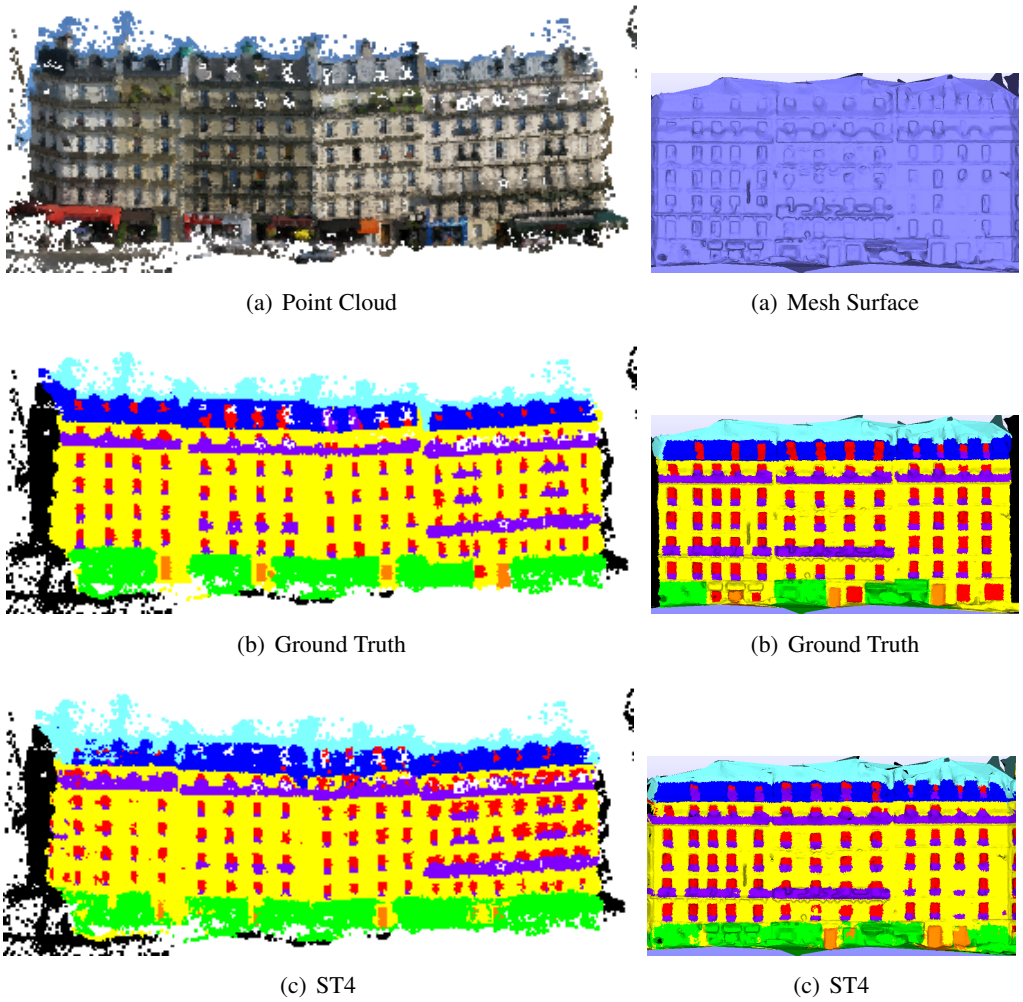
Next, similar to [Martinovic 2015], we combine the segmentation results of 2D im-

	Point Cloud Labeling Task	Door	Shop	Balcony	Window	Wall	Sky	Roof	IoU	Runtime (min)
2D	[Martinovic 2012]	-	-	-	-	-	-	-	55.39	390
	ST1 + Majority vote	18.3	53.5	57.5	52.6	75.9	74.1	57.6	55.66	25
	ST2 + Majority vote	17.6	54.6	63.0	56.2	77.7	70.1	60.5	57.10	51
	ST3 + Majority vote	19.0	53.4	65.6	56.9	78.6	71.3	62.7	58.21	77
	PW1 + Majority vote	19.4	53.7	60.4	54.4	77.3	73.9	57.7	56.68	57
	PW2 + Majority vote	17.8	54.8	64.3	57.4	78.5	70.1	59.5	57.49	83
	PW3 + Majority vote	19.6	53.8	66.7	58.1	79.3	70.5	62.4	**58.63**	**109**
3D	[Riemenschneider 2014]	-	-	-	-	-	-	-	42.32	15
	[Martinovic 2015] (RF+CRF)	-	-	-	-	-	-	-	52.09	15
	[Martinovic 2015] (RF+CRF+WR)	-	-	-	-	-	-	-	52.24	23
	ST1	12.4	59.5	40.8	51.4	76.4	64.1	54.6	51.33	15
	ST2	29.6	60.9	44.6	50.5	75.0	62.9	51.8	53.60	15
	PW1	11.3	60.1	41.8	51.7	76.8	65.2	55.8	51.80	15
	PW2	30.3	61.3	45.4	51.3	75.5	63.5	52.6	**54.25**	**16**
2D+3D	[Martinovic 2015](RF+L1+CRF)	-	-	-	-	-	-	-	60.05	317
	[Martinovic 2015](RF+L1+CRF+WR)	-	-	-	-	-	-	-	**60.83**	**325**
	ST3(2D) + ST1(3D)	18.0	60.2	68.3	60.8	82.5	71.0	63.5	60.59	77
	ST4	29.2	65.8	66.8	60.4	81.7	71.5	63.7	**62.71**	**78**
	ST3(2D) + ST1(3D) + PW	18.4	60.4	68.7	60.9	82.7	71.2	63.7	60.85	**78**
	PW4	28.2	66.3	67.0	60.6	81.8	71.8	63.8	**62.78**	**79**

Table 3.9: Segmentation results of various methods for the task of point cloud labeling on Varcity dataset. ST1, ST2, and ST3 correspond to the classification stages in the auto-context method. PW1, PW2, and PW3 refer to a Potts model using ST1, ST2, and ST3, respectively, as unaries. Published results are shown for comparisons. The runtimes shown here, in minutes, are the times taken to segment the entire dataset and includes the feature extraction and classification. Note: Just for this dataset, we report the class-wise IoU scores.

ages and 3D point clouds for further improving the performance ('2D+3D' in Tables 3.8 and 3.9). For this, we accumulate the ST3 and ST2 results of 2D images and 3D point cloud respectively. Further applying auto-context (ST4) on the accumulated unaries boosts the IoU performance by another 2.1% in labeling the point cloud, while it levels out in labeling the images. Additionally, applying a pairwise Potts CRF model similar to [Martinovic 2015] to enforce smoothness, further increases the IoU performance by 0.7% and 0.2% in labeling the images and point cloud respectively. In summary, as evident in Tables 3.8 and 3.9, better performance than existing state-of-the-art approaches is obtained in both image and 3D point cloud labelling, while being 2-3 times faster. A visual result of 3D point cloud segmentation is shown in Figure 3.4.

Finally, we compare our results with existing approaches on the mesh labeling prob-



(a) Point Cloud (a) Mesh Surface

(b) Ground Truth (b) Ground Truth

(c) ST4 (c) ST4

Figure 3.4: Sample visual results for the point cloud and mesh labeling tasks on the Rue-Monge2014 dataset.

lem. To label meshes, the 2D image segmentation results are projected onto the mesh faces. 2D segmentation results obtained with only 2D images and with both 2D+3D data are used for mesh labeling. See Table 3.10 for quantitative results and Figure 3.4 for a visual result. Again we observe similar improvements with auto-context classifiers. Our simple majority voting scheme to project the semantics from 2D images to 3D mesh shows that we perform significantly better (by +17.4% IoU) in comparison to the existing approach from [Riemenschneider 2014]. However this comes at a cost of runtime (6 $\times$ slower). The result of earlier stages of our auto-context technique still performs better, 'ST1' for example, by +15.9% with a comparable runtime. In [Riemenschneider 2014], 3D information is used to reduce redundant evaluations, thereby achieving faster runtime.

	Mesh Labeling Task	Door	Shop	Balcony	Window	Wall	Sky	Roof	IoU	Runtime (min)
2D	[Riemenschneider 2014]	-	-	-	-	-	-	-	41.92	15
	ST1 + Majority vote	16.5	54.8	65.0	57.2	78.7	70.0	62.4	57.81	25
	ST2 + Majority vote	19.6	53.9	67.0	57.8	79.5	71.0	64.1	58.97	51
	ST3 + Majority vote	15.2	54.2	68.0	58.7	79.8	70.4	64.3	58.65	77
	PW1 + Majority vote	16.3	54.9	65.7	57.9	79.2	69.8	61.7	57.94	57
	PW2 + Majority vote	20.4	54.2	67.7	58.6	80.1	70.4	63.7	**59.28**	**83**
	PW3 + Majority vote	14.5	54.1	69.0	59.7	80.2	70.2	64.2	58.83	109

Table 3.10: Segmentation results of various methods for the task of mesh labeling on Varcity dataset. ST1, ST2, and ST3 correspond to the classification stages using only image features in the auto-context method. PW1, PW2, and PW3 refer to a Potts model using ST1, ST2, and ST3, respectively, as unaries. Published results are shown for comparisons. The runtimes shown here, in minutes, are the times taken to segment the entire dataset and includes the feature extraction and classification. Note: Just for this dataset, we report the class-wise IoU scores.

3.5 Procedural Modeling

A pixel-wise classification of a facade might not be the desired input for some applications. This fact motivated shape grammar methods [Riemenschneider 2012, Ripperda 2006b, Teboul 2011c, Martinovic 2013a] that parse the facade into a high-level structured representation. 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 similar facades etc. We apply the parsing method of [Teboul 2011c], 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. We refer the reader to [Teboul 2011c] for more details about the approach. The results are shown in the last three columns of Table 3.1. 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. Figure 3.5 shows a sample visual result of various classification stages and the parsing result obtained with ST3 + [Teboul 2011c].

3.6 Conclusion

The segmentation method that we described in this chapter, part of which has been presented at conference WACV [Jampani 2015] and extended in a journal submission [Gadde 2016a], is a framework of established and proven components. It is easy to implement, fast at test-time, and outperforms all previous approaches on all published facade segmentation

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

Figure 3.5: (a) Sample facade images 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 [Teboul 2011c] using ST3 result.

datasets. It is also the fastest method amongst all those that we compared against. The runtime is dominated by feature computation, which is amenable 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. This applies to both 2D (images) and 3D (point clouds) data. For the ECP dataset, a Potts-CRF further improves the performance but this comes at the expense of a severe increase in runtime. Further, the proposed technique can be applied independently to either 2D or 3D data and also to combined 2D+3D models. For the point

cloud labeling task, on RueMonge2014 dataset, applying autocontext on the combined 2D+3D improves the IoU performance by 1.9%.

The proposed auto-context classifier raises the bar when it comes to absolute performance. Contrary to the popular belief in this domain, it largely ignores domain knowledge, but still the performs better than all the methods that include prior information in some form, for example relationship between balconies and windows, etc. We believe that it is important to evaluate methods in terms of a relative improvement over strong pixel classifier baselines. In order to facilitate a fair comparison of previous and future work, we release all code that has been used to obtain the reported results along with all predictions for easy comparison[†].

Recently, densely connected CRFs has shown significant improvements on top of pixel classifiers. Very fast inference techniques have been proposed [Krähenbühl 2012] for such densely connected graphs. A obvious extension, would be to use such densely connected CRFs with auto-context classifier. Another extension would be to train in an end-to-end fashion. Recently convolutional neural networks have shown significant improvement for the task of semantic segmentation [Chen 2014a]. Auto-context can be employed here in an end-to-end to setting. Another advantage here would be that the auto-context features can be learned directly without having the need for hand-crafted features.

Finally, we present more qualitative results in Figures 3.6, 3.8, 3.7, 3.9, 3.10, 3.11, 3.12, and 3.13. The images have been selected based on the absolute overall pixel-accuracy of ST3 and include images with the (i) highest, (ii) average, and (iii) lowest performance.

[†]http://fs.vjresearch.com

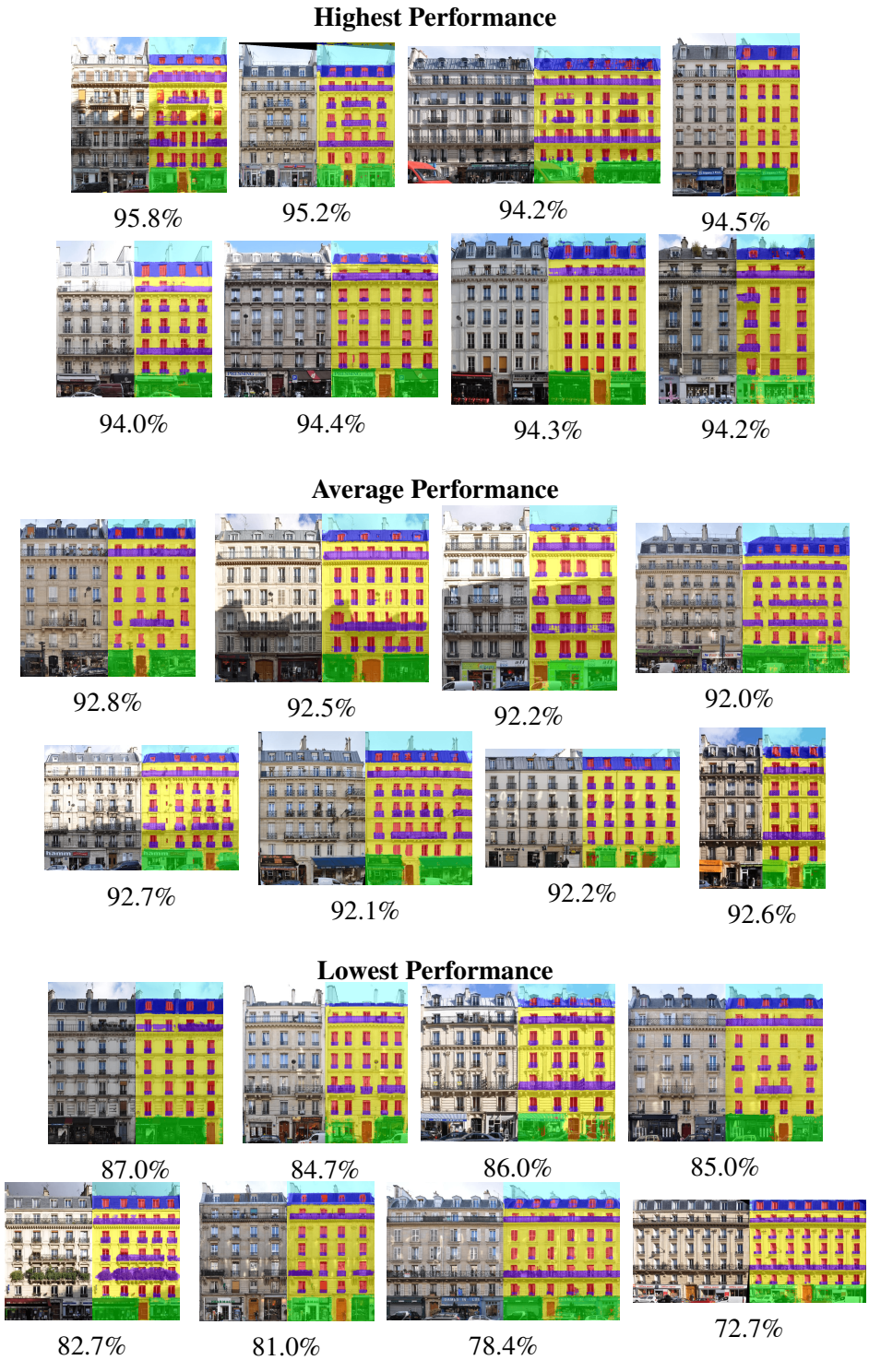


Figure 3.6: Qualitative results on ECP dataset images along with overall pixel accuracy (Stage-3 results).

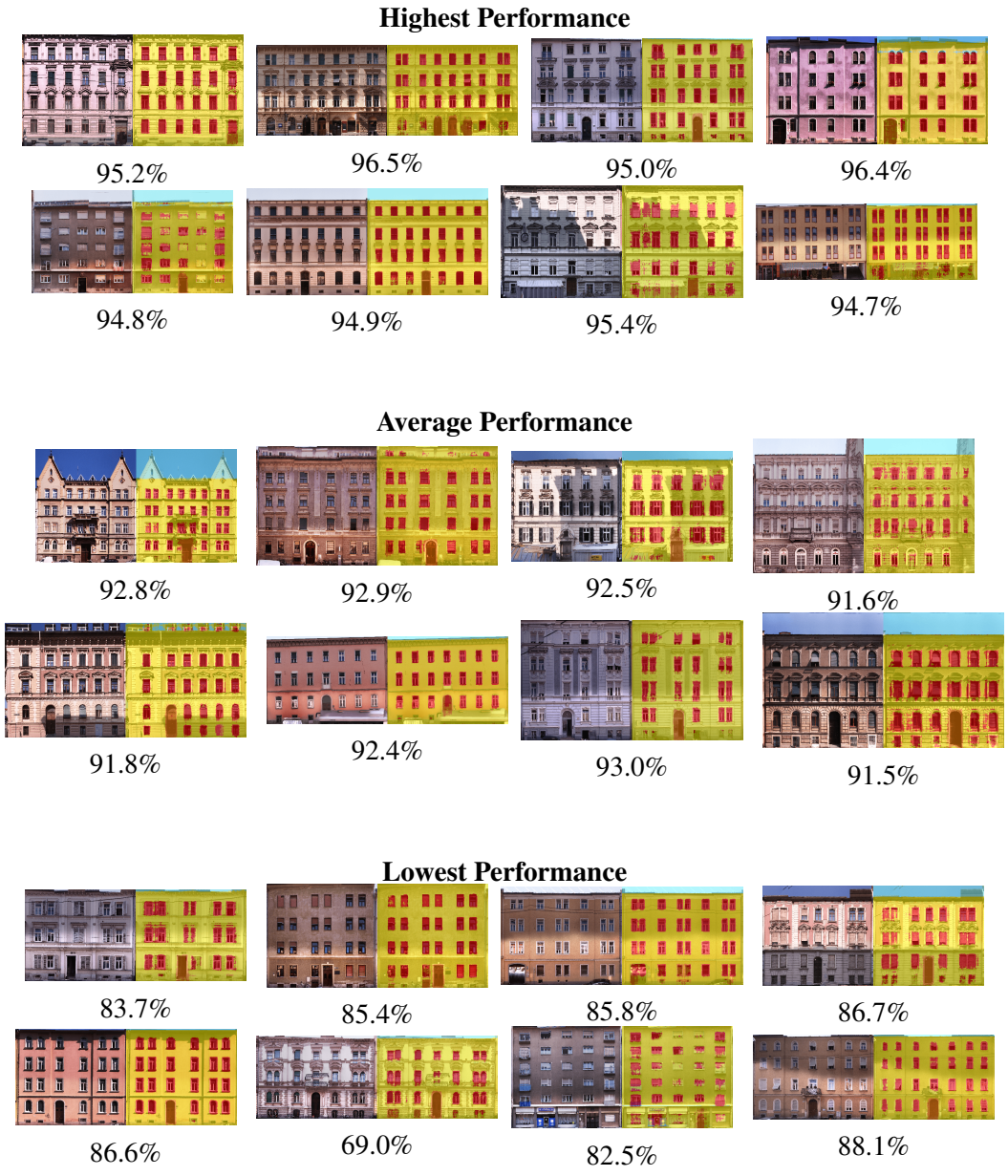


Figure 3.7: Qualitative results on Graz dataset images along with overall pixel accuracy (Stage-3 Results).

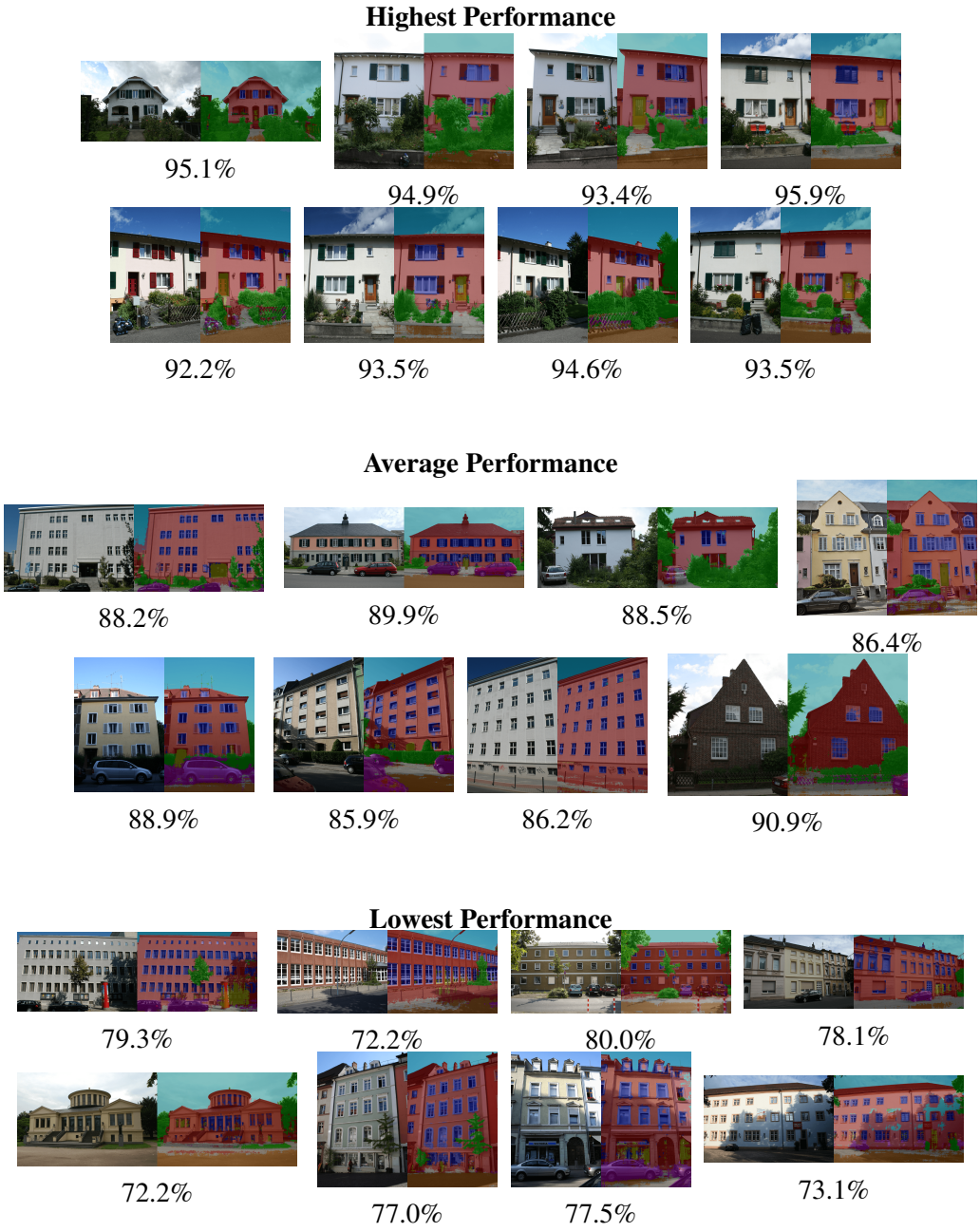


Figure 3.8: Qualitative results on eTRIMS dataset images along with overall pixel accuracy (Stage-3 results).

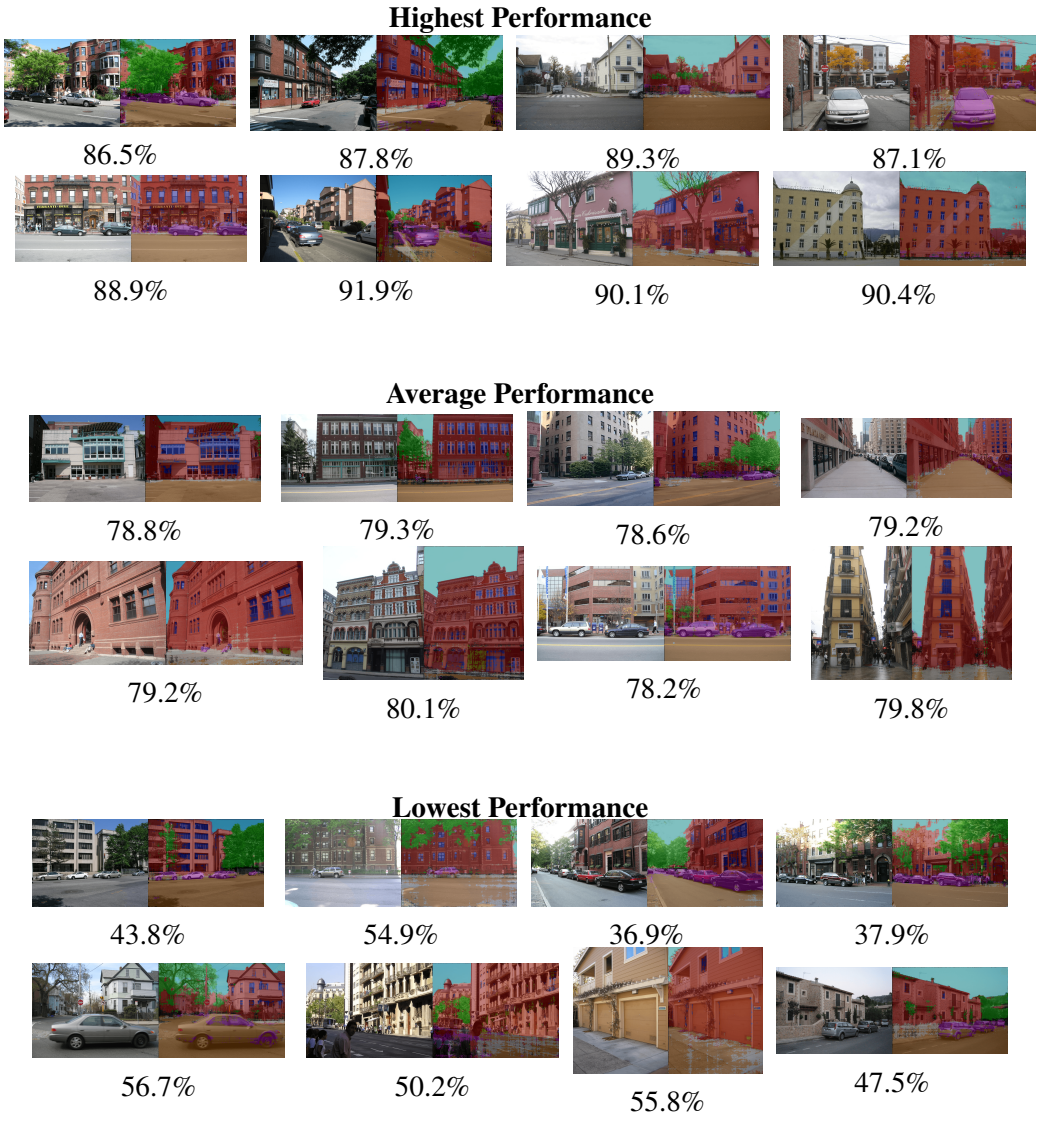


Figure 3.9: Qualitative results on labelmeFacades dataset images along with overall pixel accuracy (Stage-3 results).

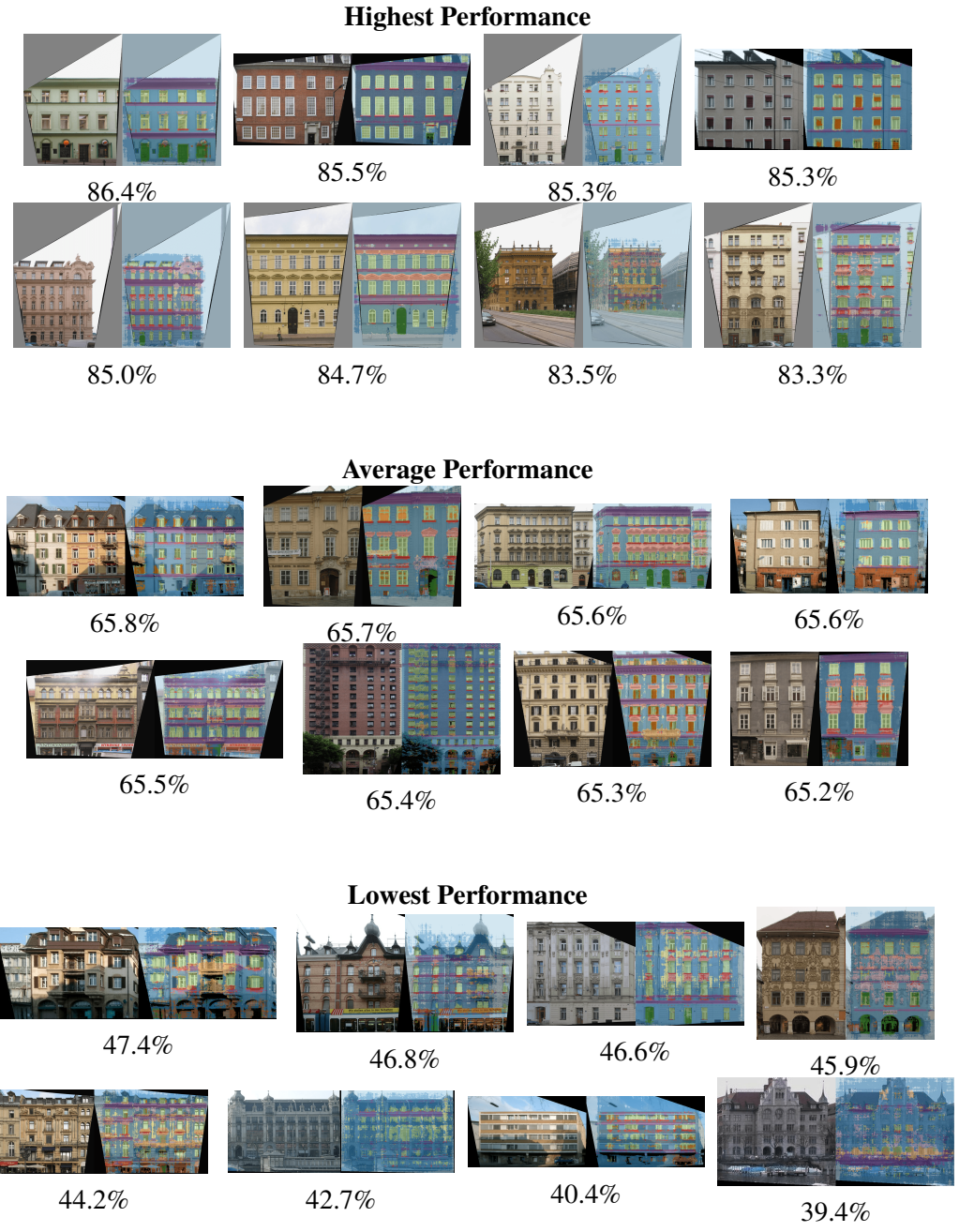


Figure 3.10: Qualitative results on CMP dataset images along with overall pixel accuracy (Stage-3 results).

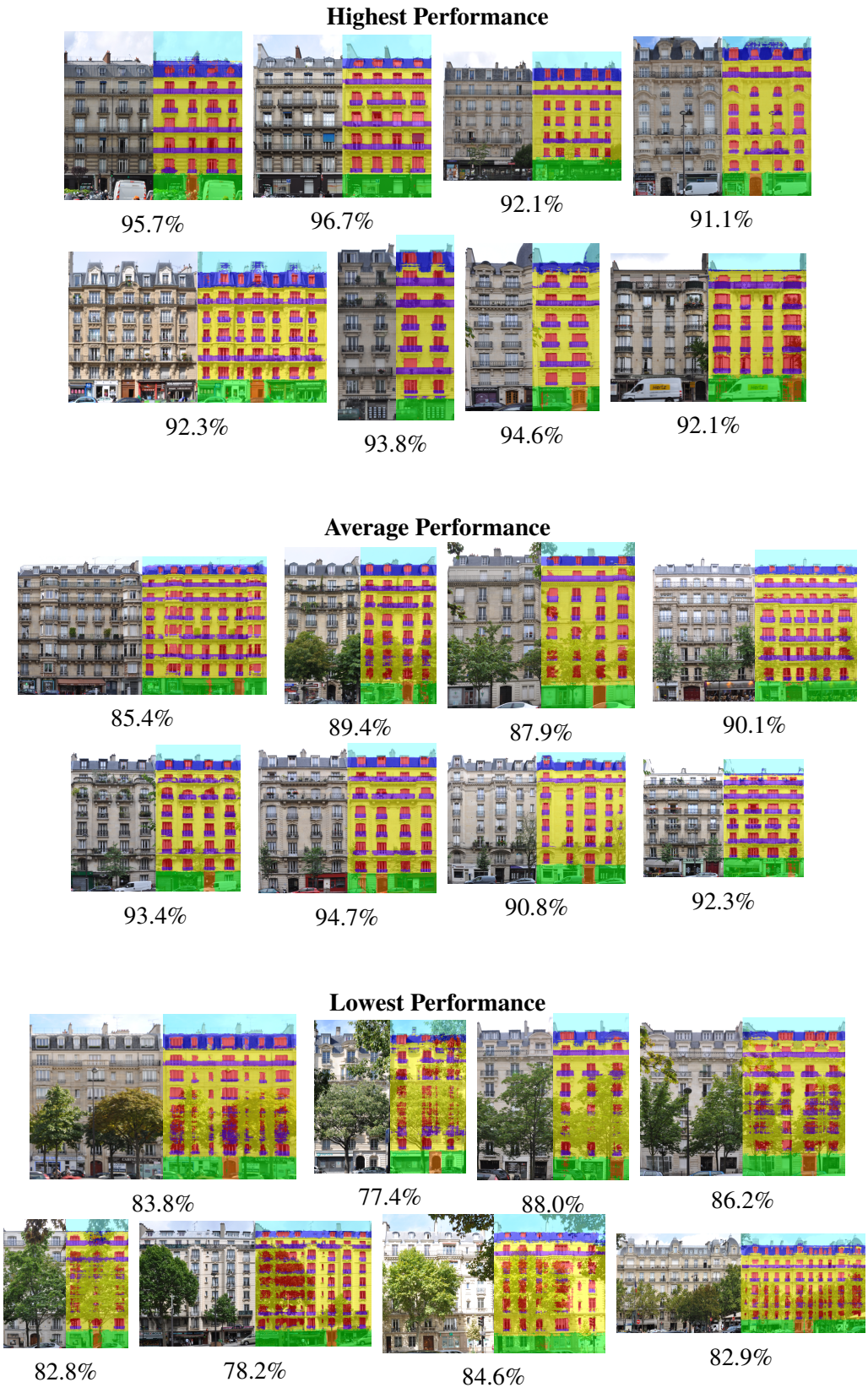
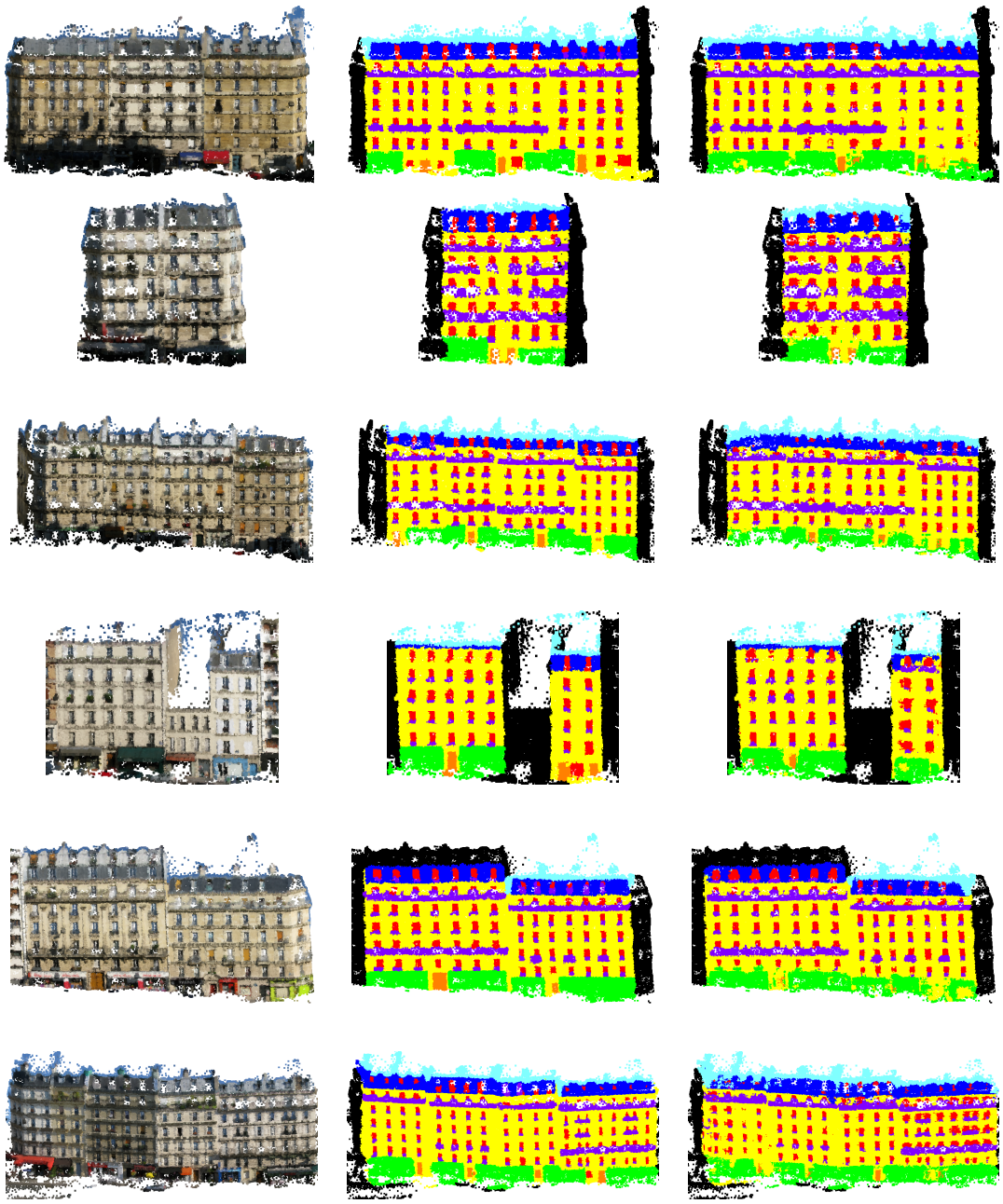
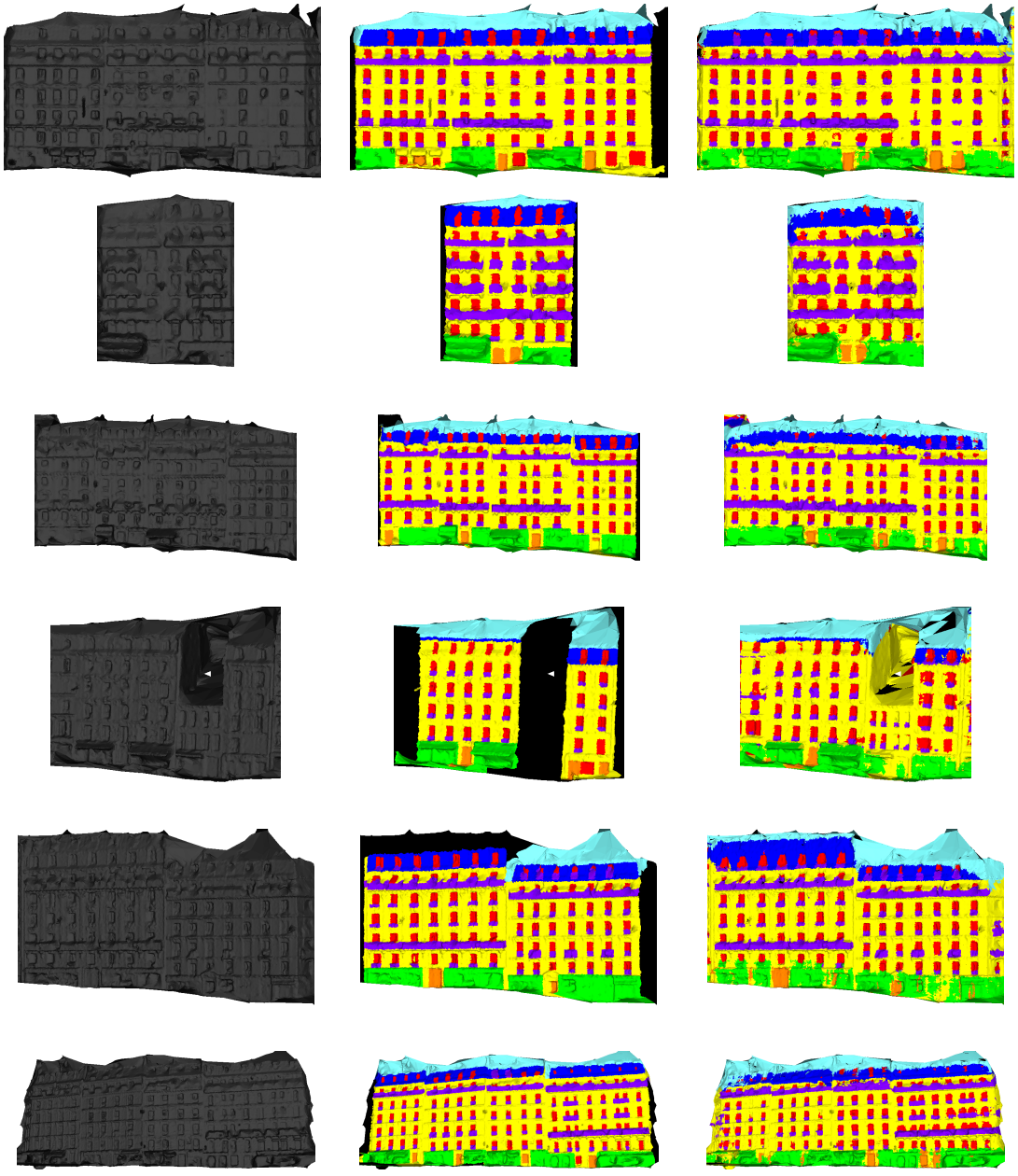


Figure 3.11: Qualitative results on Artdeco dataset images along with overall pixel accuracy (Stage-3 results).



(a) Facade (b) Ground Truth (c) ST4

Figure 3.12: RueMonge2014 point cloud labeling results.



(a) Facade (b) Ground Truth (c) ST4

Figure 3.13: RueMonge2014 mesh labeling results.

CHAPTER 4

Superpixel Convolutional Networks using Bilateral Inceptions

Recently, convolutional neural networks have significantly improved the performance of semantic segmentation among several computer vision tasks. In this chapter we propose a CNN architecture for semantic image segmentation. We introduce a new "bilateral inception" module that can be inserted in existing CNN architectures and performs bilateral filtering, at multiple feature-scales, between superpixels in an image. The feature spaces for bilateral filtering and other parameters of the module are learned end-to-end using standard backpropagation techniques. The bilateral inception module addresses two issues that arise with general CNN segmentation architectures. First, this module propagates information between (super) pixels while respecting image edges, thus using the structured information of the problem for improved results. Second, the layer recovers a full resolution segmentation result from the lower resolution solution of a CNN.

4.1 Introduction

Given an image $\mathcal{I} = (x_1, \ldots, x_N)$ with N pixels x_i the task of semantic segmentation is to infer a labeling $Y = (y_1, \ldots, y_N)$ with a label $y_i \in \mathcal{Y}$ for every pixel. This problem can be naturally formulated as a structured prediction problem $g : \mathcal{I} \rightarrow Y$. Empirical performance is measured by comparing Y to a human labeled Y^* via a loss function $\Delta(Y, Y^*)$, *e.g.*, with the Intersection over Union (IoU) or pixel-wise Hamming Loss.

A direct way to approach this problem would be to ignore the structure of the output variable Y and train a classifier that predicts the class membership of the center pixel of a given image patch. This procedure reduces the problem to a standard multi-class classification problem and allows the use of standard learning algorithms. The resulting classifier is then evaluated at every possible patch in a sliding window fashion (or using coarse-to-fine strategies) to yield a full segmentation of the image. With high capacity models and large amounts of training data this approach would be sufficient, given that the loss decomposes over the pixels. Such a per-pixel approach ignores the relationship between the variables $(y_1, \ldots, y_N)$, which are not i.i.d. since there is an underlying common image. Therefore, besides learning discriminative per-pixel classifiers, most segmentation approaches further encode the output relationship of Y. A dominating approach is to use Conditional Random Fields (CRF) [Lafferty 2001], which allows an elegant and principled way to combine single pixel predictions and shared structure through unary, pairwise and higher-order factors.

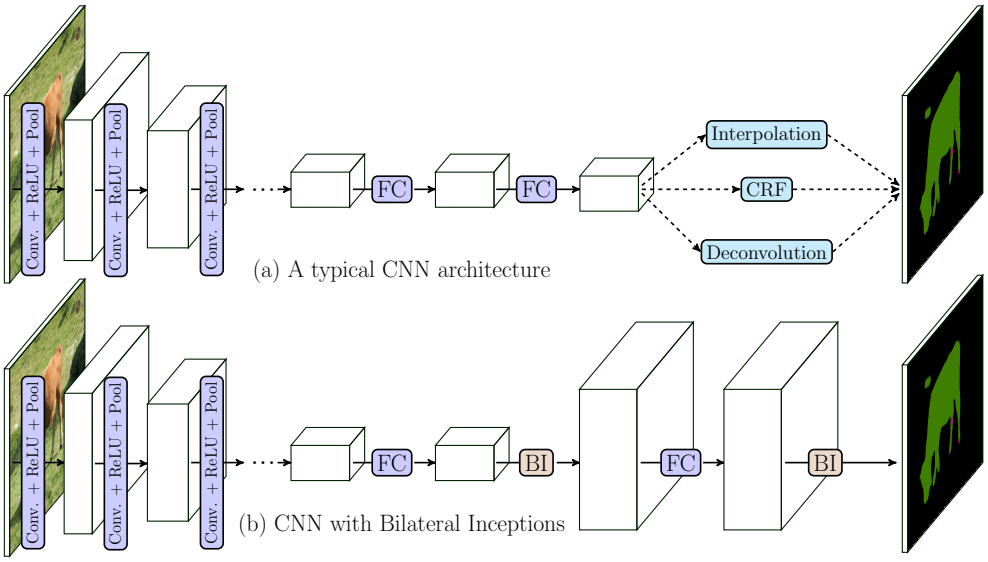


Figure 4.1: **Illustration of CNN layout.** We insert the *Bilateral Inception (BI)* modules between the *FC* (1×1 convolution) layers found in most networks thus removing the necessity of further upscaling algorithms. Bilateral Inception modules also propagate information between distant pixels based on their spatial and color similarity and work better than other label propagation approaches.

What relates the outputs $(y_1, \ldots, y_N)$? The common hypothesis that we use in this work could be summarized as: *Pixels that are spatially and photometrically similar are more likely to have the same label.* Particularly if two pixels x_i, x_j are close in the image and have similar RGB values, then their corresponding labels y_i, y_j will most likely be the same. The most prominent example of spatial similarity encoded in a CRF is the Potts model (Ising model for the binary case). The work of [Krähenbühl 2012] described a densely connected pairwise CRF (DenseCRF) that includes pairwise factors encoding both spatial *and* photometric similarity. The DenseCRF has been used in many recent works on image segmentation which find also empirically improved results over pure pixel-wise CNN classifiers [Chen 2014a, Bell 2015, Zheng 2015, Chen 2015].

In this work, we implement the above-mentioned hypothesis of photometrically similar and near-by pixels share common labels, by designing a new "Bilateral Inception" (BI) module that can be inserted before/after the last 1×1 convolution layers (which we refer to as 'FC' layers - 'Fully-Connected' in the original image classification network) of the standard segmentation CNN architectures. The bilateral inception module does edge-aware information propagation across different spatial CNN units of the previous FC layer. Instead of using the spatial grid-layout that is common in CNNs, we incorporate the superpixel-layout for information propagation. The information propagation is performed using standard bilateral filters with Gaussian kernels, at different feature scales. This construction is inspired by [Szegedy 2014, Lin 2014a]. Feature spaces and other parameters of the modules can be learned end-to-end using standard backpropagation techniques. The application of

superpixels reduces the number of necessary computations and implements a long-range edge-aware inference between different superpixels. Moreover, since superpixels provides an output at the full image resolution it removes the need for any additional post-processing step.

We introduce BI modules in the CNN segmentation models of [Chen 2014a,Zheng 2015, Bell 2015]. See Fig. 4.1 for an illustration. This achieves better segmentation results on all three datasets we experimented with than the proposed interpolation/inference techniques of DenseCRF [Bell 2015, Chen 2014a] while being faster. Moreover, the results compare favorably against some recently proposed dense pixel prediction techniques [Chen 2014a, Zheng 2015]. As illustrated in Fig. 4.1, the BI modules provides an alternative approach to commonly used up-sampling and CRF techniques.

4.2 Related Work

The literature on semantic segmentation is large and therefore we will limit our discussion to those works that perform segmentation with CNNs and discuss the different ways to encode the output structure.

A natural combination of CNNs and CRFs is to use the CNN as unary potential and combine it with a CRF that also includes pairwise or higher-order factors. For instance [Chen 2014a, Bell 2015] observed large improvements in pixel accuracy when combining a DenseCRF [Krähenbühl 2012] with a CNN. The mean-field steps of the DenseCRF can be learned and back-propagated as noted by [Domke 2013] and implemented by [Zheng 2015, Jampani 2016a,Li 2014,Schwing 2015] for semantic segmentation and [Kiefel 2014] for human pose estimation. The works of [Chen 2014b, Lin 2015, Liu 2015] use CNNs also in pairwise and higher-order factors for more expressiveness. The recent work of [Chen 2015] replaced the costly DenseCRF with a faster domain transform performing smoothing filtering while predicting the image edge maps at the same time. Our work was inspired by DenseCRF approaches but with the aim to replace the expensive mean-field inference. Instead of propagating information across unaries obtained by a CNN, we aim to do the edge-aware information propagation across *intermediate* representations of the CNN. Experiments on different datasets indicate that the proposed approach generally gives better results in comparison to DenseCRF while being faster.

A second group of works aims to inject the structural knowledge in intermediate CNN representations by using structural layers among CNN internal layers. The deconvolution layers model from [Zeiler 2010] are being widely used for local propagation of information. They are computationally efficient and are used in segmentation networks, for *e.g.* [Long 2014]. They are however limited to small receptive fields. Another architecture proposed in [He 2014] uses spatial pyramid pooling layers to max-pool over different spatial scales. The work of [Ionescu 2015] proposed specialized structural layers such as normalized-cut layers with matrix back-propagation techniques. All these works have either fixed local receptive fields and/or have their complexity increasing exponentially with longer-range pixel connections. Our technique allows for modeling long-range (super-) pixel dependencies without compromising the computational efficiency. A very recent

work [Yu 2015] proposed the use of dilated convolutions for propagating multi-scale contextual information among CNN units.

A contribution of this work is to define convolutions over superpixels by defining connectivity among them. In [He 2015], a method to use superpixels inside CNNs has been proposed by rearranging superpixels based on their features. The technique proposed here is more generic and alleviates the need for rearranging superpixels. A method to filter irregularly sampled data has been developed in [Bruna 2013] which may be applicable to superpixel convolutions. The difference being that their method requires a pre-defined graph structure for every example/image separately while our approach directly works on superpixels. We experimented with Isomap embeddings [Tenenbaum 2000] of superpixels but for speed reasons opted for the more efficient kernels presented in this work. The work of [Mostajabi 2014] extracted multi-scale features at each superpixel and perform semantic segmentation by classifying each superpixel independently. In contrast, we propagate information across superpixels by using bilateral filters with learned feature spaces.

Another core contribution of this work is the end-to-end trained bilateral filtering module. Several recent works on bilateral filtering [Barron 2015b, Barron 2015a, Kiefel 2015, Jampani 2016a] back-propagate through permutohedral lattice approximation [Adams 2010], to either learn the filter parameters [Kiefel 2015, Jampani 2016a] or do optimization in the bilateral space [Barron 2015b, Barron 2015a]. Most of the existing works on bilateral filtering use pre-defined feature spaces. In [Campbell 2013], the feature spaces for bilateral filtering are obtained via a non-parametric embedding into an Euclidean space. In contrast, by explicitly computing the bilateral filter kernel, we are able to back-propagate through features, thereby learning the task-specific feature spaces for bilateral filters through integration into end-to-end trainable CNNs.

4.3 Superpixel Convolutional Networks

We first formally introduce superpixels in Sec. 4.3.1 before we describe the bilateral inception modules in Sec. 4.3.2.

4.3.1 Superpixels

The term *superpixel* refers to a set of n_i pixels $S_i = \{t_1, \ldots, t_{n_i}\}$ with $t_k \in \{1, \ldots, N\}$ pixels. We use a set of M superpixels $S = \{S_1, \ldots, S_M\}$ that are disjoint $S_i \cap S_j = \emptyset, \forall i, j$ and decompose the image, $\cup_i S_i = \mathcal{I}$.

Superpixels have long been used for image segmentation in many previous works, *e.g.* [Gould 2014, Gonfaus 2010, Nowozin 2010, Mostajabi 2014], as they provide a reduction of the problem size. Instead of predicting a label y_i for every pixel x_i, the classifier predicts a label y_i per superpixel S_i and extends this label to all pixels within. A superpixel algorithm can pre-group pixels based on spatial and photometric similarity, reducing the number of elements and also thereby regularizing the problem in a meaningful way. The downside is that superpixels introduce a quantization error whenever pixels within one segment have different ground truth label assignments.

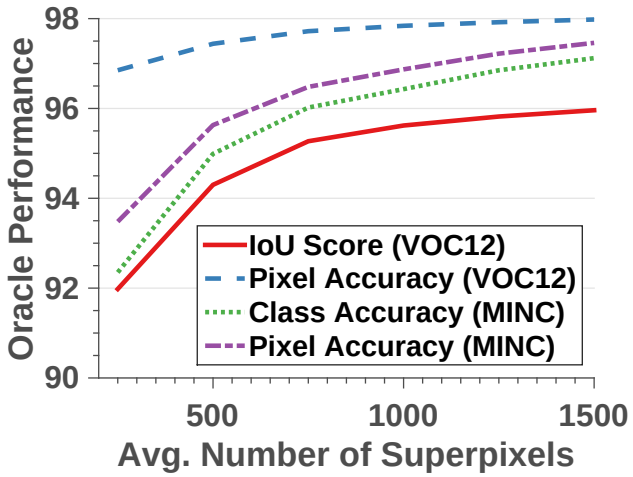


Figure 4.2: **Superpixel Quantization Error.** Best achievable segmentation performance with a varying number of superpixels on Pascal VOC12 segmentation [Everingham 2012] and MINC material classification [Bell 2015] datasets.

Figure 4.2 shows the superpixel quantization effect with the best achievable performance as a function in the number of superpixels, on two different segmentation datasets: PascalVOC [Everingham 2012] and Materials in Context [Bell 2015]. We find that the quantization effect is small compared to the current best segmentation performance. Practically, we use the SLIC superpixels [Achanta 2012] for their runtime and [Dollár 2013] for their lower quantization error to decompose the image into superpixels. For details of the algorithms, please refer to the respective papers. We use publicly-available real-time GPU implementation of SLIC, called gSLICr [Ren 2015], which runs at over 250Hz per second. And the publicly available Dollar superpixels code [Dollár 2013] computes a super-pixelization for a 400×500 image in about 300ms using an Intel Xeon 3.33GHz CPU.

4.3.2 Bilateral Inceptions

Next, we describe the *Bilateral Inception Module* (BI) that performs Gaussian Bilateral Filtering on multiple scales of the representations within a CNN. The BI module can be inserted in between layers of existing CNN architectures.

Bilateral Filtering: We first describe the Gaussian bilateral filtering, the building block of the BI module. A visualisation of the necessary computations is shown in Fig. 4.3. Given the previous layer CNN activations $\mathbf{z} \in \mathbb{R}^{P \times C}$, that is P points and C filter responses. With $\mathbf{z}_c \in \mathbb{R}^P$ we denote the vector of activations of filter c. Additionally we have for every point j a feature vector $\mathbf{f}_j \in \mathbb{R}^D$. This denotes its spatial position ($D = 2$, not necessarily a grid), position and RGB color ($D = 5$), or others. Separate from the input points with features $F_{in} = \{\mathbf{f}_1, \ldots, \mathbf{f}_P\}$ we have Q output points with features F_{out}. These can be the same set of points, but also fewer ($Q < P$), equal ($Q = P$), or more ($Q > P$) points. For example we can filter a 10×10 grid ($P = 100$) and produce the result on a 50×50 grid ($Q = 2500$) or vice versa.

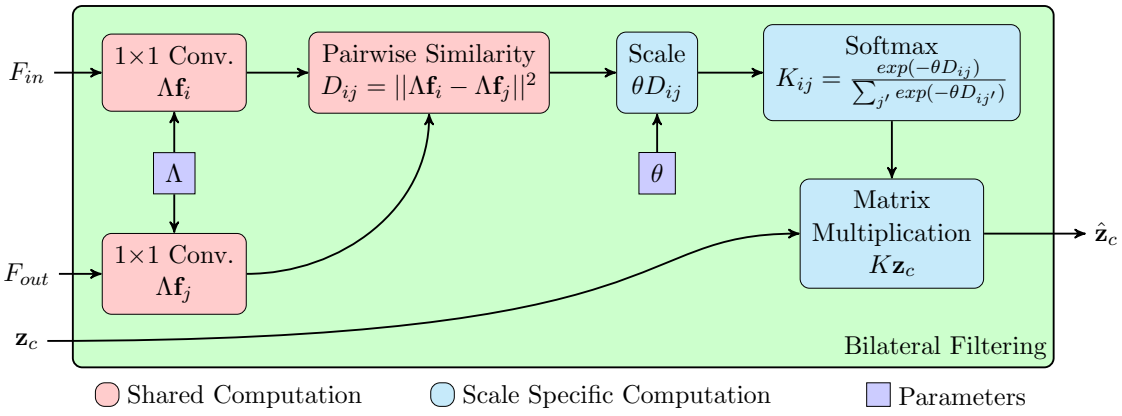


Figure 4.3: **Computation flow of the Gaussian Bilateral Filtering.** We implemented the bilateral convolution with five separate computation blocks. Λ and θ are the free parameters.

The bilateral filtered result will be denoted as $\hat{\mathbf{z}} \in \mathbb{R}^{Q \times C}$. We apply the same Gaussian bilateral filter to every channel c separately. A filter has two free parameters: the filter specific scale $\theta \in \mathbb{R}_+$ and the global feature transformation parameters $\Lambda \in \mathbb{R}^{D \times D}$. For Λ, a more general scaling could be applied using more features or a separate CNN. Technically the bilateral filtering amounts to a matrix-vector multiplication $\forall c$:

$$\hat{\mathbf{z}}_c = K(\theta, \Lambda, F_{in}, F_{out})\mathbf{z}_c, \tag{4.1}$$

where $K \in \mathbb{R}^{Q \times P}$ and values for $f_i \in F_{out}, f_j \in F_{in}$:

$$K_{i,j} = \frac{\exp(-\theta \|\Lambda \mathbf{f}_i - \Lambda \mathbf{f}_j\|^2)}{\sum_{j'} \exp(-\theta \|\Lambda \mathbf{f}_i - \Lambda \mathbf{f}_{j'}\|^2)}. \tag{4.2}$$

From a kernel learning terminology, K is nothing but a Gaussian Gram matrix and it is symmetric if $F_{in} = F_{out}$. We implemented this filtering in Caffe [Jia 2014] using different layers as depicted in Fig. 4.3. While approximate computations of $K\mathbf{z}_c$ exist and have improved runtime [Adams 2010, Paris 2006, Gastal 2011, Adams 2009], we chose an explicit computation of K due to its small size. Our implementation makes use of GPU and the intermediate pairwise similarity computations are re-used across different modules. The entire runtime is only a fraction of the CNN runtime, but of course applications to larger values of P and Q would require aforementioned algorithmic speed-ups.

Bilateral Inception Module: The *bilateral inception module* (BI) is a weighted combination of different bilateral filters. We combine the output of H different filter kernels K, with different scales $\theta^1, \ldots, \theta^H$. All kernels use the same feature transformation Λ which allows for easier pre-computation of pairwise difference and avoids an over-parametrization of the filters. The outputs of different filters $\hat{\mathbf{z}}^h$ are combined linearly to

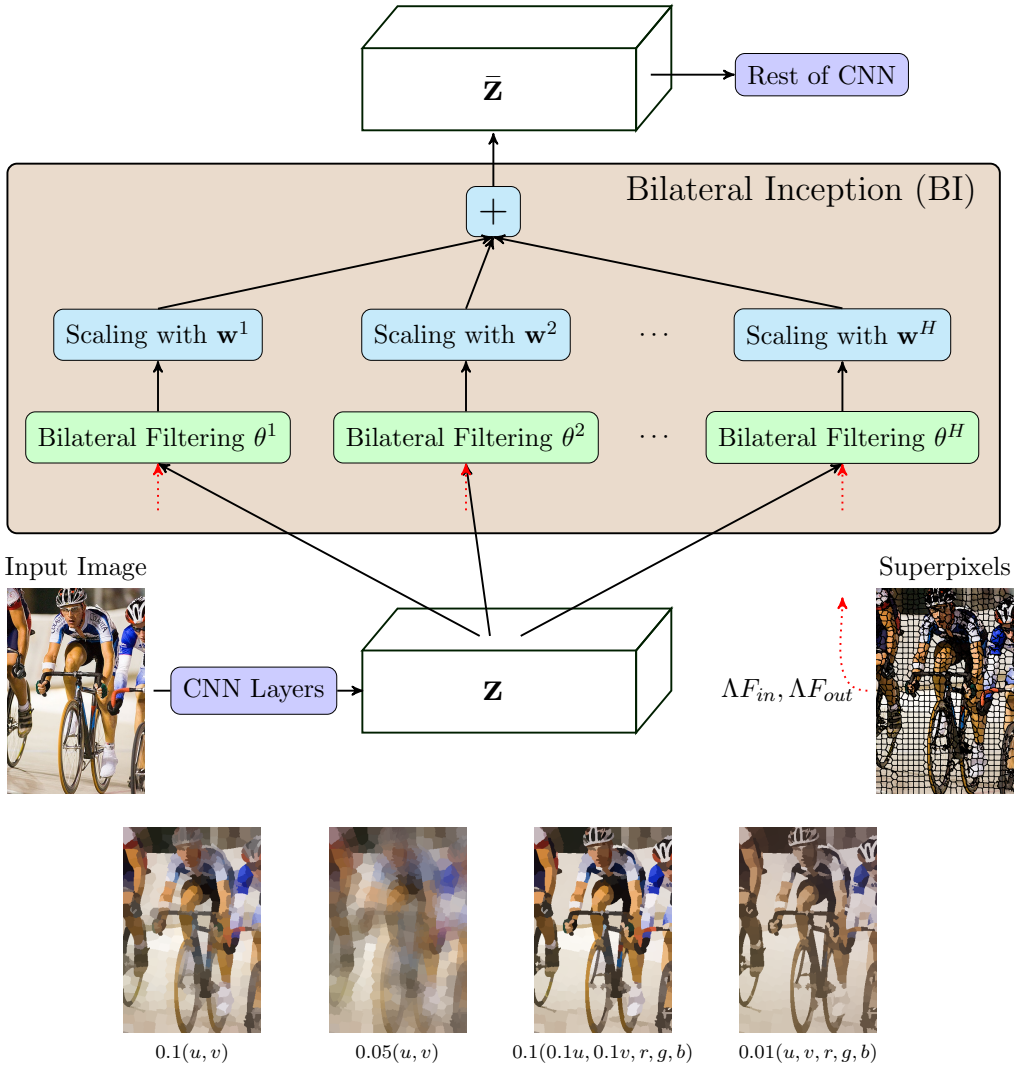


Figure 4.4: **Visualization of a Bilateral Inception (BI) Module.** The unit activations $\mathbf{z}$ are passed through several bilateral filters defined over different feature spaces. The result is linearly combined to $\bar{\mathbf{z}}$ and passed on to the next network layer. Also shown are sample filtered superpixel images using bilateral filters defined over different example feature spaces. (u, v) correspond to position and (r, g, b) correspond to color features.

produce $\bar{\mathbf{z}}$:

$$\bar{\mathbf{z}}_c = \sum_{h=1}^{H} \mathbf{w}_c^h \hat{\mathbf{z}}_c^h, \tag{4.3}$$

using individual weights $\mathbf{w}_c^h$ per scale θ^h and channel c. The weights $\mathbf{w} \in \mathbb{R}^{H \times C}$ are learned using error-backpropagation. The result of the inception module has C channels for every of its Q points, thus $\bar{\mathbf{z}} \in \mathbb{R}^{Q \times C}$. The inception module is schematically illustrated

in Fig. 4.4. In short, information from CNN layers below is filtered using bilateral filters defined in transformed feature space ($\Lambda\mathbf{f}$). Most operations in the inception module are parallelizable resulting in fast runtimes on a GPU. In this work, inspired from the DenseCRF architecture from [Krähenbühl 2012], we used pairs of BI modules: one with position features (u, v) and another with both position and colour features (u, v, r, g, b), each with multiple scales $\{\theta^h\}$.

Motivation and Comparison to DenseCRF: A BI module filters the activations of a CNN layer. Contrast this with the use of a DenseCRF on the CNN output. At that point the fine-grained information that intermediate CNN layers represent has been condensed already to a low-dimensional vector representing beliefs over labels. Using a mean-field update is propagating information between these beliefs. Similar behaviour is obtained using the BI modules but on different scales (using multiple different filters $K(\theta^h)$) and on the intermediate CNN activations $\mathbf{z}$. Since in the end, the to-be-predicted pixels are not i.i.d., this blurring leads to better performance both when using a bilateral filter as an approximate message passing step of a DenseCRF as well in the system outlined here. Both attempts are encoding prior knowledge about the problem, namely that pixels close in position and color are likely to have the same label. Therefore such pixels can also have the same intermediate representation. Consider one would average CNN representations for all pixels that have the same ground truth label. This would result in an intermediate CNN representation that would be very easy to classify for the later layers.

4.3.3 Superpixel Convolutions

The bilateral inception module allows to change how information is stored in the higher level of a CNN. This is where the superpixels are used. Instead of storing information on a fixed grid, we compute for every image, superpixels S and use the mean color and position of their included pixels as features. We can insert bilateral inception modules to change from grid representations to superpixel representations and vice versa. Inception modules in between superpixel layers convolve the unit activations between all superpixels depending on their distance in the feature space. This retains all properties of the bilateral filter, superpixels that are spatially close and have a similar mean color will have a stronger influence on each other.

Superpixels are not the only choice, in principle one can also sample random points from the image and use them as intermediate representations. We are using superpixels for computational reasons, since they can be used to propagate label information to the full image resolution. Other interpolation techniques are possible, including the well known bilinear interpolation, up-convolution networks [Zeiler 2010], and DenseCRFs [Krähenbühl 2012]. The quantization error mentioned in Sec. 4.3.1 only enters because the superpixels are used for interpolation. Also note that a fixed grid, that is independent of the image is a hard choice of where information should be stored. One could in principle evaluate the CNN densely, at all possible spatial locations, but we found that this resulted in poor performance compared to interpolation methods.

4.3.3.1 Backpropagation and Training.

All free parameters of the inception module $\mathbf{w}$, $\{\theta^h\}$ and Λ are learned via backpropagation. We also backpropagate the error with respect to the module inputs thereby enabling the integration of our inception modules inside CNN frameworks without breaking the end-to-end learning paradigm. As shown in Fig. 4.3, the bilateral filtering can be decomposed into 5 different sub-layers. Derivatives with respect to the open parameters are obtained by the corresponding layer and standard backpropagation through the directed acyclic graph. For example, Λ is optimized by back-propagating gradients through 1×1 convolution. Derivatives for non-standard layers (pairwise similarity, matrix multiplication) are straight forward to obtain using matrix calculus. To let different filters learn the information propagation at different scales, we initialized $\{\theta^h\}$ with well separated scalar values ($e.g.\{1, 0.7, 0.3, ...\}$). The learning is performed using Adam stochastic optimization method [Kingma 2014]. The implementation is done in Caffe neural network framework [Jia 2014], and the code is available online at http://segmentation.is.tuebingen.mpg.de.

4.4 Experiments

We study the effect of inserting and learning bilateral inception modules in various existing CNN architectures. As a testbed we perform experiments on semantic segmentation using the Pascal VOC2012 segmentation benchmark dataset [Everingham 2012], Cityscapes street scene dataset [Cordts 2015] and on material segmentation using the Materials in Context (MINC) dataset from [Bell 2015]. We take different CNN architectures from the works of [Chen 2014a, Zheng 2015, Bell 2015] and insert the inception modules before and/or after the spatial FC layers.

4.4.1 Semantic Segmentation

We first use the Pascal VOC12 segmentation dataset [Everingham 2012] with 21 object classes. For all experiments on VOC2012, we train using the extended training set of 10581 images collected by [Hariharan 2011]. Following [Zheng 2015], we use a reduced validation set of 346 images for validation. We experiment on two different network architectures, (a) DeepLab model from [Chen 2014a] which uses CNN followed by DenseCRF and (b) CRFasRNN model from [Zheng 2015] which uses CNN with deconvolution layers followed by DenseCRF trained end-to-end.

4.4.1.1 DeepLab

We use the publicly available state-of-the-art pre-trained CNN models from [Chen 2014a]. We use the DeepLab-LargeFOV variant as a base architecture and refer to it as 'DeepLab'. The DeepLab CNN model produces a lower resolution prediction ($\frac{1}{8}\times$) which is then bilinearly interpolated to the input image resolution. The original models have been fine-tuned using both the MSCOCO [Lin 2014b] and the extended VOC [Hariharan 2011] datasets. Next, we describe modifications to these models and show performance improvements in terms of both IoU and runtimes.

Model	Training	*IoU*	*Runtime*
DeepLab [Chen 2014a]		68.9	145 ms
With BI modules			
$\text{BI}_6(2)$	only BI	70.8	+20
$\text{BI}_6(2)$	BI+FC	71.5	+20
$\text{BI}_6(6)$	BI+FC	72.9	+45
$\text{BI}_7(6)$	BI+FC	73.1	+50
$\text{BI}_8(10)$	BI+FC	72.0	+30
$\text{BI}_6(2)$-$\text{BI}_7(6)$	BI+FC	73.6	+35
$\text{BI}_7(6)$-$\text{BI}_8(10)$	BI+FC	73.4	+55
$\text{BI}_6(2)$-$\text{BI}_7(6)$	FULL	**74.1**	+35
$\text{BI}_6(2)$-$\text{BI}_7(6)$-CRF	FULL	**75.1**	+865
DeepLab-CRF [Chen 2014a]		72.7	+830
DeepLab-MSc-CRF [Chen 2014a]		**73.6**	+880
DeepLab-EdgeNet [Chen 2015]		71.7	+30
DeepLab-EdgeNet-CRF [Chen 2015]		**73.6**	+860

Table 4.1: **Semantic Segmentation using DeepLab model.** IoU scores on Pascal VOC12 segmentation test dataset and average runtimes (ms) corresponding to different models. Also shown are the results corresponding to competitive dense pixel prediction techniques that used the same base DeepLab CNN. Runtimes also include superpixel computation (6ms). In the second column, 'BI', 'FC' and 'FULL' correspond to training 'BI', 'FC' and full model layers respectively.

We add inception modules after different FC layers in the original model and remove the DenseCRF post processing. For this dataset, we use 1000 SLIC superpixels [Achanta 2012, Ren 2015]. The inception modules after FC_6, FC_7 and FC_8 layers are referred to as $\text{BI}_6(H)$, $\text{BI}_7(H)$ and $\text{BI}_8(H)$ respectively, where H is the number of kernels. All results using the DeepLab model on Pascal VOC12 dataset are summarized in Tab. 4.1. We report the 'test' numbers without validation numbers, because the released DeepLab model that we adapted was trained using both train and validation sets. The DeepLab network achieves an IoU of 68.9 after bilinear interpolation. Experiments with $\text{BI}_6(2)$ module indicate that even only learning the inception module while keeping the remaining network fixed results in an reliable IoU improvement ($+1.9$). Additional joint training with FC layers significantly improved the performance. The results also show that more kernels improve performance. Next, we add multiple modules to the base DeepLab network at various stages and train them jointly. This results in further improvement of the performance. The $\text{BI}_6(2)$-$\text{BI}_7(6)$ model with two inception modules shows significant improvement in IoU by 4.7 and 0.9 in comparison to baseline model and DenseCRF application respectively. Finally, finetuning the entire network (FULL in Tab. 4.1) boosts the performance by 5.2 and 1.4 compared to the baseline and DenseCRF application.

Some visual results are shown in Fig. 4.5 and more are included at the end of the chapter (see Fig. 4.9 and Fig. 4.10). Several other variants of using BI are conceivable. During our experiments, we have observed that more kernels and more modules improve the performance, so we expect that even better results can be achieved. In Tab. 4.1, the runtime (ms) is included for several models. These numbers have been obtained using a Nvidia Tesla K80 GPU and standard Caffe time benchmarking [Jia 2014]. DenseCRF timings are taken from [Chen 2015]. The runtimes indicate that the overhead with BI

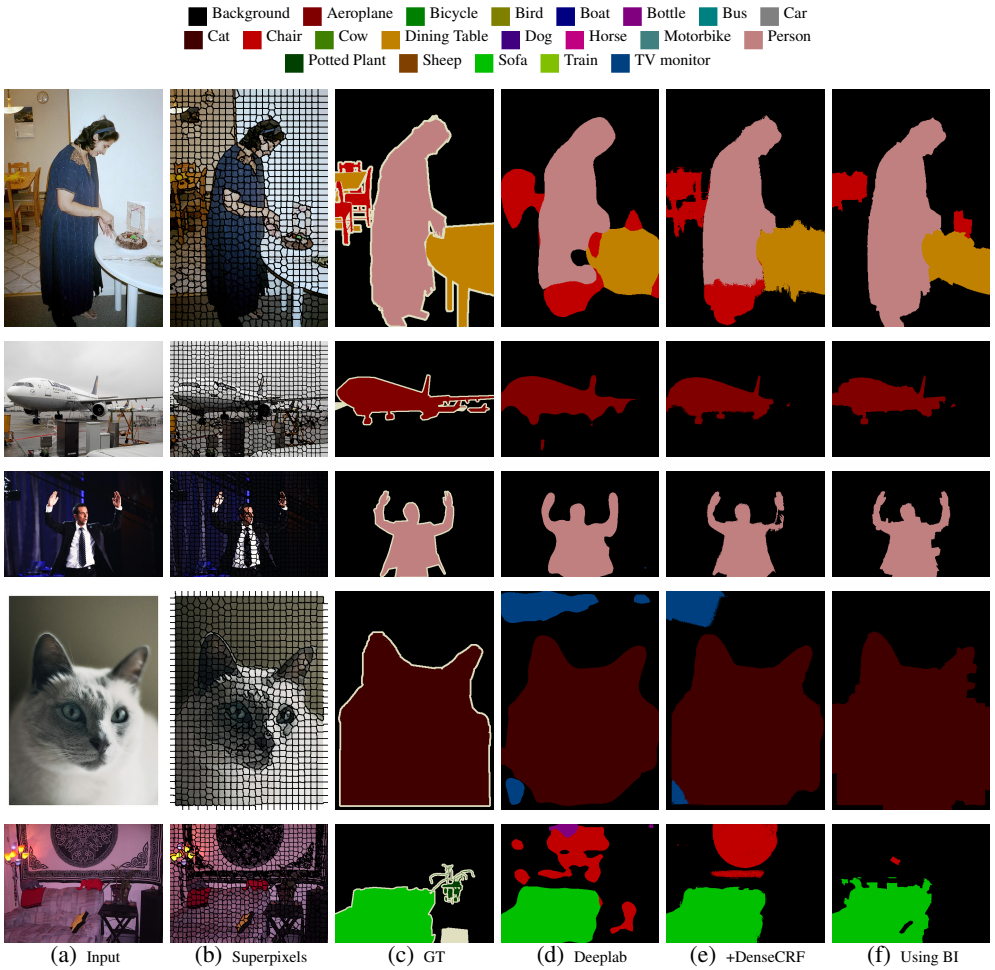


Figure 4.5: **Semantic Segmentation.** Example results of semantic segmentation on Pascal VOC12 dataset. (d) depicts the DeepLab CNN result, (e) CNN + 10 steps of mean-field inference, (f) result obtained with bilateral inception (BI) modules ($\text{BI}_6(2)$+$\text{BI}_7(6)$) between FC layers.

modules is quite minimal in comparison to using Dense CRF.

In addition, we include the results of some other dense pixel prediction methods that are build on top of the same DeepLab base model. DeepLab-MSc-CRF is a multi-scale version [Chen 2014a] of DeepLab with DenseCRF on top. DeepLab-EdgeNet [Chen 2015] is a recently proposed fast and discriminatively trained domain transform technique for propagating information across pixels. Comparison with these techniques in terms of performance and runtime indicates that our approach performs on par with latest dense pixel prediction techniques with significantly less time overhead. Several state-of-the-art CNN based systems [Lin 2015, Liu 2015] have achieved higher results than DeepLab on Pascal VOC12. These models are not yet publicly available and so we could not test the use of BI models in them. A close variant [Barron 2015b] of our work, which propose to do optimization in the bilateral space also has fast runtimes, but reported lower performance in comparison to the application of DenseCRF.

Model	*IoU*	*Runtime*
DeconvNet (CNN+Deconv.)	72.0	190 ms
With BI modules $\text{BI}_3(2)$-$\text{BI}_4(2)$-$\text{BI}_6(2)$-$\text{BI}_7(2)$	**74.9**	245
CRFasRNN (DeconvNet-CRF)	74.7	2700

Table 4.2: **Semantic Segmentation using CRFasRNN model.** IoU scores and runtimes corresponding to different models on Pascal VOC12 test dataset. Note that runtime also includes superpixel computation.

4.4.1.2 CRFasRNN

As a second architecture, we modified the CNN architecture trained by [Zheng 2015] that produces a result at an even lower resolution ($\frac{1}{16}\times$). Multiple deconvolution steps are employed to obtain the segmentation at input image resolution. This result is then passed onto the DenseCRF recurrent neural network to obtain the final segmentation result. We insert BI modules after score-pool3, score-pool4, FC_6 and FC_7 layers, please see [Long 2014, Zheng 2015] for the network architecture details. Instead of combining outputs from the above layers with deconvolution steps, we introduce BI modules after them and linearly combined the outputs to obtain final segmentation result. Note that we entirely removed both the deconvolution and the DenseCRF parts of the original model [Zheng 2015]. See Tab. 4.2 for results on the DeconvNet model. Without the DenseCRF part and only evaluating the deconvolutional part of this model, one obtains an IoU score of 72.0. Ten steps of mean-field inference increase the IoU to 74.7 [Zheng 2015]. Our model, with few additional parameters compared to the base CNN, achieves a IoU performance of 74.9, showing an improvement of 0.2 over the CRFasRNN model. The BI layers lead to better performance than deconvolution and DenseCRF combined while being much faster.

4.4.1.3 Hierarchical Clustering Analysis

We learned the network parameters using 1000 gSLIC superpixels per image, however the inception module allows to change the resolution (a non-square K). To illustrate this, we perform agglomerative clustering of the superpixels, sequentially merging the nearest two superpixels into a single one. We then evaluated the DeepLab-$\text{BI}_6(2)$-$\text{BI}_7(6)$ network using different levels of the resulting hierarchy re-using all the trained network parameters. Results in Fig. 4.6 show that the IoU score on the validation set decreases slowly with decreasing number of points and then drops for less than 200 superpixels. This validates that the network generalizes to different superpixel layouts and it is sufficient to represent larger regions of similar color by fewer points. In future, we plan to explore different strategies to allocate the representation to those regions that require more resolution and to remove the superpixelization altogether. Fig. 4.6 shows example image with 200, 600, and 1000 superpixels and their obtained segmentation with BI modules.

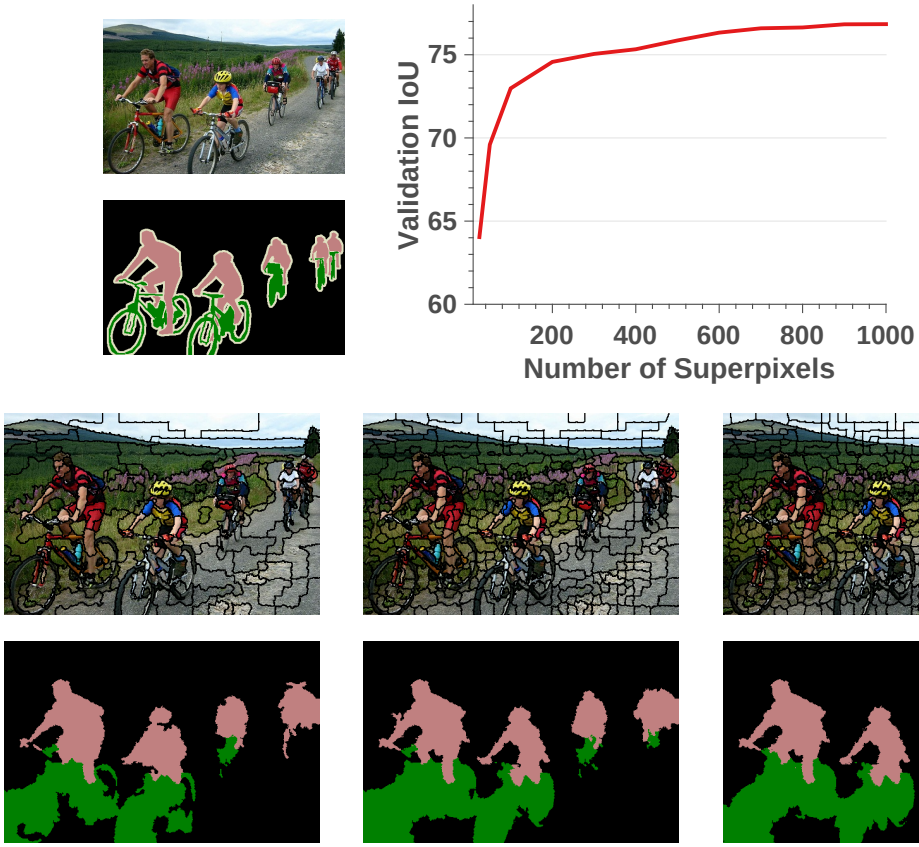


Figure 4.6: **Hierarchical Clustering Analysis.** Top: Validation performance when using different super-pixel layouts and an example image with its corresponding ground-truth. Bottom: From left to right, visualization of superpixelization and segmentation results of the example image using the $\text{BI}_6(2)$-$\text{BI}_7(6)$ model with 200, 600, and 1000 superpixels.

4.4.2 Material Segmentation

We also experimented on a different pixel prediction task of material segmentation by adapting a CNN architecture finetuned for Materials in Context (MINC) [Bell 2015] dataset. MINC consists of 23 material classes and is available in three different resolutions with the same aspect ratio: low (550^2), mid (1100^2) and an original higher resolution. The authors of [Bell 2015] train CNNs on the mid resolution images and then combine with a DenseCRF to predict and evaluate on low resolution images. We build our work based on the AlexNet model [Krizhevsky 2012] released by the authors of [Bell 2015]. To obtain a per pixel labeling of a given image, there are several processing steps that [Bell 2015] use for good performance. First, a CNN is applied at several scales with different strides followed by an interpolation of the predictions to reach the input image resolution and is then followed by a DenseCRF. For simplicity, we choose to run the CNN network with single scale and no-sliding. The authors used just one kernel with (u, v, L, a, b) features in the DenseCRF part. We used the same features in our inception modules. We modified

Model	Class / Total accuracy	Runtime
Alexnet CNN	55.3 / 58.9	300 ms
$\text{BI}_7(2)$-$\text{BI}_8(6)$	67.7 / 71.3	410
$\text{BI}_7(6)$-$\text{BI}_8(6)$	**69.4 / 72.8**	470
AlexNet-CRF	65.5 / 71.0	3400

Table 4.3: **Material Segmentation using AlexNet.** Pixel accuracies and runtimes (in ms) of different models on 'test' split of MINC material segmentation dataset [Bell 2015]. Runtimes also include the time for superpixel extraction (15 ms).

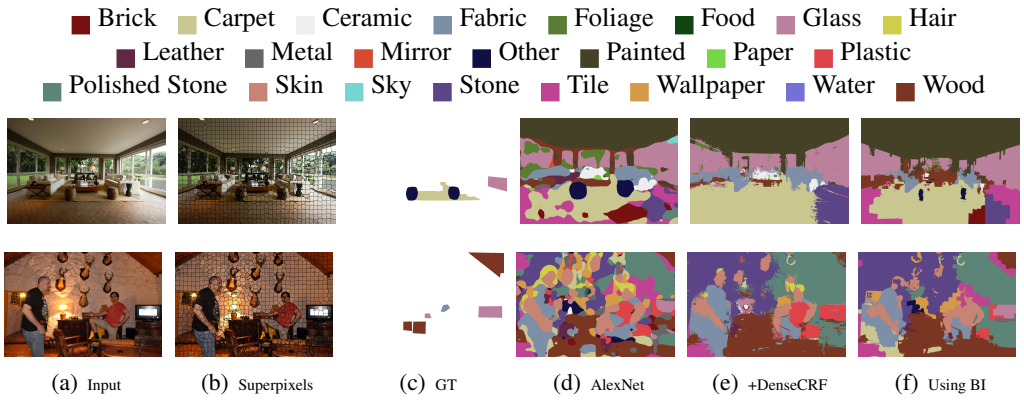


Figure 4.7: **Material Segmentation.** Example results of material segmentation. (d) depicts the AlexNet CNN result, (e) CNN + 10 steps of mean-field inference, (f) results obtained with bilateral inception (BI) modules ($\text{BI}_7(2)$+$\text{BI}_8(6)$) between FC layers. Note that only few regions of the images are annotated in the ground-truth.

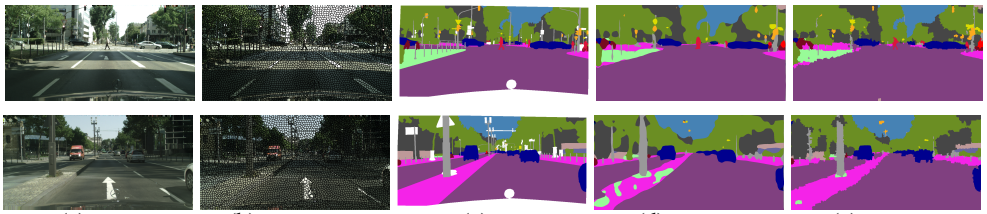
the base AlexNet model by inserting BI modules after FC_7 and FC_8 layers. Again, 1000 SLIC superpixels are used for all experiments. Results on the test set are shown in Table 4.3. When inserting BI modules, the performance improves both in total pixel accuracy as well as in class-averaged accuracy. We observe an improvement of 12% compared to CNN predictions and $2-4\%$ compared to CNN+DenseCRF results. Qualitative examples are shown in Fig. 4.7 and more are included at the end of the chapter (see Fig. 4.11). The weights to combine outputs in the BI layers are found by validation on the validation set. For this model we do not provide any learned setup due very limited segment training data.

4.4.3 Street Scene Segmentation

We further evaluate the use of BI modules on the Cityscapes dataset [Cordts 2015]. Cityscapes contains 20K high-resolution (1024×2048) images of street scenes with coarse pixel annotations and another 5K images with fine annotations, all annotations are from 19 semantic classes. The 5K images are divided into 2975 train, 500 validation and remaining test images. Since there are no publicly available pre-trained models for this dataset yet, we trained a DeepLab model. We trained the base DeepLab model with half resolution images (512×1024) so that the model fits into GPU memory. The result is then interpolated to

Model	IoU (Half-res.)	IoU (Full-res.)	Runtime
DeepLab CNN	62.2	65.7	0.3s
$BI_6(2)$	62.7	66.5	5.7
$BI_6(2)$-$BI_7(6)$	**63.1**	**66.9**	6.1
DeepLab-CRF	63.0	66.6	6.9

Table 4.4: **Street Scene Segmentation using DeepLab model.** IoU scores and runtimes (in sec) of different models on Cityscapes segmentation dataset [Cordts 2015], for both half-resolution and full-resolution images. Runtime computations also include superpixel computation time (5.2s).



(a) Input (b) Superpixels (c) GT (d) Deeplab (e) Using BI

Figure 4.8: **Street Scene Segmentation.** Example results of street scene segmentation. (d) depicts the DeepLab results, (e) result obtained by adding bilateral inception (BI) modules ($BI_6(2)$+$BI_7(6)$) between FC layers. More at the end of the chapter.

full-resolution using bilinear interpolation.

We experimented with two layouts: only a single $BI_6(2)$ and one with two inception $BI_6(2)$-$BI_7(6)$ modules. We notice that the SLIC superpixels [Achanta 2012] give higher quantization error than on VOC and thus used 6000 superpixels using [Dollár 2013] for our experiments. Quantitative results on the validation set are shown in Tab. 4.4. In contrast to the findings on the previous datasets, we only observe modest improvements with both DenseCRF and our inception modules in comparison to the base model. Similar to the previous experiments, the inception modules achieve better performance than DenseCRF while being faster. The majority of the computation time in our approach is due to the extraction of superpixels ($5.2s$) using a CPU implementation. Some visual results with $BI_6(2)$-$BI_7(6)$ model are shown in Fig. 4.8 with more at the end of the chapter (see Fig. 4.12 and Fig. 4.13).

4.5 Conclusion

The DenseCRF [Krähenbühl 2012] with mean-field inference has been used in many CNN segmentation approaches. Its main ingredient and reason for the improved performance is the use of a bilateral filter applied to the beliefs over labels. We have introduced a CNN approach that uses this key component in a novel way: filtering intermediate representations of higher levels in CNNs while jointly learning the task-specific feature spaces. This propagates information between earlier and more detailed intermediate representations of the classes instead of beliefs over labels. Further we show that image-adaptive layouts in

the higher levels of CNNs can be used to an advantage in the same spirit as CRF graphs have been constructed using superpixels in previous works on semantic segmentation. The computations in the 1×1 convolution layers scales in the number of superpixels which may be an advantage. Further we have shown that the same representation can be used to interpolate the coarser representations to the full image. This work has been presented at the European Conference on Computer Vision2016 [Gadde 2015].

The use of image-adaptive convolutions in between the FC layers retains the appealing effect of producing segmentation masks with sharp edges. This is not necessarily due superpixels. Using superpixels to represent information in FC layers and their use to interpolate to the full resolution are orthogonal. Different interpolation steps can be used to propagate the label information to the entire image, including bilinear interpolation, up-convolutions and DenseCRFs. We plan to investigate the effect of different sampling strategies to represent information in the higher layers of CNNs and apply similar image-adaptive ideas to videos.

We believe that the Bilateral Inception models are an interesting step that aims to directly include the model structure of CRF factors into the forward architecture of CNNs. The BI modules are easy to implement and are applicable to CNNs that perform structured output prediction.

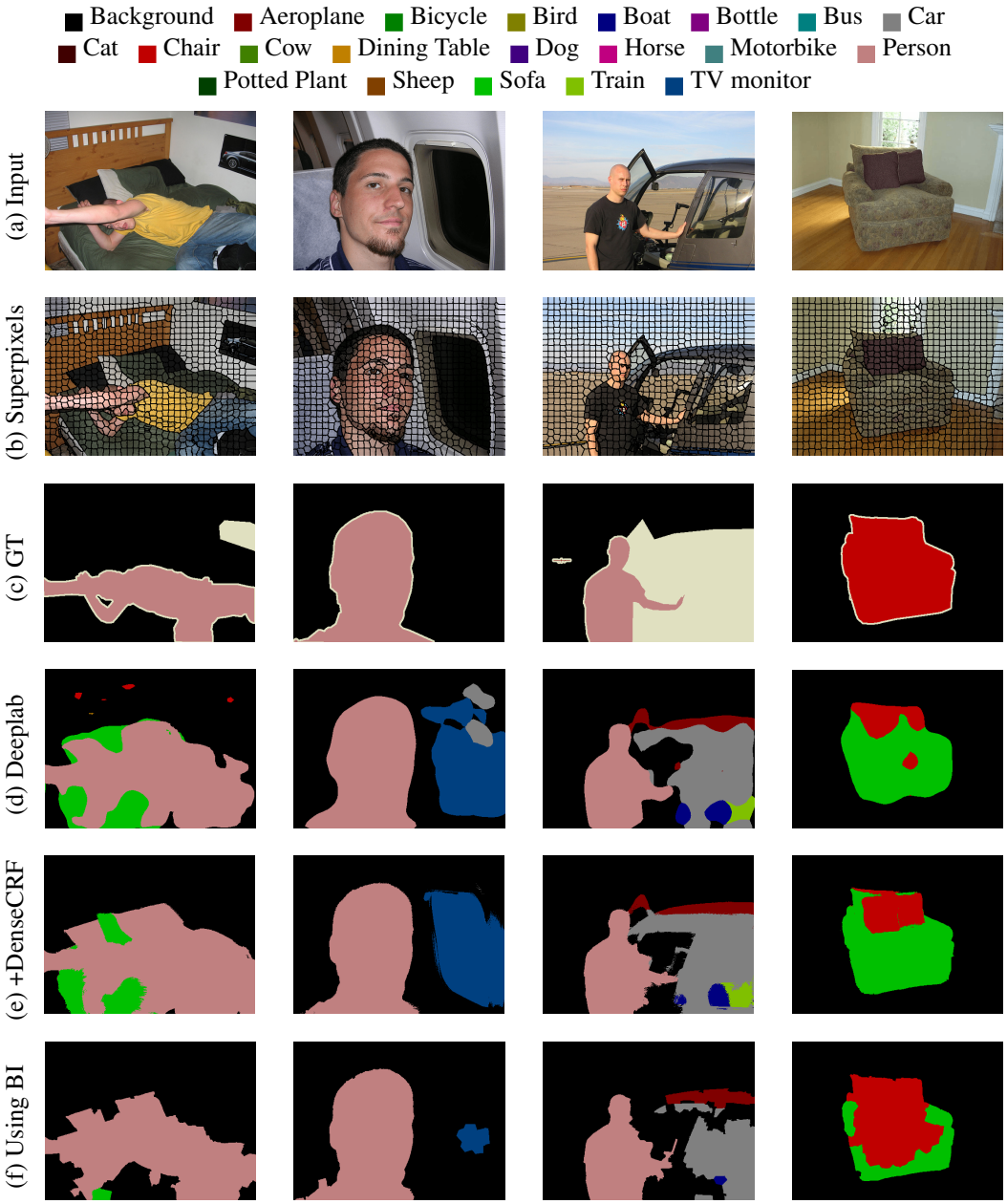


Figure 4.9: **Semantic Segmentation.** Example results of semantic segmentation on Pascal VOC12 dataset. (d) depicts the DeepLab CNN result, (e) CNN + 10 steps of mean-field inference, (f) result obtained with bilateral inception (BI) modules ($\text{BI}_6(2)$+$\text{BI}_7(6)$) between FC layers.

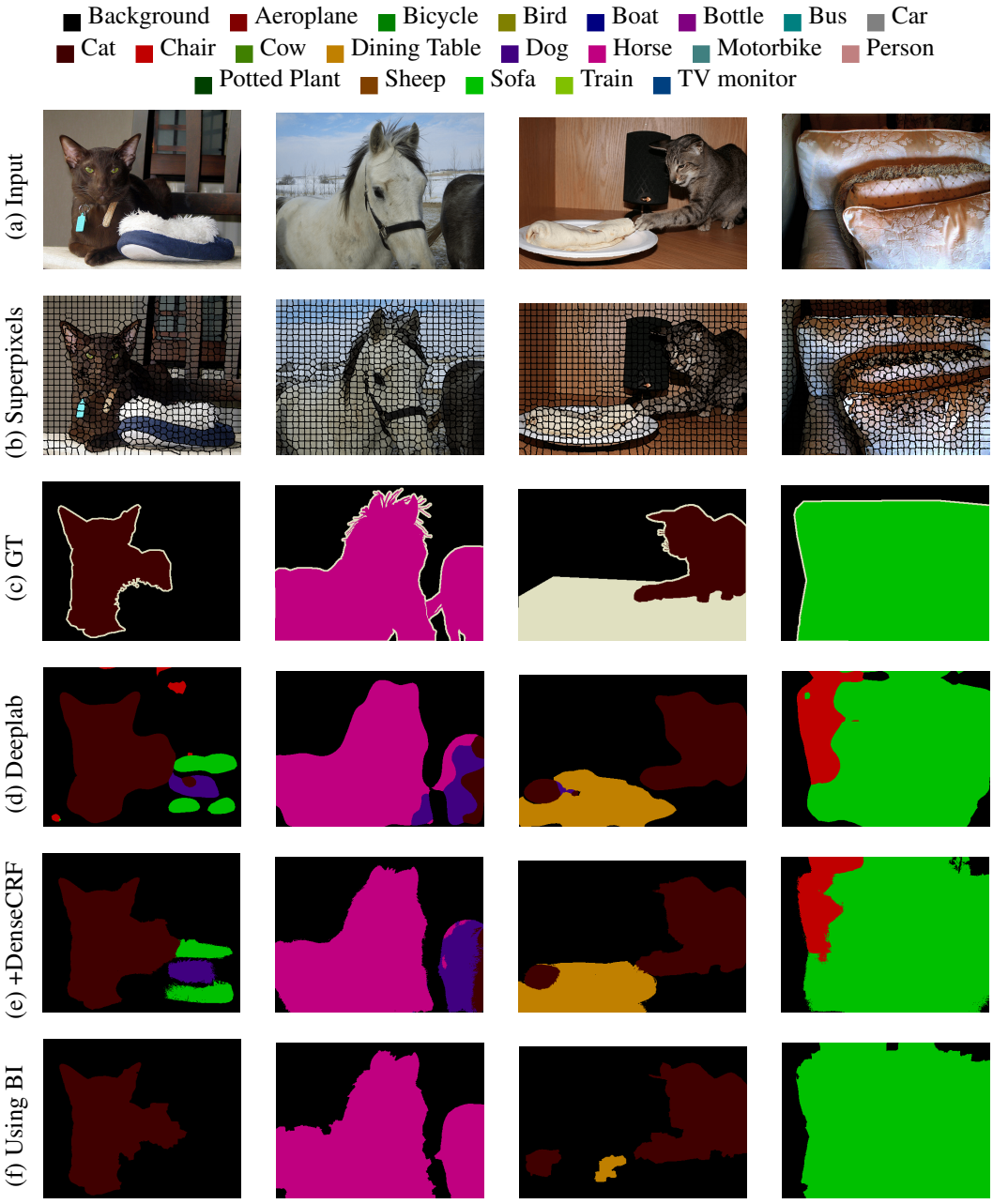


Figure 4.10: **Semantic Segmentation.** More example results of semantic segmentation on Pascal VOC12 dataset. (d) depicts the DeepLab CNN result, (e) CNN + 10 steps of mean-field inference, (f) result obtained with bilateral inception (BI) modules ($\text{BI}_6(2)$+$\text{BI}_7(6)$) between FC layers.

Figure 4.11: **Material Segmentation.** Example results of material segmentation. (d) depicts the AlexNet CNN result, (e) CNN + 10 steps of mean-field inference, (f) result obtained with bilateral inception (BI) modules ($\text{BI}_7(2)$+$\text{BI}_8(6)$) between FC layers.

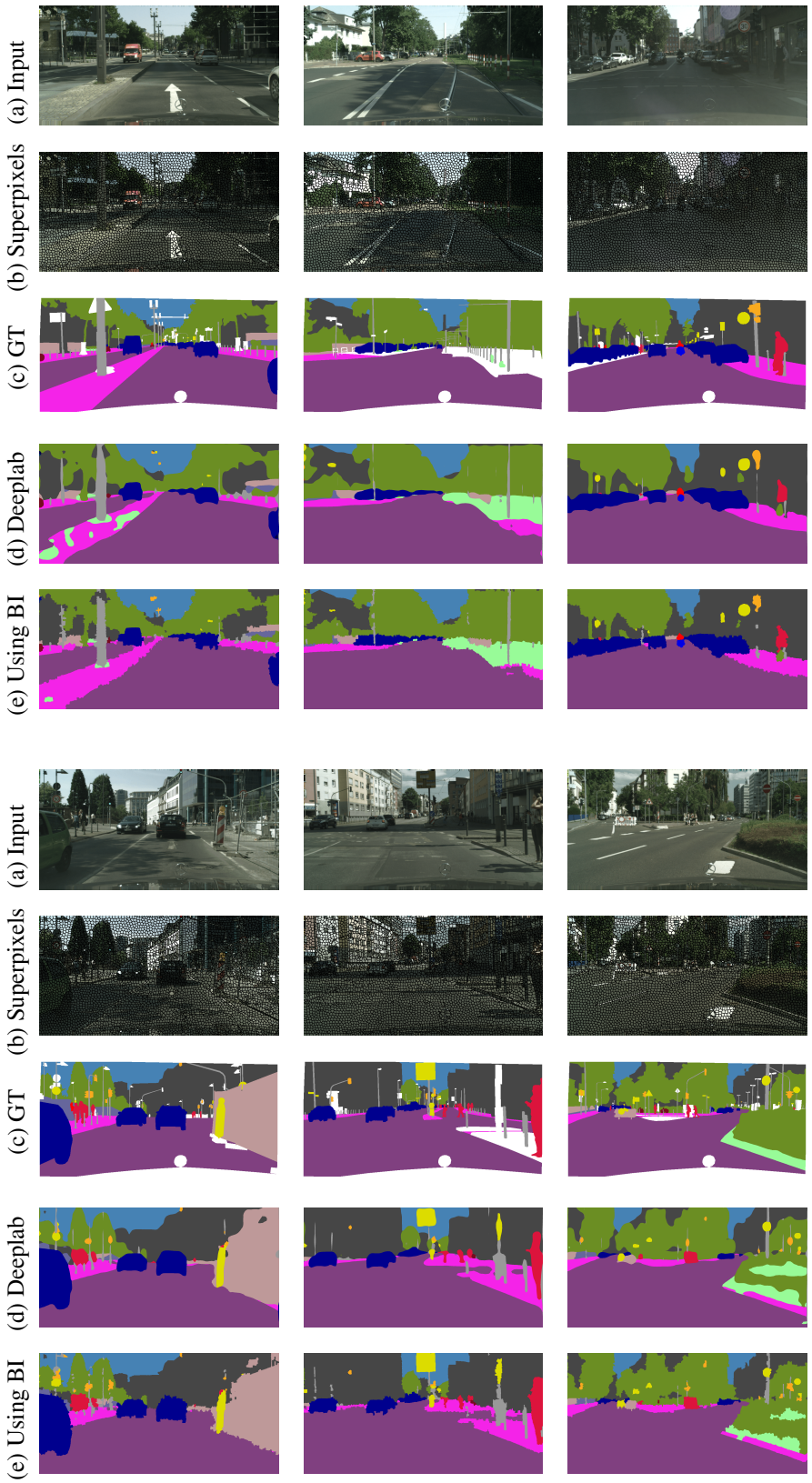


Figure 4.12: **Street Scene Segmentation.** Example results of street scene segmentation. (d) depicts the DeepLab results, (e) result obtained by adding bilateral inception (BI) modules ($\text{BI}_6(2)$+$\text{BI}_7(6)$) between FC layers.

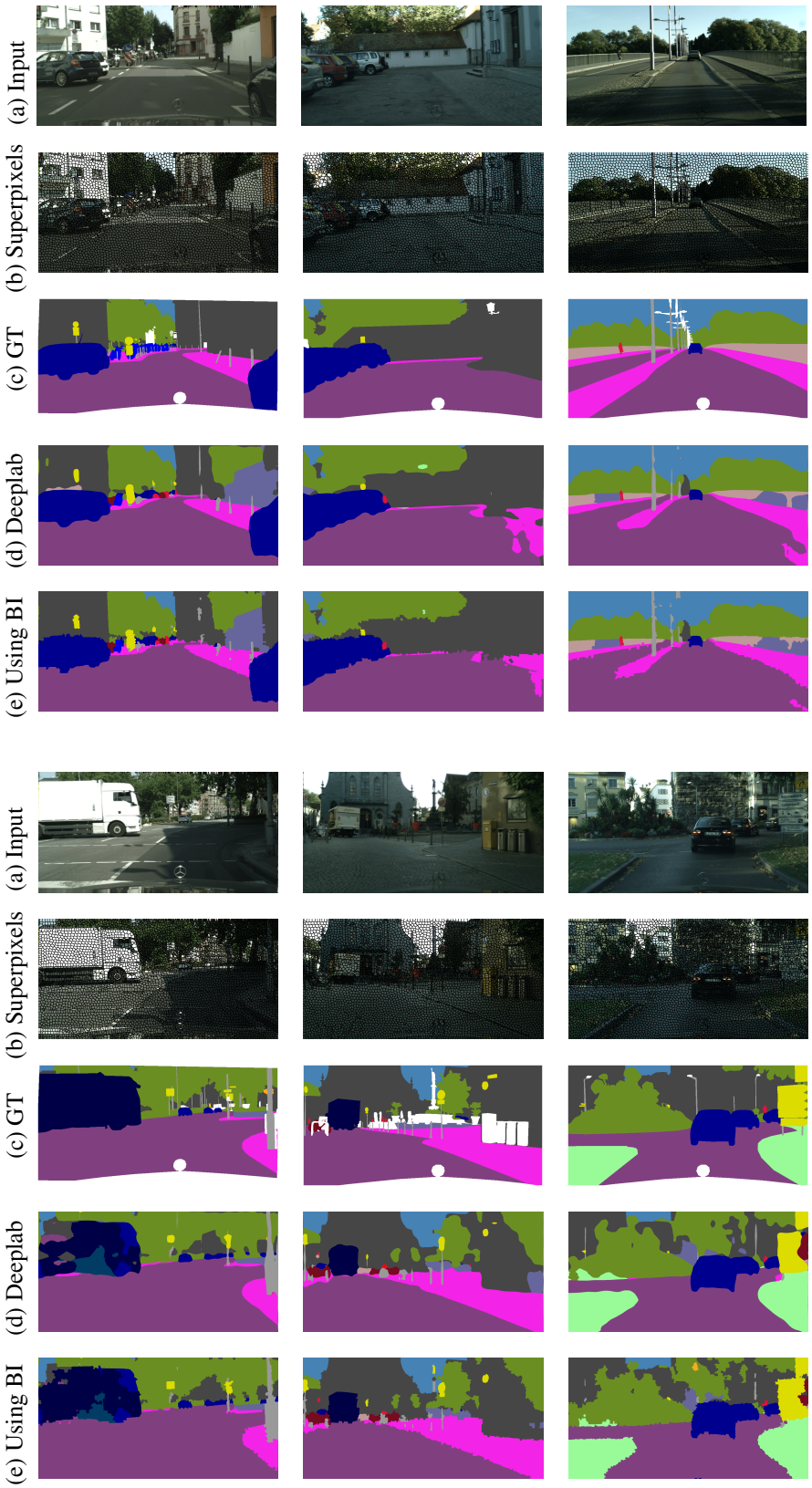


Figure 4.13: **Street Scene Segmentation.** More example results of street scene segmentation. (d) depicts the DeepLab results, (e) result obtained by adding bilateral inception (BI) modules ($\text{BI}_6(2)$+$\text{BI}_7(6)$) between FC layers.

CHAPTER 5

Conclusions and Future Work

This thesis studies semantic segmentation of strongly regular images (building facades segmented into wall, window, balcony, etc.) and weakly regular images (natural scenes segmented into cow, horse, aeroplane, etc). The main contributions of the thesis are as follows. First we proposed an approach to learn prior knowledge in terms of grammars for segmenting strongly regular facade images, given a small training set of annotated facade images [Gadde 2016b]. These learned grammars can be used in several applications like procedural and inverse procedural modeling, structure based facade retrieval, etc. The second contribution is a fast, efficient and domain-independent approach for semantic segmentation of 2D images and 3D point clouds of facades [Gadde 2016a]. The third contribution addresses the problem of encoding domain knowledge in a CNN by performing bilateral filtering between successive layers for segmenting natural images [Gadde 2015]. In the following, we analyze these contributions and suggest future work.

In chapter 2, we have proposed a novel method for learning split grammars from annotated images, and we have used it to learn typologies of architectures. The method assumes a simple generic grammar which is used to parse the training set. Reasoning on the associated derivation trees, to first identify common subtrees and then merge similar trees, determines the set of meta-rules corresponding the observed typology of buildings. It leads to a grammar which is compact (in terms of derivation trees) and simple (in terms of inference process). We obtained state-of-the-art results with respect to typology-specific handcrafted grammars or to grammars learned from data which demonstrates the extreme potentials of our method.

Extending this to other typologies of architecture is an obvious work, such as applying the concept to modern architectures. Improving the process of establishing the set of meta-rules by reasoning simultaneously on the compact derivations of all training examples is a natural extension of our method. Considering more trees at the rule-merging stage may also lead to an improved performance. Such an effort will require further exploiting the mutual relations between trees in terms of derivation sequences and will further compact the grammar in terms of rules, resulting in facilitated inference. Furthermore, extending the proposed approach to 3D grammars is an extremely promising task, and in particular when taking into account the difficulty of defining such a grammar manually. Another challenging direction is to perform unsupervised clustering on the set of obtained results towards capturing image-driven architectural differences. This will be an initial and interesting step towards compact large-scale urban modeling and understanding.

In chapter 3, we described a fast and efficient segmentation framework that is a combination of proven and established components (features, classifiers and stacked generalization of classifiers). We observe on several datasets that adding stacked classifiers using

auto-context features improves the performance. This applies to both 2D (images) and 3D (point clouds) data. The approach largely ignores domain knowledge, but still performs better than all competing methods that include prior information in some form, for example relationship between balconies and windows, etc.

An obvious extension to this work is to include domain knowledge using presumably complicated inference techniques for structured prediction. A more interesting extension would be to train in an end-to-end fashion. The use of convolutional neural networks may remove the need of hand-crafted features. Further, the bilateral inception module introduced in chapter 4 can be used for structured prediction which can still be trained in end-to-end fashion (for example, CNN followed by auto-context followed by bilateral inceptions).

Finally, in chapter 4, we presented an efficient technique based on convolutional neural networks for segmenting weakly-structured natural scene images. The standard state-of-the-art approach is to use the predictions from CNN's as unaries and later use a CRF to do structured prediction [Zheng 2015, Chen 2016] as a post processing step. Using our proposed bilateral inception module, we are able to do structured prediction with in a CNN by performing image-adaptive (*bilateral*) filtering. Our experiments shows that incorporating our BI modules into a CNN significantly improved the performance of the resulting CNN compared to the original baseline CNN and to the CNN followed by CRF approaches. In terms of computational performance, the demonstrated improvements were achieved with a very little overhead (in the order of few milliseconds) on top of the runtime of the CNN.

Several extensions based on this work are conceivable. The proposed BI modules were introduced between the deeper layers of a CNN. It will be interesting to explore how the BI modules fare when incorporated between the initial or shallow layers of the CNN and whether such a usage can help in reducing the depth of the network. The belief here is that this will reduce the need for very deep architectures by increasing the receptive field. Further, the convolution filters can be learned in bilateral space by using the permutohedral lattice approximation based on the work of [Jampani 2016b]. Further, CNN-based techniques for segmenting high-resolution images (typically more than 2MP) suffer from insufficient GPU memory and are computationally costly. A possible solution path is to investigate the usage of our BI modules in the early stages of a CNN.

Another exciting direction would be to extend the technique for semantic segmentation of videos and 3D data using 3D CNNs. CNNs are computationally expensive when applied to 3D point clouds due to sparse and uneven sampling, or to videos due to high redundancy across frames. The challenges are different for videos and 3D which require encoding spatio-temporal smoothness and spatial smoothness respectively. These two issues, i.e., encoding domain structure and computational complexity in video and 3D segmentation could be addressed by performing data-adaptive bilateral filtering using super-pixels/voxels within a CNN. Data adaptive CNNs can be designed by stacking these bilateral layers. Moreover, the bilateral filters/convolutions can be learned from data using [Jampani 2016b].

Another direction is to investigate for better bilateral features to perform data-adaptive filtering. Conventionally in image segmentation, the pixel-wise photometric (RGB) and spatial (XY) features are used to compute the bilateral filter. However, as the spatio-

temporal features cannot be used as bilateral features in videos directly (as both camera and the scene can be moving), investigation for better features is a natural and important extension. One can try to investigate non-linear embeddings, such as isomap embeddings or learn the embeddings of features. Such embeddings can also be useful to obtain better video superpixels (analogous to the SLIC superpixels for images) which can be used for other computer vision tasks. Similarly for 3D data, what features constitute a better combination for computing the bilateral filter is an open question.

Finally, a useful study would be to evaluate the performance of the BI modules on other dense pixel-wise prediction tasks such as optical flow estimation, depth estimation from monocular images, etc. We believe that the bilateral inception module could be a core building block for any dense pixel prediction task on different visual data (images, videos, 3D, etc).

Bibliography

[Achanta 2012] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua and Sabine Susstrunk. *SLIC superpixels compared to state-of-the-art superpixel methods*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pages 2274–2282, 2012. (Cited on pages 8, 83, 88 and 93.)

[Adams 2009] Andrew Adams, Natasha Gelfand, Jennifer Dolson and Marc Levoy. *Gaussian kd-trees for fast high-dimensional filtering*. In ACM Transactions on Graphics (TOG), volume 28, page 21. ACM, 2009. (Cited on page 84.)

[Adams 2010] Andrew Adams, Jongmin Baek and Myers Abraham Davis. *Fast High-Dimensional Filtering Using the Permutohedral Lattice*. In Computer Graphics Forum, volume 29, pages 753–762. Wiley Online Library, 2010. (Cited on pages 82 and 84.)

[Alegre 2004] Fernando Alegre and Frank Dellaert. *A probabilistic approach to the semantic interpretation of building facades*. In CIPA International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres, pages 25–27, 2004. (Cited on page 8.)

[Barron 2015a] Jonathan T Barron, Andrew Adams, YiChang Shih and Carlos Hernández. *Fast Bilateral-Space Stereo for Synthetic Defocus*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4466–4474, 2015. (Cited on page 82.)

[Barron 2015b] Jonathan T Barron and Ben Poole. *The Fast Bilateral Solver*. arXiv preprint arXiv:1511.03296, 2015. (Cited on pages 82 and 89.)

[Bay 2006] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. *SURF: Speeded up robust features*. In European Conference on Computer Vision, pages 404–417, 2006. (Cited on page 1.)

[Bell 2015] Sean Bell, Paul Upchurch, Noah Snavely and Kavita Bala. *Material Recognition in the Wild with the Materials in Context Database*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. (Cited on pages 80, 81, 83, 87, 91 and 92.)

[Benz 2013] Florian Benz and Timo Kötzing. *An effective heuristic for the smallest grammar problem*. In Proceedings of the 15th annual conference on Genetic and evolutionary computation, pages 487–494. ACM, 2013. (Cited on page 13.)

[Berg 2007] Alexander C Berg, Floraine Grabler and Jitendra Malik. *Parsing images of architectural scenes*. In Proceedings of the IEEE Conference on Computer Vision, pages 1–8, 2007. (Cited on page 8.)

[Bod 2003] Rens Bod. *An Efficient Implementation of a New DOP Model*. In 10th Conference on European Chapter of the Association for Computational Linguistics (EACL 2003), Volume 1, pages 19–26, 2003. (Cited on page 12.)

[Bod 2006] Rens Bod. *An All-subtrees Approach to Unsupervised Parsing*. In 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006), pages 865–872. Association for Computational Linguistics, 2006. (Cited on page 11.)

[Breiman 2001] Leo Breiman. *Random Forests*. Machine Learning, vol. 45, no. 1, pages 5–32, 2001. (Cited on page 1.)

[Bruna 2013] Joan Bruna, Wojciech Zaremba, Arthur Szlam and Yann LeCun. *Spectral networks and locally connected networks on graphs*. arXiv preprint arXiv:1312.6203, 2013. (Cited on page 82.)

[Campbell 2013] Neill Campbell, Kartic Subr and Jan Kautz. *Fully-connected CRFs with non-parametric pairwise potential*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1658–1665, 2013. (Cited on page 82.)

[Carrasco 2001] Rafael C. Carrasco, Jose Oncina and Jorge Calera-Rubio. *Stochastic Inference of Regular Tree Languages*. Machine Learning, vol. 44, no. 1-2, pages 185–197, 2001. (Cited on page 11.)

[Charikar 2002] Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, April Rasala, Amit Sahai*et al.* *Approximating the smallest grammar: Kolmogorov complexity in natural models*. In Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing (STOC), pages 792–801. ACM, 2002. (Cited on page 13.)

[Charikar 2005] Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai and Abhi Shelat. *The smallest grammar problem*. Information Theory, IEEE Transactions on, vol. 51, no. 7, pages 2554–2576, 2005. (Cited on page 13.)

[Chen 2007] Yuanhao Chen, Long Zhu, Chenxi Lin, Hongjiang Zhang and Alan L Yuille. *Rapid inference on a novel and/or graph for object detection, segmentation and parsing*. In Advances in Neural Information Processing Systems, pages 289–296, 2007. (Cited on page 2.)

[Chen 2014a] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy and Alan L Yuille. *Semantic image segmentation with deep convolutional nets and fully connected CRFs*. arXiv preprint arXiv:1412.7062, 2014. (Cited on pages 2, 69, 80, 81, 87, 88 and 89.)

[Chen 2014b] Liang-Chieh Chen, Alexander G Schwing, Alan L Yuille and Raquel Urtasun. *Learning deep structured models*. arXiv preprint arXiv:1407.2538, 2014. (Cited on page 81.)

[Chen 2015] Liang-Chieh Chen, Jonathan T Barron, George Papandreou, Kevin Murphy and Alan L Yuille. *Semantic Image Segmentation with Task-Specific Edge Detection Using CNNs and a Discriminatively Trained Domain Transform*. arXiv preprint arXiv:1511.03328, 2015. (Cited on pages 80, 81, 88 and 89.)

[Chen 2016] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy and Alan L Yuille. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. arXiv:1606.00915, 2016. (Cited on page 102.)

[Chi 2005] Yun Chi, Richard R Muntz, Siegfried Nijssen and Joost N Kok. *Frequent subtree mining – An overview*. Fundamenta Informaticae, vol. 66, no. 1, pages 161–198, 2005. (Cited on page 28.)

[Clark 2010] Alexander Clark. *Distributional learning of some context-free languages with a minimally adequate teacher*. In Grammatical Inference: Theoretical Results and Applications, pages 24–37. Springer, 2010. (Cited on page 11.)

[Cohen 2013] Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster and Lyle H. Ungar. *Experiments with Spectral Learning of Latent-Variable PCFGs*. In Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL 2013), pages 148–157, 2013. (Cited on page 12.)

[Cohen 2014a] Andrea Cohen, Alexander G Schwing and Marc Pollefeys. *Efficient Structured Parsing of Facades Using Dynamic Programming*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3206–3213, 2014. (Cited on pages 7, 8, 35, 36 and 38.)

[Cohen 2014b] Andrea Cohen, Alexander Gerhard Schwing and Marc Pollefeys. *Efficient structured parsing of facades using dynamic programming*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3206–3213, 2014. (Cited on pages 50, 51, 58, 59 and 63.)

[Cohen 2014c] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster and Lyle Ungar. *Spectral learning of latent-variable PCFGs: Algorithms and sample complexity*. Journal of Machine Learning Research, vol. 15, no. 1, pages 2399–2449, 2014. (Cited on page 12.)

[Cohn 2010] Trevor Cohn, Phil Blunsom and Sharon Goldwater. *Inducing tree-substitution grammars*. Journal of Machine Learning Research, vol. 11, pages 3053–3096, 2010. (Cited on page 12.)

[Comaniciu 2002] Dorin Comaniciu and Peter Meer. *Mean shift: A robust approach toward feature space analysis*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pages 603–619, 2002. (Cited on page 8.)

[Cordts 2015] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth and Bernt Schiele. *The Cityscapes Dataset*. In CVPR Workshop on The Future of Datasets in Vision, 2015. (Cited on pages 87, 92 and 93.)

[Cortes 1995] Corinna Cortes and Vladimir Vapnik. *Support-vector networks*. Machine Learning, vol. 20, no. 3, pages 273–297, 1995. (Cited on page 1.)

[Dai 2012] Dengxin Dai, Mukta Prasad, Gerhard Schmitt and Luc Van Gool. *Learning domain knowledge for façade labelling*. In European Conference on Computer Vision, pages 710–723, 2012. (Cited on pages 7 and 8.)

[Dalal 2005] Navneet Dalal and Bill Triggs. *Histograms of oriented gradients for human detection*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 886–893, 2005. (Cited on pages 1 and 53.)

[Davies 1979] David L Davies and Donald W Bouldin. *A cluster separation measure*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 1, no. 2, pages 224–227, 1979. (Cited on pages 33 and 34.)

[De La Higuera 2005] Colin De La Higuera. *A bibliographical study of grammatical inference*. Pattern recognition, vol. 38, no. 9, pages 1332–1348, 2005. (Cited on page 11.)

[De la Higuera 2010] Colin De la Higuera. Grammatical inference: learning automata and grammars. Cambridge University Press, 2010. (Cited on page 11.)

[Dollár 2009] Piotr Dollár, Zhuowen Tu, Pietro Perona and Serge Belongie. *Integral Channel Features.* In British Machine Vision Conference, volume 2, page 5, 2009. (Cited on page 53.)

[Dollár 2013] Piotr Dollár and C. Lawrence Zitnick. *Structured Forests for Fast Edge Detection*. In Proceedings of the IEEE Conference on Computer Vision, pages 1841–1848, 2013. (Cited on pages 83 and 93.)

[Dollár 2014] Piotr Dollár. *Piotr's Computer Vision Matlab Toolbox (PMT)*. http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html, 2014. (Cited on page 53.)

[Domke 2013] Justin Domke. *Learning graphical model parameters with approximate marginal inference*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 10, pages 2454–2467, 2013. (Cited on page 81.)

[D'Ulizia 2011] Arianna D'Ulizia, Fernando Ferri and Patrizia Grifoni. *A survey of grammatical inference methods for natural language learning*. Artificial Intelligence Review, vol. 36, no. 1, pages 1–27, 2011. (Cited on page 11.)

[Dunn 1974] Joseph C Dunn. *Well-separated clusters and optimal fuzzy partitions*. Journal of cybernetics, vol. 4, no. 1, pages 95–104, 1974. (Cited on page 33.)

[Everingham 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn and Andrew Zisserman. *The pascal visual object classes (voc) challenge*. International Journal of Computer Vision, vol. 88, no. 2, pages 303–338, 2010. (Cited on pages 1 and 56.)

[Everingham 2012] Mark Everingham, Luc Van Gool, C.K. Williams, John Winn and Andrew Zisserman. *The PASCAL VOC2012 challenge results*, 2012. (Cited on pages 83 and 87.)

[Flajolet 1990] Philippe Flajolet, Paolo Sipala and Jean-Marc Steyaert. *Analytic Variations on the Common Subexpression Problem*. In Proceedings of the 17th International Colloquium on Automata, Languages and Programming, pages 220–234. Springer, 1990. (Cited on page 28.)

[Frey 2007] Brendan J Frey and Delbert Dueck. *Clustering by passing messages between data points*. science, vol. 315, no. 5814, pages 972–976, 2007. (Cited on page 31.)

[Frohlich 2010] Bjorn Frohlich, Erik Rodner and Joachim Denzler. *A fast approach for pixelwise labeling of facade images*. In International Conference on Pattern Recognition, pages 3029–3032, 2010. (Cited on pages 56 and 57.)

[Fröhlich 2012] Björn Fröhlich, Erik Rodner and Joachim Denzler. *Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach*. In Asian Conference on Computer Vision, pages 218–231, 2012. (Cited on pages 1, 52, 59 and 61.)

[Gadde 2015] Raghudeep Gadde, Varun Jampani, Martin Kiefel and Peter V Gehler. *Superpixel Convolutional Networks using Bilateral Inceptions*. arXiv preprint arXiv:1511.06739, 2015. (Cited on pages 2, 94 and 101.)

[Gadde 2016a] Raghudeep Gadde, Varun Jampani, Renaud Marlet and Peter V Gehler. *Efficient 2D and 3D Facade Segmentation using Auto-Context*. arXiv preprint arXiv:1606.06437, 2016. (Cited on pages 67 and 101.)

[Gadde 2016b] Raghudeep Gadde, Renaud Marlet and Nikos Paragios. *Learning Grammars for Architecture-Specific Facade Parsing*. International Journal of Computer Vision, pages 1–27, 2016. (Cited on pages 44, 57, 61 and 101.)

[Gastal 2011] Eduardo SL Gastal and Manuel M Oliveira. *Domain transform for edge-aware image and video processing*. In ACM Transactions on Graphics (TOG), volume 30. ACM, 2011. (Cited on page 84.)

[Gatta 2014] C Gatta and F Ciompi. *Stacked sequential scale-space taylor context*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014. (Cited on pages 52 and 59.)

[Gehler 2009] Peter Gehler and Sebastian Nowozin. *On feature combination for multiclass object classification*. In Proceedings of the IEEE Conference on Computer Vision, pages 221–228, 2009. (Cited on page 56.)

[Gonfaus 2010] Josep M Gonfaus, Xavier Boix, Joost Van de Weijer, Andrew D Bagdanov, Joan Serrat and Jordi Gonzalez. *Harmony potentials for joint classification and segmentation*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3280–3287, 2010. (Cited on page 82.)

[Gould 2012a] Stephen Gould. *DARWIN: a framework for machine learning and computer vision research and development*. Journal of Machine Learning Research, vol. 13, no. 1, pages 3533–3537, 2012. (Cited on pages 35, 37 and 43.)

[Gould 2012b] Stephen Gould. *DARWIN: A Framework for Machine Learning and Computer Vision Research and Development*. Journal of Machine Learning Research, vol. 13, pages 3533–3537, Dec 2012. (Cited on pages 53, 56 and 58.)

[Gould 2014] Stephen Gould, Jiecheng Zhao, Xuming He and Yuhang Zhang. *Superpixel Graph Label Transfer with Learned Distance Metric*. In European Conference on Computer Vision, 2014. (Cited on page 82.)

[Grünwald 1996] Peter Grünwald. *A Minimum Description Length Approach to Grammar Inference*. In Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing, pages 203–216. Springer-Verlag, 1996. (Cited on page 12.)

[Han 2009] Feng Han and Song-Chun Zhu. *Bottom-up/top-down image parsing with attribute grammar*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 1, pages 59–73, 2009. (Cited on page 2.)

[Hariharan 2011] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji and Jitendra Malik. *Semantic Contours from Inverse Detectors*. In Proceedings of the IEEE Conference on Computer Vision, 2011. (Cited on page 87.)

[He 2014] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Spatial pyramid pooling in deep convolutional networks for visual recognition*. In European Conference on Computer Vision, pages 346–361, 2014. (Cited on page 81.)

[He 2015] Shengfeng He, Rynson WH Lau, Wenxi Liu, Zhe Huang and Qingxiong Yang. *SuperCNN: A Superpixelwise Convolutional Neural Network for Salient Object Detection*. International Journal of Computer Vision, pages 1–15, 2015. (Cited on page 82.)

[Ionescu 2015] Catalin Ionescu, Orestis Vantzos and Cristian Sminchisescu. *Matrix Backpropagation for Deep Networks with Structured Layers*. In Proceedings of the IEEE Conference on Computer Vision, pages 2965–2973, 2015. (Cited on page 81.)

[Jampani 2015] Varun Jampani, Raghudeep Gadde and Peter V Gehler. *Efficient Facade Segmentation Using Auto-context*. In Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on, pages 1038–1045. IEEE, 2015. (Cited on pages 11, 43, 45 and 67.)

[Jampani 2016a] Varun Jampani, Martin Kiefel and Peter V. Gehler. *Learning Sparse High Dimensional Filters: Image filtering, Dense CRFs and Bilateral Neural Networks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4452–4461, 2016. (Cited on pages 81 and 82.)

[Jampani 2016b] Varun Jampani, Martin Kiefel and Peter V. Gehler. *Learning Sparse High Dimensional Filters: Image filtering, Dense CRFs and Bilateral Neural Networks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4452–4461, 2016. (Cited on page 102.)

[Jia 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama and Trevor Darrell. *Caffe: Convolutional architecture for fast feature embedding*. In Proceedings of the ACM International Conference on Multimedia, pages 675–678. ACM, 2014. (Cited on pages 84, 87 and 88.)

[Johnson 1999] Andrew E Johnson and Martial Hebert. *Using spin images for efficient object recognition in cluttered 3D scenes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999. (Cited on page 54.)

[Johnson 2007] Mark Johnson, Thomas Griffiths and Sharon Goldwater. *Bayesian Inference for PCFGs via Markov Chain Monte Carlo*. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, pages 139–146, April 2007. (Cited on page 12.)

[Kass 1988] Michael Kass, Andrew Witkin and Demetri Terzopoulos. *Snakes: Active contour models*. International Journal of Computer Vision, vol. 1, no. 4, pages 321–331, 1988. (Cited on page 8.)

[Kiefel 2014] Martin Kiefel and Peter Vincent Gehler. *Human pose estimation with fields of parts*. In European Conference on Computer Vision, pages 331–346, 2014. (Cited on page 81.)

[Kiefel 2015] Martin Kiefel, Varun Jampani and Peter Gehler. *Permutohedral Lattice CNNs*. In International Conference on Learning Representation Workshops, 2015. (Cited on page 82.)

[Kingma 2014] Diederik Kingma and Jimmy Ba. *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980, 2014. (Cited on page 87.)

[Kolmogorov 2004] Vladimir Kolmogorov and Ramin Zabin. *What energy functions can be minimized via graph cuts?* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 2, pages 147–159, 2004. (Cited on page 8.)

[Komodakis 2009] Nikos Komodakis, Nikos Paragios and Georgios Tziritas. *Clustering via LP-based Stabilities*. In Advances in Neural Information Processing Systems, pages 865–872, 2009. (Cited on pages 31 and 42.)

[Korc 2009] F. Korc and W. Forstner. *eTRIMS Image Database for Interpreting Images of Man-Made Scenes*. Technical report TR-IGG-P-2009-01, Dept. of Photogrammetry, University of Bonn, April 2009. (Cited on pages 1, 35 and 57.)

[Koutsourakis 2009] Panagiotis Koutsourakis, Loic Simon, Olivier Teboul, Georgios Tziritas and Nikos Paragios. *Single view reconstruction using shape grammars for urban environments*. In Proceedings of the IEEE Conference on Computer Vision, pages 1795–1802, 2009. (Cited on page 8.)

[Koziński 2014a] Mateusz Koziński and Renaud Marlet. *Image Parsing with Graph Grammars and Markov Random Fields*. In Winter Association on Computer Vision, 2014. (Cited on page 9.)

[Koziński 2014b] Mateusz Koziński, Guillaume Obozinski and Renaud Marlet. *Beyond procedural facade parsing: Bidirectional alignment via linear programming*. In Asian Conference on Computer Vision, 2014. (Cited on pages 9 and 50.)

[Koziński 2015a] Mateusz Koziński. *Segmentation of Facade Images with Shape Priors*. PhD thesis, Ecole des Ponts ParisTech, 2015. (Cited on page 5.)

[Koziński 2015b] Mateusz Koziński, Raghudeep Gadde, Sergey Zagoruyko, Renaud Marlet and Guillaume Obozinski. *A MRF Shape Prior for Facade Parsing with Occlusions*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. (Cited on pages 35, 50, 51, 58 and 59.)

[Krähenbühl 2012] Philipp Krähenbühl and Vladlen Koltun. *Efficient inference in fully connected CRFs with Gaussian edge potentials*. arXiv preprint arXiv:1210.5644, 2012. (Cited on pages 69, 80, 81, 86 and 93.)

[Krizhevsky 2012] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks*. In Advances in Neural Information Processing Systems, pages 1097–1105, 2012. (Cited on page 91.)

[Ladickỳ 2010] L'ubor Ladickỳ, Paul Sturgess, Karteek Alahari, Chris Russell and Philip HS Torr. *What, where and how many? Combining object detectors and CRFs*. In European Conference on Computer Vision, pages 424–437, 2010. (Cited on page 52.)

[Lafferty 2001] John D. Lafferty, Andrew McCallum and Fernando C. N. Pereira. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. In Proceedings of the International Conference on Machine Learning, pages 282–289, 2001. (Cited on page 79.)

[Lehman 2002] Eric Lehman and Abhi Shelat. *Approximation algorithms for grammar-based compression*. In Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, pages 205–212. Society for Industrial and Applied Mathematics, 2002. (Cited on page 13.)

[Li 2014] Yujia Li and Richard Zemel. *Mean-Field Networks*. arXiv preprint arXiv:1410.5884, 2014. (Cited on page 81.)

[Lin 2014a] M. Lin, Q. Chen and S. Yan. *Network In Network*. In International Conference on Learning Representations (ICLR), 2014. (Cited on page 80.)

[Lin 2014b] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár and C Lawrence Zitnick. *Microsoft COCO: Common objects in context*. In European Conference on Computer Vision, pages 740–755, 2014. (Cited on page 87.)

[Lin 2015] Guosheng Lin, Chunhua Shen, Ian Reid*et al.* *Efficient piecewise training of deep structured models for semantic segmentation*. arXiv preprint arXiv:1504.01013, 2015. (Cited on pages 81 and 89.)

[Liu 2015] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy and Xiaoou Tang. *Semantic image segmentation via deep parsing network*. In Proceedings of the IEEE Conference on Computer Vision, pages 1377–1385, 2015. (Cited on pages 81 and 89.)

[Long 2014] Jonathan Long, Evan Shelhamer and Trevor Darrell. *Fully convolutional networks for semantic segmentation*. arXiv preprint arXiv:1411.4038, 2014. (Cited on pages 2, 81 and 90.)

[Lowe 2004] David G Lowe. *Distinctive image features from scale-invariant keypoints*. International Journal of Computer Vision, vol. 60, no. 2, pages 91–110, 2004. (Cited on page 1.)

[Mäkinen 1989] E. Mäkinen. *On the subtree isomorphism problem for ordered trees*. Information Processing Letters, vol. 32, no. 5, pages 271–273, 1989. (Cited on page 28.)

[Manning 2011] Christopher D. Manning. *Part-of-speech Tagging from 97% to 100%: Is It Time for Some Linguistics?* In 12th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2011) - Volume Part I, pages 171–189. Springer-Verlag, 2011. (Cited on page 11.)

[Martinovic 2012] Andelo Martinovic, Markus Mathias, Julien Weissenberg and Luc Van Gool. *A three-layered approach to facade parsing*. In European Conference on Computer Vision, pages 416–429, 2012. (Cited on pages 3, 7, 8, 36, 37, 38, 45, 50, 51, 53, 58, 59, 63, 64 and 65.)

[Martinovic 2013a] Andelo Martinovic and Luc Van Gool. *Bayesian Grammar Learning for Inverse Procedural Modeling*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 201–208, 2013. (Cited on pages 3, 7, 8, 9, 11, 13, 14, 22, 24, 25, 36, 37, 38, 42, 49, 50, 58 and 67.)

[Martinović 2013b] Anđelo Martinović and Luc Van Gool. *Earley Parsing for 2D Stochastic Context Free Grammars*. Technical report KUL/ESAT/PSI/1301, KU Leuven, 2013. (Cited on page 11.)

[Martinovic 2015] Andelo Martinovic, Jan Knopp, Hayko Riemenschneider and Luc Van Gool. *3d all the way: Semantic segmentation of urban scenes from start to end in 3d*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4456–4465, 2015. (Cited on pages 3, 52, 54, 57, 63, 64 and 65.)

[Mathias 2015] Markus Mathias, Anđelo Martinović and Luc Van Gool. *ATLAS: A Three-Layered Approach to Facade Parsing*. International Journal of Computer Vision, 2015. (Cited on pages 51 and 58.)

[Matsuzaki 2005] Takuya Matsuzaki, Yusuke Miyao and Jun'ichi Tsujii. *Probabilistic CFG with Latent Annotations*. In 43rd Annual Meeting on Association for Computational Linguistics (ACL 2005), pages 75–82, 2005. (Cited on page 12.)

[Miller 1999] Philip Miller. Strong generative capacity. CSLI Publications, 1999. (Cited on page 25.)

[Mostajabi 2014] Mohammadreza Mostajabi, Payman Yadollahpour and Gregory Shakhnarovich. *Feedforward semantic segmentation with zoom-out features*. arXiv preprint arXiv:1412.0774, 2014. (Cited on page 82.)

[Müller 2006] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer and Luc Van Gool. *Procedural Modeling of Buildings*. In ACM SIGGRAPH 2006 / ACM Transactions on Graphics, pages 614–623, 2006. (Cited on page 7.)

[Nevill-Manning 1997] Craig G. Nevill-Manning and Ian H. Witten. *Identifying hierarchical structure in sequences: A linear-time algorithm*. Journal of Artificial Intelligence Research, pages 67–82, 1997. (Cited on page 11.)

[Nivre 2007] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov and E Marsi. *Malt Parser: A language-independent system for data-driven dependency parsing*. Natural Language Engineering, vol. 13, no. 2, pages 95–135, 2007. (Cited on page 12.)

[Nowozin 2010] Sebastian Nowozin, Peter V Gehler and Christoph H Lampert. *On parameter learning in CRF-based approaches to object class image segmentation*. In European Conference on Computer Vision, pages 98–111, 2010. (Cited on page 82.)

[Nowozin 2014] Sebastian Nowozin. *Optimal Decisions from Probabilistic Models: the Intersection-over-Union Case*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 548–555, 2014. (Cited on pages 59 and 61.)

[Ojala 2002] Timo Ojala, Matti Pietikäinen and Topi Mäenpää. *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002. (Cited on page 53.)

[Ok 2012] David Ok, Mateusz Koziński, Renaud Marlet and Nikos Paragios. *High-Level Bottom-Up Cues for Top-Down Parsing of Facade Images*. In 2nd Joint 3DIM/3DPVT Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012. (Cited on pages 11 and 21.)

[Osher 2003] Stanley Osher and Nikos Paragios. Geometric level set methods in imaging, vision, and graphics. Springer, 2003. (Cited on page 8.)

[Paris 2006] Sylvain Paris and Frédo Durand. *A fast approximation of the bilateral filter using a signal processing approach*. In European Conference on Computer Vision, pages 568–580, 2006. (Cited on page 84.)

[Parisot 2011] Sarah Parisot, Hugues Duffau, Stéphane Chemouny and Nikos Paragios. *Graph based spatial position mapping of low-grade gliomas*. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011, pages 508–515. Springer, 2011. (Cited on page 35.)

[Parisot 2012] Sarah Parisot, Hugues Duffau, Stéphane Chemouny and Nikos Paragios. *Graph-based detection, segmentation & characterization of brain tumors*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 988–995, 2012. (Cited on page 35.)

[Petrov 2007] Slav Petrov and Dan Klein. *Improved Inference for Unlexicalized Parsing*. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, pages 404–411. Association for Computational Linguistics, 2007. (Cited on page 12.)

[Ren 2015] C. Y Ren, V. A. Prisacariu and I. D Reid. *gSLICr: SLIC superpixels at over 250Hz*. arXiv preprint arXiv:1509.04232, 2015. (Cited on pages 83 and 88.)

[Riemenschneider 2012] Hayko Riemenschneider, Ulrich Krispel, Wolfgang Thaller, Michael Donoser, Sven Havemann, Dieter Fellner and Horst Bischof. *Irregular lattices for complex shape grammar facade parsing*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1640–1647, 2012. (Cited on pages v, 7, 8, 9, 35, 38, 39, 49, 50, 57, 59, 60 and 67.)

[Riemenschneider 2014] Hayko Riemenschneider, András Bódis-Szomorú, Julien Weissenberg and Luc Van Gool. *Learning where to classify in multi-view semantic segmentation*. In European Conference on Computer Vision, pages 516–532, 2014. (Cited on pages 52, 56, 57, 63, 64, 65, 66 and 67.)

[Ripperda 2006a] Nora Ripperda and Claus Brenner. *Reconstruction of façade structures using a formal grammar and RJMCMC*. In Pattern Recognition, pages 750–759. Springer, 2006. (Cited on pages 8 and 19.)

[Ripperda 2006b] Nora Ripperda and Claus Brenner. *Reconstruction of façade structures using a formal grammar and RjMCMC*. In DAGM, 2006. (Cited on page 67.)

[Rousseeuw 1987] Peter J Rousseeuw. *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis*. Journal of computational and applied mathematics, vol. 20, pages 53–65, 1987. (Cited on pages 33 and 34.)

[Russell 2008] Bryan C Russell, Antonio Torralba, Kevin P Murphy and William T Freeman. *LabelMe: a database and web-based tool for image annotation*. International Journal of Computer Vision, vol. 77, no. 1-3, pages 157–173, 2008. (Cited on page 57.)

[Sakakibara 1999] Yasubumi Sakakibara and Mitsuhiro Kondo. *GA-based learning of context-free grammars using tabular representations*. In Proceedings of the International Conference on Machine Learning, pages 354–360, 1999. (Cited on page 11.)

[Schwing 2015] Alexander G Schwing and Raquel Urtasun. *Fully connected deep structured networks*. arXiv preprint arXiv:1503.02351, 2015. (Cited on page 81.)

[Shotton 2006] Jamie Shotton, John Winn, Carsten Rother and Antonio Criminisi. *Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation*. In European Conference on Computer Vision, pages 1–15, 2006. (Cited on page 53.)

[Si 2013] Zhangzhang Si and Song-Chun Zhu. *Learning AND-OR Templates for Object Recognition and Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 9, pages 2189–2205, 2013. (Cited on page 10.)

[Simon 2011] Loic Simon, Olivier Teboul, Panagiotis Koutsourakis and Nikos Paragios. *Random exploration of the procedural space for single-view 3D modeling of buildings*. International Journal of Computer Vision, vol. 93, no. 2, pages 253–271, 2011. (Cited on pages 3, 8 and 11.)

[Simon 2012] Loic Simon, Olivier Teboul, Panagiotis Koutsourakis, Luc Van Gool and Nikos Paragios. *Parameter-free/Pareto-driven procedural 3D reconstruction of buildings from ground-level sequences*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 518–525, 2012. (Cited on pages 8, 11 and 19.)

[Sutton 1998] Richard S Sutton and Andrew G Barto. Introduction to reinforcement learning. MIT Press, 1998. (Cited on pages 19 and 20.)

[Szegedy 2014] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. *Going Deeper with Convolutions*. arXiv preprint arXiv:1409.4842, 2014. (Cited on page 80.)

[Teboul 2010a] Olivier Teboul. *Ecole centrale paris facades database*, 2010. (Cited on pages 51 and 56.)

[Teboul 2010b] Olivier Teboul, Loic Simon, Panagiotis Koutsourakis and Nikos Paragios. *Segmentation of building facades using procedural shape priors*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3105–3112, 2010. (Cited on pages 3, 37 and 43.)

[Teboul 2011a] Olivier Teboul. *Shape Grammar Parsing: Application to Image-based Modeling*. PhD thesis, Ecole Centrale Paris, 2011. (Cited on pages 17, 20, 21 and 23.)

[Teboul 2011b] Olivier Teboul, Iasonas Kokkinos, Loic Simon, Panagiotis Koutsourakis and Nikos Paragios. *Shape grammar parsing via reinforcement learning*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2273–2280, 2011. (Cited on pages v, 7, 8, 11, 12, 14, 17, 19, 20, 21, 22, 23, 24, 27, 35, 36, 37, 38 and 43.)

[Teboul 2011c] Olivier Teboul, Iasonas Kokkinos, Loïc Simon, Panagiotis Koutsourakis and Nikos Paragios. *Shape grammar parsing via reinforcement learning*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2273–2280, 2011. (Cited on pages 2, 3, 49, 50, 58, 67 and 68.)

[Teboul 2013] Olivier Teboul, Iasonas Kokkinos, Loic Simon, Panagiotis Koutsourakis and Nikos Paragios. *Parsing facades with shape grammars and reinforcement learning*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 7, pages 1744–1756, 2013. (Cited on pages 3 and 20.)

[Tenenbaum 2000] Joshua B Tenenbaum, Vin De Silva and John C Langford. *A global geometric framework for nonlinear dimensionality reduction*. Science, vol. 290, no. 5500, pages 2319–2323, 2000. (Cited on page 82.)

[Tighe 2010] Joseph Tighe and Svetlana Lazebnik. *Superparsing: scalable nonparametric image parsing with superpixels*. In European Conference on Computer Vision, pages 352–365, 2010. (Cited on page 52.)

[Tomita 1991] Masaru Tomita. *Parsing 2-Dimensional Language*. In Masaru Tomita, editor, Current Issues in Parsing Technology, volume 126 of *The Springer International Series in Engineering and Computer Science*, pages 277–289. Springer US, 1991. (Cited on page 11.)

[Tu 2005] Zhuowen Tu, Xiangrong Chen, Alan L Yuille and Song-Chun Zhu. *Image parsing: Unifying segmentation, detection, and recognition*. International Journal of Computer Vision, vol. 63, no. 2, pages 113–140, 2005. (Cited on page 2.)

[Tu 2008] Zhuowen Tu. *Auto-context and its application to high-level vision tasks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, 2008. (Cited on pages 2, 50, 52 and 53.)

[Tu 2010] Zhuowen Tu and Xiang Bai. *Auto-context and its application to high-level vision tasks and 3d brain image segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 10, pages 1744–1757, 2010. (Cited on pages 2 and 3.)

[Tu 2013] Kewei Tu, Maria Pavlovskaia and Song-Chun Zhu. *Unsupervised structure learning of stochastic and-or grammars*. In Advances in Neural Information Processing Systems, pages 1322–1330, 2013. (Cited on page 10.)

[Tylecek 2012] Radim Tylecek. *The CMP facade database*. Technical report, CTU–CMP–2012–24, Czech Technical University, 2012. (Cited on pages v, 35, 39 and 40.)

[Tylecek 2013] Radim Tylecek and Radim Sara. *Spatial Pattern Templates for Recognition of Objects with Regular Structure*. In German Conference on Pattern Recognition, 2013. (Cited on pages 49, 50, 51, 57, 59 and 60.)

[Valiente 2002] Gabriel Valiente. Algorithms on trees and graphs. Springer, 2002. (Cited on page 28.)

[Weissenberg 2013] Julien Weissenberg, Hayko Riemenschneider, Mukta Prasad and Luc Van Gool. *Is there a Procedural Logic to Architecture?* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 185–192, 2013. (Cited on pages 9, 11, 13, 14, 22, 23, 25, 27, 36, 37, 38, 39, 40, 41, 42 and 43.)

[Wolpert 1992] David H. Wolpert. *Stacked generalization*. Neural Networks, vol. 5, pages 241–259, 1992. (Cited on pages 50, 52 and 55.)

[Wonka 2003] Peter Wonka, Michael Wimmer, François Sillion and William Ribarsky. *Instant Architecture*. ACM Transactions on Graphics (TOG)., vol. 22, no. 3, pages 669–677, 2003. (Cited on page 15.)

[Yang 2011] Michael Ying Yang and Wolfgang 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. (Cited on pages 49, 50 and 51.)

[Yu 2015] Fisher Yu and Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. International Conference on Learning Representation (ICLR), 2015. (Cited on page 82.)

[Zaki 2002] Mohammed J Zaki. *Efficiently mining frequent trees in a forest*. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 71–80. ACM, 2002. (Cited on page 28.)

[Zeiler 2010] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor and Rob Fergus. *Deconvolutional networks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2528–2535, 2010. (Cited on pages 81 and 86.)

[Zheng 2015] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang and Philip H.S. Torr. *Conditional Random Fields as Recurrent Neural Networks*. In Proceedings of the IEEE Conference on Computer Vision, 2015. (Cited on pages 2, 80, 81, 87, 90 and 102.)

[Zhu 2009] Leo Zhu, Yuanhao Chen, Yuan Lin, Chenxi Lin and Alan L Yuille. *Recursive Segmentation and Recognition Templates for 2D Parsing*. In Advances in Neural Information Processing Systems, pages 1985–1992, 2009. (Cited on page 2.)